Opportunistic Routing in Low Duty-Cycle Wireless Sensor Networks

EUHANNA GHADIMI, KTH – Royal Institute of Technology
OLAF LANDSIEDEL, Chalmers University of Technology
PABLO SOLDATI, Huawei Technologies Sweden AB
SIMON DUQUENNOY, SICS Swedish ICT AB
MIKAEL JOHANSSON, KTH – Royal Institute of Technology

Opportunistic routing is widely known to have substantially better performance than unicast routing in wireless networks with lossy links. However, wireless sensor networks are usually duty cycled, that is, they frequently enter sleep states to ensure long network lifetime. This renders existing opportunistic routing schemes impractical, as they assume that nodes are always awake and can overhear other transmissions. In this article we introduce ORW, a practical opportunistic routing scheme for wireless sensor networks. ORW uses a novel opportunistic routing metric, EDC, that reflects the expected number of duty-cycled wakeups that are required to successfully deliver a packet from source to destination. We devise distributed algorithms that find the EDC-optimal forwarding and demonstrate using analytical performance models and simulations that EDC-based opportunistic routing results in significantly reduced delay and improved energy efficiency compared to traditional unicast routing. In addition, we evaluate the performance of ORW in both simulations and testbed-based experiments. Our results show that ORW reduces radio duty cycles on average by 50% (up to 90% on individual nodes) and delays by 30% to 90% when compared to the state-of-the-art.

Categories and Subject Descriptors: C.2.1 [**Network Architecture and Design**]: Wireless Communication; C.2.2 [**Network Protocols**]: Routing protocols

General Terms: Theory, Design, Algorithms, Performance, Experimentation

Additional Key Words and Phrases: Wireless sensor networks, energy efficiency, opportunistic routing, duty cycle, end-to-end delay

ACM Reference Format:
Euhanna Ghadimi, Olaf Landsiedel, Pablo Soldati, Simon Duquennoy, and Mikael Johansson. 2014. Opportunistic routing in low duty-cycle wireless sensor networks. ACM Trans. Sensor Netw. 10, 4, Article 67 (June 2014), 39 pages.
DOI: http://dx.doi.org/10.1145/2533686

1. INTRODUCTION

In Wireless Sensor Networks (WSNs), forwarding of packets to their intended destination is typically done in a two-step process: first, the routing protocol determines the next hop node using a routing metric, commonly based on the routing progress offered by neighboring nodes and wireless link estimation; second, the MAC protocol waits for the intended next-hop node to wake up and to successfully receive the packet.

This work was partially funded by the Swedish Research Council (VR) and the Swedish Foundation for Strategic Research (SSF).
Authors' addresses: E. Ghadimi (corresponding author), School of Electrical Engineering, KTH – Royal Institute of Technology, Sweden; e-mail: euhanna@kth.se; O. Landsiedel, Department of Computer Science and Engineering, Chalmers University of Technology, Sweden; P. Soldati, Huawei Technologies Sweden AB, Sweden; S. Duquennoy, The Swedish Institute of Computer Science (SICS) Swedish ICT AB, Sweden; M. Johansson, School of Electrical Engineering, KTH – Royal Institute of Technology, Sweden.

DOI: http://dx.doi.org/10.1145/2533686

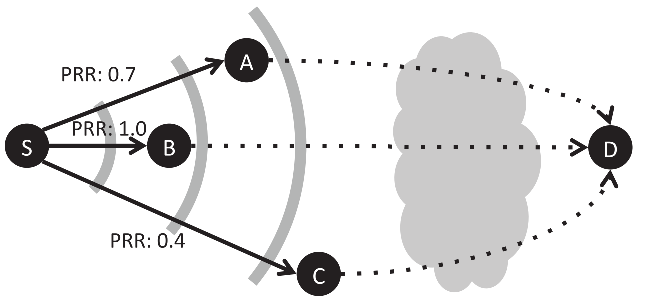


Fig. 1. Opportunistic routing in ORW: The first awoken neighbor (out of A to C) that successfully receives a packet from S and provides routing progress then forwards it to the destination D. ORW utilizes all neighbors that provide routing progress independent of link quality.

In this article, we depart from this unicast design paradigm. Instead, we transmit packets opportunistically in a fashion tailored to duty-cycled sensor networks: A packet is forwarded by the first awoken neighbor that successfully receives it and offers routing progress towards the destination (see Figure 1). As a result, we significantly improve energy efficiency, reduce end-to-end delay, and increase resilience to wireless link dynamics when compared to traditional unicast routing in WSNs.

1.1. Significance and Distinction

Low-power links in WSNs are highly dynamic [Srinivasan et al. 2008, 2010]. Link estimation [Woo et al. 2003; Fonseca et al. 2007] allows for WSN routing protocols (such as RPL [Winter et al. 2012] and CTP [Gnawali et al. 2009]) to restrict forwarding attempts to links of consistently high reliability, thereby ensuring stable topologies. In contrast, our opportunistic forwarding protocol explicitly utilizes all neighbors, that is, both stable and unstable links, for packet forwarding. As a result, we show significant improvements in terms of energy efficiency, delay, and resilience to link dynamics.

Originally, opportunistic routing [Larsson 2001; Choudhury and Vaidya 2004; Biswas and Morris 2005; Chachulski et al. 2007] was developed to improve throughput in multihop, mesh networks. These designs exploit the fact that, in wireless mesh networks, radios are always on and hence can overhear messages at practically no additional cost. However, sensor networks are commonly duty cycled to ensure long node and network lifetime, which limits the use of overhearing for opportunistic routing. Moreover, WSN applications demand high energy efficiency and low delays rather than high throughput. The main distinction of this work is that it adapts the concept of opportunistic routing to WSNs, accounting for critical system limitations and targeting the specific demands of sensor networks and their applications.

1.2. Contribution

This article has four contributions. First, it presents Opportunistic Routing in Wireless sensor networks (ORW). ORW adapts the concept of opportunistic routing to the particular requirements and challenges in WSNs by focusing on energy as a key metric and tailoring the design to duty-cycled nodes. Second, it introduces a detailed analytical model to compute the expected number of duty-cycled wakeups as an indicator of the average end-to-end delay in WSNs. Analytical insight from the model allows us to formulate a novel anycast routing metric that approximates the exact expressions for the expected number of duty-cycled wakeups. The metric has several attractive properties: it reduces both the radio-on time to reach a destination and the end-to-end packet delay, it enables an efficient construction of an anypath routing gradient, and it ensures the assignment of forwarders for opportunistic routing. We present a lightweight distributed routing algorithm and show that it is loop free and converges to optimality in a few iterations. Third, we introduce a lightweight, coarse-grained link estimator that reduces probe traffic and the required state information. It reflects the reduced requirements of opportunistic routing in terms of timeliness and accuracy in link estimation. Fourth, we present a practical realization of opportunistic routing and evaluate its benefits in both simulations and TinyOS-based testbed experiments. We show that ORW reduces radio duty cycles on average by 50% (up to 90% on individual nodes) and delays by 30% to 90% when compared to the state-of-the-art. Additionally, we show an increased stability to link dynamics and node failures. This work builds upon our two previous papers [Landsiedel et al. 2012; Ghadimi et al. 2012]. In the present work, besides several extensions, we align the theoretical and practical design perspectives of our previous papers and provide a unified overview of an opportunistic routing in WSNs with all its building blocks.

The remainder of this article is structured as follows. Section 2 provides the required background on opportunistic routing and introduces the basic concept of ORW. In Section 3, we tailor opportunistic routing to the specific demands of WSNs and detail on mechanisms for forwarder selection, anycast routing metric, and link estimation. We compare our design to the state-of-the-art in Section 4 and discuss the related work in Section 5. Section 6 concludes.

2. ORW DESIGN OVERVIEW

In this section, we provide the required background on opportunistic routing in mesh networks and discuss why it cannot be directly utilized in wireless sensor networks. Next, we introduce the basic concepts of our opportunistic routing scheme, motivate them by simple examples, and outline how they are tailored to the particular demands of wireless sensor networks.

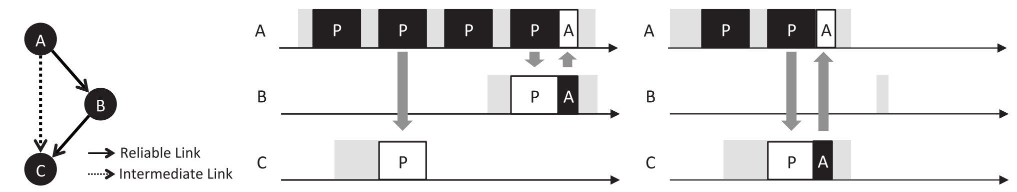
2.1. Preliminaries

Opportunistic routing [Biswas and Morris 2005; Chachulski et al. 2007; Larsson 2001; Choudhury and Vaidya 2004] improves network throughput in the context of multihop mesh and ad hoc networks. In contrast to traditional unicast routing, the underlying concept of opportunistic routing is to delay the forwarding decision until after the transmission, thereby exploiting the spatial diversity of the wireless channel. For example, in ExOR [Biswas and Morris 2005] each packet is addressed to a set of potential forwarding nodes, prioritized by routing progress. Based on their priority, each node in the forwarder set is assigned a time slot for forwarding, which it only utilizes if it did not overhear the packet being forwarded in an earlier time slot. This arbitration protocol allows ExOR to avoid duplicate forwarding.

A large forwarder set ensures that at least one of the forwarders successfully receives the packet with high probability, even if it contains unreliable links [Srinivasan et al. 2010; Biswas and Morris 2005; Chachulski et al. 2007]. This allows the routing scheme to utilize long-range but unstable links in the forwarder set which is beneficial, since these links often provide large routing progress. Typically, such links are avoided by unicast routing schemes due to their high dynamics [Gnawali et al. 2009; Alizai et al. 2009; Becher et al. 2008; Srinivasan et al. 2008]. Leveraging on the spatial diversity, opportunistic routing ensures high routing progress and limits the impact of link dynamics. This leads to a significant throughput improvement when compared to traditional routing schemes [Biswas and Morris 2005; Chachulski et al. 2007].

2.2. Opportunistic Routing in WSNs

Wireless sensor networks and their applications pose special requirements, such as low power consumption and severe resource constraints, that distinguish them from



(a) Sample topology: Node A reaches C via B on reliable links or directly on an unreliable link

(b) Traditional unicast routing in WSNs: Although C might overhear some transmission from A, packets are addressed to B to ensure stable routing

(c) Opportunistic routing in ORW: The first node that wakes up, receives a packet and provides sufficient routing progress acknowledges and forwards it

Fig. 2. Basic idea of ORW: Utilizing the first woken neighbor as forwarder, ORW reduces energy consumption and delay. This exploitation of spatial and temporal link diversity also increases resilience to link dynamics.

traditional multihop mesh networks. These requirements limit the direct applicability of existing opportunistic routing protocols in three key aspects.

Energy Efficiency vs. Throughput as Performance Metric. Opportunistic routing is designed to improve network throughput. However, WSN applications commonly demand reliable forwarding at high energy efficiency and not high throughput. In this article, we show how opportunistic routing can be adapted to improve energy efficiency compared to traditional WSN routing.

Duty-Cycled Radios. WSNs are commonly duty cycled to ensure long node and network lifetime. Hence, nodes are in deep sleep states with their radios turned off for most of the time. Duty cycling limits the number of nodes that concurrently overhear a packet (assuming no prior synchronization). As a result, it limits the spatial reuse in the forwarding process, one of the key benefits of opportunistic routing. However, we show in this article that opportunistic routing can be exploited to achieve low latency in duty-cycled networks: Instead of waiting for a given forwarder to wake up, our anycast primitive allows a node to send to the first awoken parent.

Low-Complexity Mechanisms for Unique Forwarder Selection. Opportunistic routing often relies on an arbitration protocol to select a unique forwarder among the receiving nodes. For example, each packet in ExOR contains a list of potential forwarders and their priorities and the receiver with the highest priority that successfully receives the packet shall forward it. However, due to the small packet size in sensor networks, such forwarder lists are not feasible. Similarly, assigning time slots to each potential forwarder poses implementation challenges. We introduce a lightweight algorithm for unique forwarder selection tailored to the resource constraints in WSNs.

In this article we argue that the concept of opportunistic routing, that is, delaying the decision of selecting a forwarder until the packet has been received, is well suited for the large node densities and high link dynamics in WSNs. However, many aspects of its realization need to be revisited and adapted to the specific requirements of WSNs.

2.3. Basic Idea of ORW

ORW targets duty-cycled protocol stacks. For simplicity we illustrate the basic concept of ORW in the context of an asynchronous low-power listening MAC, such as in X-MAC [Buettner et al. 2006][1]. In low-power listening, a sender transmits a stream of packets until the intended receiver wakes up and acknowledges it (see Figure 2(b)). To integrate opportunistic routing into duty-cycled environments, ORW departs from the traditional unicast forwarding scheme in one key aspect: the first node that: (a) wakes up, (b)

[1]The concepts in ORW are generic and apply also to both phase-locking and receiver-initiated schemes.

Table I. Design Overview

Design element	Section
Motivating EDC as the routing metric	3.1
Analytical model for EDC	3.2
Simplifying EDC to a practical metric	3.3
Forwarder Sets: selecting potential forwarders	3.4
Lightweight link estimation and neighbor discovery	3.5
Unique forwarder selection	3.6
System integration	3.7
Examples: EDC for sample topologies	3.8

receives the packet, and (c) provides routing progress then acknowledges and forwards the packet. For example, in Figure 2(a) Node A can reach Node C either directly via an unreliable link or via B. Traditional routing typically ignores the unreliable link $A \rightarrow C$ and relies on $A \rightarrow B \rightarrow C$ for forwarding. On the other hand, ORW also includes $A \rightarrow C$ into the routing process: If $A \rightarrow C$ is temporarily available and C wakes up before B, ORW utilizes it for forwarding. This efficiently reduces energy consumption and delay (see Figure 2(c)).

Our design enables an efficient adaptation of opportunistic routing to the specific demands of wireless sensor networks. (1) In contrast to opportunistic routing in mesh networks, forwarder selection in ORW focuses on energy efficiency and delay instead of network throughput: it minimizes the number of MAC-layer probes (retransmissions) until a packet is received by a potential forwarder. (2) ORW introduces a new anycast routing metric that reduces both the radio-on time to reach a destination and the end-to-end packet delay. (3) It integrates well into duty-cycled environments and ensures that many potential forwarders can overhear a packet within a single wakeup period. Therefore, ORW exploits spatial and temporal link diversity to improve resilience to wireless link dynamics. (4) The fact that only a small number of nodes receive a probe at a specific point in time simplifies the design of a coordination scheme to select a single forwarder. This reduces the overhead of control traffic.

3. DESIGN

After having introduced the basic idea of ORW, we now present its design in detail. We describe both algorithmic elements and system-level mechanisms and highlight how to realize the idea of anycast routing in duty-cycled WSNs (see Table I). We also address the differences between ORW and traditional unicast routing in WSNs and discuss key challenges such as stability and avoiding routing loops, duplicates, and asymmetric links. ORW consists of five main building blocks.

Analytical Model for Expected Duty-Cycled Wakeups. Traditional routing protocols for WSNs rely on the average number of end-to-end (re)transmissions as an indicator for the cost of delivery (refer to ETX or EAX [De Couto et al. 2003; Biswas and Morris 2005; Zhong and Nelakuditi 2007]). This is inefficient in duty-cycled WSNs, since these metrics do not take the essential factor of radio-on time into account. For this reason, the first element of our design is devoted to a framework that calculates the cost of forwarding in the presence of duty cycling and unreliable links. For this, we introduce the notion of Expected number of Duty-Cycled wakeups (EDC) as routing metric. This EDC routing metric acts as an indicator for both the required radio-on time to reach a destination and the end-to-end packet delay. Next, we develop a unified theoretical framework to study the delay in terms of the average number of wakeups required for end-to-end packet delivery (see Section 3.2). Later in Section 4, this model is used as a baseline for comparing different routing metrics.

EDC Metric. While our analytical model provides a complete picture of the expected number of wakeups required for the end-to-end packet delivery in duty-cycled WSNs, it is computationally demanding to evaluate. The EDC metric as the second building block of ORW serves as practical routing metric, simplifying the comprehensive analytical model while being able to find near-optimal routing solutions.

Forwarder Sets. Based on our routing metric EDC, each node selects a number of potential forwarders among its neighbors. Each node in this forwarder set is allowed to relay traffic of this sender. Including more neighbors in the forwarding set helps to reduce the required time until one of the potential forwarders wakes up and successfully receives. On the other hand, irrationally expanding the forwarding set increases the risk of choosing inefficient routes or even routing loops. We develop a policy to construct the forwarder set based on local knowledge and present formal proofs of optimality with respect to the EDC metric (see Section 3.4). Later, Section 4.1.3 shows that our EDC-based selection of forwarder sets is highly accurate when compared to the optimal routing solutions constructed via exhaustive searches on the full analytical model.

Link Estimator and Neighbor Discovery. As any other routing scheme, ORW requires mechanisms for neighbor discovery and link-quality estimation. However, ORW utilizes a pool of forwarders, where each packet takes its own, potentially different, route. Therefore, the influence of individual link and node failures on the overall performance is very limited when compared to traditional unicast routing schemes such as CTP. For this reason, we tailor a lightweight link estimator and neighbor discovery scheme that utilizes overheard packets for link estimation. As a result, ORW requires significantly less probing when compared to the state-of-the-art (see Section 3.5).

Unique Forwarder Selection. ORW implements a coordination protocol to prohibit multiple nodes from simultaneously relaying the same received packet. Based on the observation that a packet is commonly received only by a single forwarder in a duty-cycled environment, we introduce a lightweight, distributed mechanism that: (1) determines the number of receivers of a packet, and (2) ensures a unique forwarder in case of multiple receivers (see Section 3.6).

In the remainder of this section, we discuss the preceding building blocks in detail; see Table I. After the design elements have been explored, we present system integration of ORW in Section 3.7 and illustrate how routing topologies are built with EDC in Section 3.8.

3.1. Expected Number of Duty-Cycled Wakeups (EDC)

In ORW, a packet is forwarded by the first awoken neighbor that successfully receives it and provides routing progress. As a result, the routing topology towards a destination is not a tree anymore as in traditional unicast-based routing protocols. Instead, it is a Directed Acyclic Graph (DAG) with a single destination (such DAGs are often called Destination-Oriented DAGs, DODAGs). In this DODAG, ORW allows each packet to traverse on a different route to the destination (anycast). Note that DODAGs are sometimes used instead of trees even in unicast-based routing protocols, such as RPL [Winter et al. 2012]. In that case, a single parent is selected *before* transmitting any packet.

ORW introduces EDC (Expected Duty-Cycled wakeups) as a routing metric. EDC is an adaptation of ETX [De Couto et al. 2003] to energy-efficient, anycast routing in duty-cycled WSNs. In ORW, nodes use asynchronous duty cycling and low-power listening (as implemented by, e.g., X-MAC [Buettner et al. 2006] or BoX-Mac [Moss and Levis 2008]), and a forwarding node retransmits a packet repeatedly until a receiver wakes up and acknowledges it. In this setting, EDC represents the expected duration, that is, number of wakeups of duty-cycled nodes, until a packet has reached its intended destination, possibly across multiple hops.

The EDC routing metric is an indicator of both the radio-on time to reach a destination and the end-to-end packet delay. Thus, choosing routes with low EDC leads to reduced energy consumption and small delays, both key metrics in sensor networks. Multiple routing choices offered by anycast routing decreases the waiting time until one of the potential forwarders wakes up and successfully receives the packet, thereby lowering the end-to-end delay and overall energy consumption. In the following we introduce an analytical model for EDC and show that it depends on two key parameters: the number of potential forwarders and the quality of the wireless links to these. We then derive a simplified model for EDC that reflects the resource constraints of wireless sensor nodes.

3.2. Analytical Model for Expected Number of Duty-Cycled Wakeups

In this section, we introduce an analytical model for our anycast routing metric EDC. We represent the topology of the network as a labeled directed graph $\mathcal{G} = \{\mathcal{N}, \mathcal{L}, \mathcal{P}\}$ with node set $\mathcal{N} = \{1, 2, \ldots, |\mathcal{N}|\}$, link set $\mathcal{L} \subseteq \{(i, j) \mid i \in \mathcal{N}, j \in \mathcal{N}\backslash i\}$, and link parameters $\mathcal{P} = \{p(i, j) \mid (i, j) \in \mathcal{L}\}$. The link parameters model the success probabilities on links. Specifically, the analytical model assumes that the packet-loss process on each link $(i, j) \in \mathcal{L}$ follows a Bernoulli process with success probability $p(i, j)$, and is independent of the packet-loss processes on the other links in the network.

We define the forwarder sets $\mathcal{F}(i) \subseteq \mathcal{N}(i)$ of a node i as the subset of its neighbor sets $\mathcal{N}(i)$ that are selected to relay i's packets. For the moment, we assume that the forwarder set of each node is fixed. Later, in Section 3.4, we address how individual nodes can perform this forwarder selection. Further, we will assume that nodes use the same radio-on time length T. At the n-th wakeup and before going to dormant state, node i draws the next wakeup time t_n uniformly in the interval $[0, T]$; that is, node i wakes up at $t_0, T + t_1, 2T + t_2$, etc. In this setting, we are interested in analyzing the expected number of duty-cycled wakeups for a transmission from source to destination. To this end, let DC be the random number of duty-cycled wakeups required to complete the end-to-end transmission, and note that we can divide it into two independent components: first, the number of wakeups required for a single-hop transmission from the source node to one of its forwarders, and second, the number of wakeups the packet takes to reach the sink from the forwarder node. Since link losses are assumed independent, we can write

$$\mathbf{E}\{\mathrm{DC}(i)\} = \mathbf{E}\{\mathrm{DC}_s(i)\} + \mathbf{E}\{\mathrm{DC}_m(i)\}, \tag{1}$$

where $\mathbf{E}\{\mathrm{DC}_s(i)\}$ is the expected number of duty-cycled wakeups required for the single-hop transmission; that is, until the packet has been received by one of the forwarders, and $\mathbf{E}\{\mathrm{DC}_m(i)\}$ is the expected number of remaining duty-cycled wakeups it takes to complete the multihop transmission. The number of wakeups required for a single-hop transmission can be seen as the sum of two independent random variables, $\mathrm{DC}_s(i) = X_s(i) + Y_s(i)$ where $X_s(i)$ is the number of failed intervals (in which all forwarders wake up and fail to receive the transmission from node i) and $Y_s(i)$ is the waiting time within a wakeup period of a node containing the successful transmission. A failed interval requires that all transmissions between i and $j \in \mathcal{F}(i)$ fail, hence $X_s(i)$ follows a geometric distribution with pdf

$$\Pr\{X_s(i) = \tau\} = \prod_{j\in\mathcal{F}(i)} (1 - p(i, j))^{\tau} \left(1 - \prod_{j\in\mathcal{F}(i)} (1 - p(i, j))\right)$$

and mean

$$\mathbf{E}\{X_s(i)\} = \sum_{\tau=0}^{\infty} \tau \Pr\left\{X_s(i) = \tau\right\} = \frac{\prod_{j\in\mathcal{F}(i)}(1-p(i,j))}{1-\prod_{j\in\mathcal{F}(i)}(1-p(i,j))}. \tag{2}$$

To characterize $Y_s(i)$, note that for node $j \in \mathcal{F}(i)$ to be the forwarder: (1) it must experience a successful transmission and (2) transmissions to all other nodes that wake up before j must fail. Moreover, the underlying event is conditioned on that at least one node will successfully receive in the current duty cycle. Hence, the probability of reception for $j \in \mathcal{F}(i)$ in an arbitrary duty cycle conditioned on at least one reception is given by

$$p_s(j) = \frac{p(i,j)}{1-\prod_{l\in\mathcal{F}(i)}(1-p(i,l))}. \tag{3}$$

In general, this event can happen for $2^{|\mathcal{F}(i)|-1} - 1$ cases which is based on exploring the probabilities of having all nodes included in the possible subsets of forwarders $\{\forall f \subset \mathcal{F}(i)\backslash j\}$ that have failed before node j. Let $\mathcal{F}_{k\backslash j}$ denote the set of all the subsets of forwarders of node i with cardinality equal k not containing node j. The probability of having exactly k failed transmissions excluding node j is given by

$$p_f(k\backslash j) = \sum_{l\in\mathcal{F}_{k\backslash j}} \prod_{m\in l}(1-p(i,m)). \tag{4}$$

Due to continuity of the random variable, the probability of having two nodes with the same activation time is zero. Thus, the mean waiting time is achieved by iterating among all the nodes $j \in \mathcal{F}(i)$ with i.i.d. (independently and identically distributed) uniform wakeups and different link qualities:

$$\mathbf{E}\{Y_s(i)\} = \frac{1}{T}\int_0^{\infty} x \sum_{j\in\mathcal{F}(i)} \sum_{k=0}^{|\mathcal{F}(i)|-1} \frac{p_s(j)}{T} p_f(k\backslash j)\left(\frac{x}{T}\right)^k \left(\frac{T-x}{T}\right)^{|\mathcal{F}(i)|-k-1} dx.$$

Since $\mathbf{E}\{Y_s(i)\}$ does not have dimension (it is a portion of duty cycle), we normalize the previous equation by dividing it by T. As an example consider node i with n forwarders with the same success probabilities $p(i,j) = 1$. It turns out that in this case, $\mathbf{E}\{\mathrm{DC}_s(i)\}$ has a closed form. Note that $\mathbf{E}\{X_s(i)\} = 0$ and $\mathbf{E}\{\mathrm{DC}_s(i)\} = \mathbf{E}\{Y_s(i)\} = \frac{1}{T}\int_0^T n\frac{x}{T}(\frac{T-x}{T})^{n-1}dx = \frac{1}{n+1}$. The mean single-hop waiting time decreases hyperbolically with increasing number of neighbors.

On the other hand, $\mathbf{E}\{\mathrm{DC}_m(i)\}$ is the expected number of duty-cycled wakeups it takes to send the packet from the forwarder set $\mathcal{F}(i)$ to the sink, given that a successful transmission has already taken place between i and a node in $\mathcal{F}(i)$. Hence, $\mathbf{E}\{\mathrm{DC}_m(i)\}$ is given by

$$\mathbf{E}\{\mathrm{DC}_m(i)\} = \sum_{j\in\mathcal{F}(i)} \Pr\{j \text{ is the forwarder}\}\mathbf{E}\{\mathrm{DC}(j)\}, \tag{5}$$

where

$$\Pr\{j \text{ is the forwarder}\} = \sum_{k=0}^{|\mathcal{F}(i)|-1} \frac{1}{\binom{|\mathcal{F}(i)|-1}{k}} \sum_{l\in\mathcal{F}_{k\backslash j}} \prod_{m\in l}\left(1-p(i,m)\right)\frac{p_s(j)}{|\mathcal{F}(i)|}.$$

In words, this corresponds to the probability of node j being the first successful receiver, given that at least one forwarder receives in the current duty cycle. Up to now, we have developed an analytical framework to measure the cost of packet delivery for each sender in terms of the average number of duty cycles. With the exception of a few cases—like when all link reliabilities are equal to 1—the analysis and even simulations tend to be intractable with respect to increased network density. Thus, the expected number of duty-cycled wakeups $\mathbf{E}\{\mathrm{DC}(i)\}$ cannot be used for selecting the optimal forwarder set in practical opportunistic routing protocols.

Let us quickly summarize the assumptions that we make in this model. We assume that: (1) packet-loss probabilities are independent of each other (this assumption is common for routing metrics [De Couto et al. 2003]); (2) all nodes have the same length of wakeup interval, T, and consequently, in our model we compute the portions of wakeup interval instead of the actual delay in seconds; and (3) that the wakeup times are continuous and hence there are no multiple receivers. However, in reality even though nodes wake up in different times, their ACKs can still collide. Later, in Section 3.6.1 we account for the possibility of having multiple receivers.

3.3. Simplifying Expected Duty-Cycled Wakeups as Routing Metric

We have argued that an accurate evaluation of the expected number of duty-cycled wakeups under opportunistic forwarding is quite complex, even when the forwarder sets are fixed. While the protocol does not need an analytical formula for the expected number of wakeups, it does need a procedure for selecting the optimal forwarder sets. To this end, we need to define a lightweight metric that captures the essential features of opportunistic forwarding, yet allows us to develop provably correct algorithms for distributed forwarder set selection.

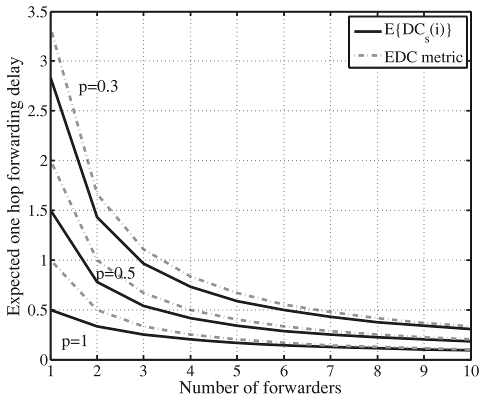
3.3.1. Introducing EDC as Routing Metric. We have found that a metric, which we call EDC, strikes an appealing balance between effectiveness in the selection of the forwarder set and the simplicity of analysis. The EDC of a node i can be computed recursively via

$$\mathrm{EDC}(i) = \frac{1}{\sum_{j\in\mathcal{F}(i)} p(i,j)} + \frac{\sum_{j\in\mathcal{F}(i)} p(i,j)\cdot \mathrm{EDC}(j)}{\sum_{j\in\mathcal{F}(i)} p(i,j)} + w. \tag{6}$$

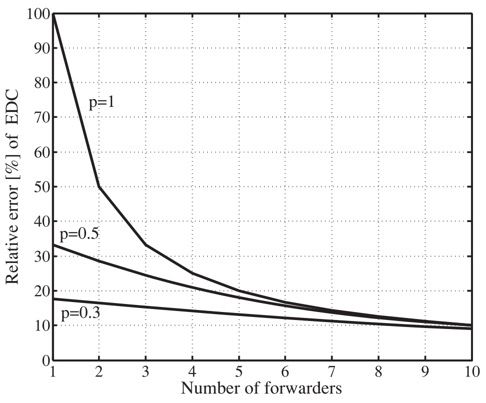
The first term is per-hop cost of forwarding because it approximates the expected one-hop forwarding delay $\mathbf{E}\{\mathrm{DC}_s(i)\}$ while the second term attempts to capture the essential features of the subsequent delay $\mathbf{E}\{\mathrm{DC}_m(i)\}$ from forwarders to the sink. The third term, $w \geq 0$, adds a weight to reflect the cost of forwarding. Please note that for single-path forwarding and $w = 0$, EDC equals to ETX and leads to the same topology. As a result, we view EDC as an extension of ETX to anycast routing.

The per-hop cost of forwarding with EDC shares many important features of the analytical model: it is a hyperbolic function of the link reliabilities of the forwarders, and adding a forwarder or increasing the link reliability decrease the single-hop cost. Figure 3 illustrates a situation with homogeneous links ($p(i,j) = p$ for all j) and compares $\mathbf{E}\{\mathrm{DC}_s(i)\}$ with the proposed hyperbolic approximation in the EDC metric. We see that the analytical model and approximation tend to agree with each other when the number of forwarders increases.

In the analytical model, the complexity of the multihop cost comes from the fact that the probability of j being a forwarder depends on the link reliabilities of the other nodes (the probability of a node j being a forwarder not only depends on that it could successfully receive the transmission when it is awake, but it also depends on the probability that nodes that woke up earlier all failed to receive the packet from i). The EDC metric simply assumes that the probability that j is a forwarder is directly proportional to $p(i,j)$. With this assumption, the probability of being forwarder for each node j is independent of others. However, since the probability of forwarders must add up to one, we normalize each individual forwarding probability. Hence, in the EDC metric, the probability that node j with reliability $p(i,j)$ being forwarder



(a) $\mathbf{E}\{\mathrm{DC}_s(i)\}$ versus EDC metric



(b) Relative error of EDC metric with respect to the analytical model $\mathbf{E}\{\mathrm{DC}_s(i)\}$

Fig. 3. Comparison of the analytical model versus EDC metric for $\mathcal{F}(i) = [1, 10]$ forwarders. The success probabilities p are equal for each forwarder. The plot shows that the relative error of EDC metric with respect to the analytical model $\mathbf{E}\{\mathrm{DC}_s(i)\}$ is bounded by 10% for the practical numbers of forwarders in WSNs.

for node i is $p(i, j)/(\sum_{j\in\mathcal{F}(i)} p(i, j))$. Our evaluation results in Section 4.1.3 confirm the accuracy of the EDC metric compared with the analytical scheme under realistic network scenarios.

3.3.2. Cost of Forwarding: w. We include a cost of forwarding, w, in EDC for a simple and practical reason. Processing and forwarding a packet consumes energy on a sensor node and adds delay. Thus, each additional hop increases both delay and energy consumption. As a routing metric, EDC models energy and delay and we argue that it is important to also include the cost of forwarding a packet into the routing metric.

We model w as a constant value and it describes the cost of forwarding a packet over one hop (see Eq. (6)). It is important to note that the choice of w impacts the resulting routing topology. Augmenting w increases the routing progress that a forwarding neighbor j of a node i is required to provide to be included in the forwarder set $\mathcal{F}(i)$. We discuss forwarder sets in detail in Section 3.4 where we argue that increasing w leads to a smaller forwarder set. Based on different choices of w, one faces the following trade-offs.

(1) A high value of w limits the forwarding to nodes that provide high routing progress and leads to fewer hops until a packet reaches the destination.
(2) However, reducing the size of the forwarder set increases delay and energy consumption for packet delivery.
(3) A large forwarder set increases the resilience to link dynamics. The more links can be utilized, the lower the probability that none of them is available at a single point in time [Srinivasan et al. 2010; Biswas and Morris 2005; Chachulski et al. 2007].
(4) Although we demonstrate in Section 3.4.1 that the EDC metric leads to loop-free routing topologies, in practical systems a too-low choice of w increases the risk of *temporary* routing loops, as packets are forwarded by nodes that provide even minimal routing progress (see Sections 3.4.2 and 4.2).

Given these effects, a proper tuning of w allows ORW to balance delay and energy with routing progress and stability. In terms of topology stability, w also plays a similar role as the parent-switch threshold of unicast routing protocols such as CTP and RPL: both aim to ensure stable routing topolgies and avoid loops. Please note that w is independent of the actual routing metric that is in use. For example, it is applicable to traditional unicast routing based on ETX in a similar way. By default, ETX merely focuses on link reliability and not the forwarding cost. However, with adding w to ETX, one can extend it to include the cost of forwarding. As a result, paths with fewer hops would be preferred over paths with many hops in order to reduce the cost of processing, etc.

In our evaluation in Section 4.2, we determine a range of values for w that ensure both stable and efficient routing with EDC. From this we choose a default configuration that provides both high performance and high stability. We show that this default choice is independent from individual deployments and holds across all our evaluation scenarios.

3.4. Forwarder Sets

Based on the routing metric EDC, each node selects a number of potential forwarders among its neighbors. Each node in this forwarder set is allowed to relay traffic of this sender. Two key factors impact the forwarder set: (1) adding more neighboring nodes to the forwarder set reduces the time until one of the potential forwarders wakes up to receive. Hence, it decreases the single-hop EDC of the node and improves spatial diversity. However, (2) adding too many neighboring nodes to the forwarder set may decrease its average routing progress, as typically not all neighbors provide good progress (see Eq. (6)).

In this section, we describe how to maintain the forwarder set and the EDC metric in a network. Our key contribution is a distributed algorithm for forwarder selection that minimizes the EDC of each node. In particular, we show that, after the appropriate ordering of potential forwarders, nodes can use a greedy algorithm to find the optimal forwarder set, and that this algorithm ensures loop-free topologies.

3.4.1. Distributed, Optimal Forwarder Selection. In the forwarder selection problem, each node i is given a set of potential forwarders $\mathcal{N}(i)$, their EDC metrics EDC(j) for $j \in \mathcal{N}(i)$, and the probability $p(i, j)$ of successful transmission from i to $j \in \mathcal{N}(i)$. Each node shall determine the subset of forwarders $\mathcal{F}^\star(i) \subseteq \mathcal{N}(i)$ that minimizes EDC(i). To develop our algorithm, we first establish some basic properties of the EDC metric.

LEMMA 3.1. *Let $p(i, j) \geq 0$ and $c_j > 0$ for every $j \in \mathcal{N}(i)$. Define the set functions $f_i^{(1)} : 2^{|\mathcal{N}(i)|} \mapsto \mathbf{R}$ and $f_i^{(2)} : 2^{|\mathcal{N}(i)|} \mapsto \mathbf{R}$ with $f_i^{(1)}(\emptyset) = \infty$ and $f_i^{(2)}(\emptyset) = \infty$ as*

$$f_i^{(1)}(\mathcal{A}) = \frac{1}{\sum_{j \in \mathcal{A}} p(i, j)}, \quad f_i^{(2)}(\mathcal{A}) = \frac{\sum_{j \in \mathcal{A}} c_j \cdot p(i, j)}{\sum_{j \in \mathcal{A}} p(i, j)},$$

for $\mathcal{A} \subseteq \mathcal{N}(i)$. Then $f_i^{(1)}(\mathcal{A})$ is strictly decreasing in $p(i, j)$ and $f_i^{(2)}(\mathcal{A})$ is strictly increasing in c_j.

PROOF. See appendix for this and all other proofs. □

In the preceding lemma, c_j in $f_i^{(2)}(\mathcal{A})$ corresponds to EDC(j) in the definition of the EDC metric. Let $f_i(\mathcal{A}) = f_i^{(1)}(\mathcal{A}) + f_i^{(2)}(\mathcal{A}) + w$, then the following lemma presents a necessary and sufficient condition for when the insertion of a new member $k \in \mathcal{N}(i) \backslash \mathcal{A}$ in the forwarder set decreases EDC(i).

LEMMA 3.2. *Let $p(i, k) > 0$ then $f_i(\mathcal{A} \cup \{k\}) < f_i(\mathcal{A})$ if and only if $c_k < f_i(\mathcal{A}) - w$.*

Next, we observe the behavior of $f_i(\mathcal{A})$ after adding a new member into the forwarding set $\mathcal{F}(i)$.

LEMMA 3.3. *If $c_k < f_i(\mathcal{A})$ and $p(i, k) > 0$ then $f_i(\mathcal{A} \cup \{k\}) > c_k$.*

Up to now, we have concluded that it is beneficial for node i to add a new neighbor k to the forwarder set if its EDC is less than the EDC of node i minus the positive offset w. Moreover, after adding node k, the updated EDC of node i is greater than EDC(k). The next theorem characterizes the optimum forwarder set of node i.

THEOREM 3.4. *Let π be an ordering of the nodes in $\mathcal{N}(i)$ such that $\{c_{\pi(1)} \leq c_{\pi(2)} \leq \cdots \leq c_{\pi(|\mathcal{N}(i)|)}\}$. Then, the optimal forwarder set that minimizes the EDC value is $\mathcal{F}^\star(i) = \{\pi(1), \ldots, \pi(k)\}$ where k satisfies $c_k < f_i(\mathcal{F}^\star(i)) - w$ and $c_{k+1} > f_i(\mathcal{F}^\star(i)) - w$.*

Given a network topology $\mathcal{G} = \{\mathcal{N}, \mathcal{L}, \mathcal{P}\}$, Lemmas 3.2 and 3.3 and Theorem 3.4 suggest a greedy algorithm to compute the EDC metric in the network and to locally construct the set $\mathcal{F}^\star(i)$ at each node. Starting from the sink with EDC(sink) $= 0$, each node i in the network sorts its neighbors in the set $\mathcal{N}(i)$ in increasing order of their EDCs. A potential forwarder $j \in \mathcal{N}(i)$ is added to the set of forwarders $\mathcal{F}^\star(i)$ of node i if EDC(j) $<$ EDC(i) $- w$, upon which EDC(i) is updated on the basis of the new set $\mathcal{F}^\star(i) \cup \{j\}$. The procedure repeats until the forwarding list and the EDC values of all the nodes remain unchanged. This procedure is described in Algorithm 1. The stopping criterion of the algorithm is distributed in the sense that each node stops updating its own EDC value (and its forwarder set) when all of its neighbors EDC values remain unchanged.

One may ask what happens if two or more neighbors have equal EDC but different link reliability. The next lemma shows that Algorithm 1 first picks the node with higher reliability.

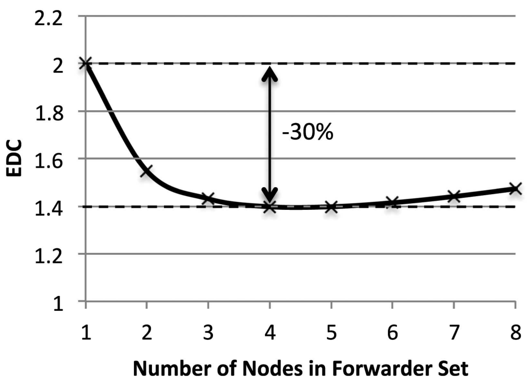
LEMMA 3.5. *If $c_k = c_{k'} < f_i(\mathcal{A}) - w$ and $p(i,k) > p(i,k') > 0$ then*

$$f_i(\mathcal{A} \cup \{k\}) < f_i(\mathcal{A} \cup \{k'\}) < f_i(\mathcal{A}).$$

Under the hypotheses of Algorithm 1 and Lemma 3.5, the optimal forwarder selection procedure first picks the node k because it has a higher link reliability compared to k'. After adding k into the optimal forwarding set, namely, $\{k\} \in \mathcal{F}^\star(i)$, by Lemma 3.3 we have $c'_k = c_k < f_i(\mathcal{F}^\star(i))$. Now, whether we should include k' in the optimal forwarder set or not depends on the value of w. This is due to the forwarder insertion criterion specified in Lemma 3.2 that requires $c_{k'}$ to be less than $f_i(\mathcal{F}^\star(i)) - w$. Note that for $w = 0$, such requirement is automatically satisfied by Lemma 3.3 and the optimal forwarder set either includes both k and k' or none of them.

ALGORITHM 1: EDC-based opportunistic routing.

```
1: Input: $\mathcal{G} = \{\mathcal{N}, \mathcal{L}, \mathcal{P}\}$, neighbor set $\mathcal{N}(i)\ \forall i \in \mathcal{N}$.
2: Initialize: EDC(Sink) $\leftarrow 0$, EDC($i$) $\leftarrow \infty$.
3: repeat
4:    Set $\mathcal{F}^\star(i) = \emptyset\ \forall i \in \mathcal{N}$
5:    for all $i \in \mathcal{N}$ do
6:       Sort $\mathcal{N}(i) = \{n_1, \ldots, n_{|\mathcal{N}(i)|}\}$ s.t. EDC($n_j$) $<$ EDC($n_{j+1}$).
7:       for $j = 1 \to |\mathcal{N}(i)|$ do
8:          if EDC($j$) $<$ EDC($i$) $- w$ then
9:             a. Update: $\mathcal{F}^\star(i) = \mathcal{F}^\star(i) \cup \{n_j\}$.
10:            b. Update: EDC($i$) based on eq. (6).
11:         end if
12:      end for
13:   end for
14: until EDC of all nodes are unchanged.
```



Node	EDC	Use
1	1	✔
2	1.1	✔
3	1.2	✔
4	1.3	✔
5	1.4	✗
6	1.5	✗
7	1.6	✗
8	1.7	✗

Fig. 4. Example: Optimal forwarder set. Employing the practical algorithm discussed in Section 3.4.2, the forwarding node computes an EDC of 1.4 for itself and includes 4 neighbors with an EDC strictly lower than 1.4 in its forwarder set. Smaller or larger forwarder sets would lead to a worse EDC. When compared to single-path routing, it reduces the number of expected wakeup periods to reach the sink by 30%. For simplicity, PRR is 1 and w is 0 in this example.

Each iteration t of Algorithm 1 produces a new routing topology $\mathcal{R}^{(t)} = \{\mathcal{N}, \mathcal{L}^{(t)}, \mathcal{E}^{(t)}\}$ where $\mathcal{L}^{(t)}$ consists of links (i, j) from a node i to all its forwarders $j \in \mathcal{F}^\star(i)$, and $\mathcal{E}^{(t)}$ is a network-wide set of updated EDC. We next prove that each routing topology $\mathcal{R}^{(t)}$ is loop free.

LEMMA 3.6. *Any routing topology $\mathcal{R}^{(t)} = \{\mathcal{N}, \mathcal{L}^{(t)}, \mathcal{E}^{(t)}\}$ produced by Algorithm 1 is a directed acyclic graph (DAG).*

Our next result states that an opportunistic routing algorithm converges in a finite number of steps.

THEOREM 3.7. *If $\mathcal{G} = \{\mathcal{N}, \mathcal{L}, \mathcal{P}\}$ remains constant through the execution of Algorithm 1, the algorithm terminates after at most $|\mathcal{N}|$ passes of the outer loop.*

Later in Section 4.2.5, we show that the EDC construction algorithm converges fairly fast. In the early iterations of the algorithm, the nodes closer to the sink will stabilize their EDC values. Afterward, nodes that are located far from the sink start to add closer ones in their forwarders and reach the optimality of the metric; this procedure propagates to the entire network.

3.4.2. Practical Considerations and Discussion. In practice, the forwarder set $\mathcal{F}(i)$ is computed by adding neighboring nodes sorted by their EDC (starting with the lowest EDC) to the forwarder set and determining the set with the minimum EDC (see example in Figure 4). $\mathcal{F}(i)$ defines the EDC(i) of a node i and all neighboring nodes j that provide routing progress, that is, EDC(j) < EDC(i) − w, are employed as potential forwarders. As a node only selects nodes that provide strictly more progress than itself, the resulting topology forms a loop-free graph, namely, is a DAG (refer to Lemma 3.6).

However, a slow spreading of updates, such as new EDC values due to changed link reliability of neighboring nodes, can lead to temporary loops until an update reaches all neighbors. This is common in routing protocols and not specific to ORW. We address this issue with three standard techniques: (1) when a parent node is downgraded to a child node, a node observes this and forwards its packets without dropping duplicates. Additionally, these packets are delayed shortly to allow the topology to stabilize. (2) A TTL field in each packet avoids infinite loops. (3) The forwarding cost w (see Section 3.3.2) ensures a threshold between forwarding nodes to avoid oscillating packets.

This design is in contrast to traditional opportunistic routing, such as ExOR or MORE [Chachulski et al. 2007] in the following key aspect: opportunistic routing protocols typically maintain a prioritized forwarder set in the packet header while in ORW all nodes providing routing progress potentially forward the packet. This leads to two important benefits: (1) instead of long address lists in the packet header denoting a forwarder set, which is not feasible in resource-constrained WSNs, a single value describes the minimum EDC that a forwarder must provide. Upon receiving a message, each potential forwarder decides whether it provides sufficient routing progress or not; (2) it allows ORW to utilize spurious neighbors and neighbors it did not yet discover for forwarding. These benefits enable ORW to better adapt to the link dynamics that are common in low-power, wireless networking compared to other alternatives.

3.5. Link Estimation and Discovery

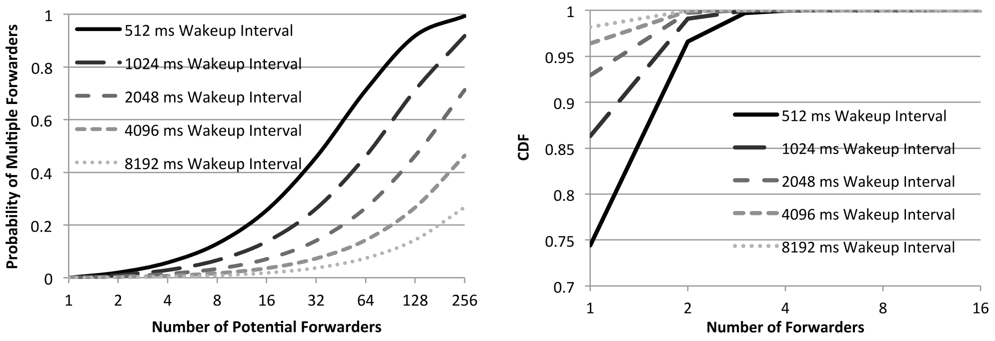
Anycast routing in ORW utilizes a pool of forwarders that allows each packet to potentially travel on a different route. Hence, when links to individual forwarders temporary fail or show reduced reliability, their impact on the overall performance and the quality of the forwarder set is limited. As a result, ORW does not require up-to-date estimates to each candidate forwarder, as is typical in traditional unicast routing.

Hence, we design link estimation and neighbor discovery in ORW to fulfill these specific demands. It mainly relies on overhearing: When a duty-cycled node in ORW wakes up to check for energy on the channel and subsequently receives a packet, it: (1) forwards it when providing routing progress, and (2) it updates its link-quality estimate. For link estimation, a node maintains the link-reception ratio from each neighbor. To this end, packets in ORW contain a header field that denotes the average rate at which a node is forwarding data. The link quality is obtained by dividing the rate of packets overheard from a neighbor by the forwarding rate of the same neighbor noted in the header field. As ORW operates with large forwarder sets and targets coarse-grained link estimation, we argue that individual, asymmetric links have limited impact on the estimation and simplify link estimation by assuming $p(i, j) = p(j, i)$. This design deviates from the traditional link estimation used for unicast routing in WSNs in the following two key points.

Stability. While agility is a key design criterion for modern link estimators as they shall adapt quickly to changes in link quality, ORW may not even recognize when a link temporarily fails, assuming it utilizes multiple links for forwarding. This is a design goal: as long as the aggregate of the neighbors performs stably, the dynamics of individual links will be masked. Aging slowly removes broken links from the forwarder set and neighbor table.

Limited Use of Probing. Traditional link estimation employs probing to determine the link qualities to neighboring nodes. In contrast, ORW mainly relies on overhearing during wakeups to update its neighbor table and link estimates. Probing is only used when not a single route is available. In our evaluation, we show that the routing topology in ORW converges quickly after boot-up and node failures without extensive probing. This reduces the overhead of control traffic in ORW.

Eventually, the bootstrap of a network using ORW is as follows: First, the nodes with no known parent probe their neighbors via a broadcast. The probing node i receives responses from a subset of its neighbors, depending on the link quality. For each received response from node j, node i adds the following entry to its neighbors table: (j, EDC(j), $p(i, j)$), where $p(i, j) = 1$. The link-quality estimation, entirely based on overhearing, will refine the value of $p(i, j)$ and may add new entries to the neighbor table, when overhearing a neighbor that did not answer to the original probe. Node i won't use probing anymore, unless all its entries reach an estimated link quality of 0.



(a) Probability: Probability that multiple forwarders receive the same packet for typical wakeup periods in ORW. For example, for 16 neighbors that provide routing progress, this probability is between 2% and 26% depending on the wakeup period. Please note the log scale with base 2 on the x-axis.

(b) CDF: Cumulative distribution for the probability of multiple forwarders plotted for 16 potential forwarders. The plots indicate that if we have multiple forwarders, then these are commonly only two. The probability to have three and/or even more forwarders receiving the packet is very low. Please note the log scale with base 2 on the x-axis and that the y-axis begins at 0.7.

Fig. 5. Probability that multiple forwarders receive the same packet for typical wakeup intervals in ORW.

3.6. Unique Forwarder Selection

Once a packet has been received by one or more nodes in the forwarding set, the next step is to ensure that only a single node forwards it. In this section, we first show that in the majority of the cases a packet is only received by a single forwarder. Next, we introduce a lightweight coordination protocol to determine a unique forwarder in case the packet was received by multiple nodes.

3.6.1. Probability of Multiple Receivers. In ORW, a packet is forwarded by multiple nodes if: (1) multiple nodes are awake while the packet is transmitted, (2) more than one of these awake nodes successfully receives it, and (3) provides routing progress. This probability of multiple forwarders depends on two factors: the node density and the wakeup rate of each node. Both a high node density and a high wakeup rate increase the probability of a packet being received by multiple forwarders.

To have a better understanding of the possibility of having multiple forwarders we consider the following scenario. Let n be the number of forwarders that wake up exactly once uniformly in a interval $[0, T]$. Assume that once node i wakes up it will remain active for a period a (typically a few milliseconds) listening for the incoming traffic and if there is no packet destined to i, it will switch to dormant state. For simplification assume that success probabilities are equal to 1. Having a forwarder with a packet in hand, we want to calculate the probability of having at least a second forwarder that receives the same packet. This event happens when the active period of at least one forwarder (out of remaining $n-1$ forwarders) intersects with the current forwarder. The probability $q(n)$ that someone in a set of n forwarders receives the same packet as a particular forwarder (assuming fully reliable links) is then

$$q(n) = 1 - \left(1 - \frac{a}{T}\right)^{n-1}. \tag{7}$$

Figure 5 depicts the probability of a packet being received by multiple forwarders for typical wakeup periods in ORW. For example, if 16 neighbors provide routing progress, the probability of multiple forwarders concurrently overhearing the same packet is between 2% and 26% for wakeup periods from 8192 to 512ms, respectively (assuming fully reliable links). In this figure, the active period a is set to 10ms for all cases, which is the default in TinyOS (BoX-MAC). The probability of multiple forwarders decreases with increasing wakeup intervals, a key benefit as ORW targets low-power networking utilizing large wakeup intervals. Energy on the channel, such as other data transmissions or noise, may extend the duration that a node is listening and hence may increase the risk of multiple receivers.

It is worth pointing out that we leave the choice of the wakeup rate to the designer of the WSN since it has a direct relation to the application requirements. However, Figure 5 indicates that at the low wakeup rates that are targeted in this article, a packet is received by only a small number of nodes, and practically only by a single forwarder. For moderate wakeup rates where one expects multiple simultaneous forwarders, we design a lightweight coordination protocol for ORW to determine a unique forwarder out of multiple nodes, which we introduce next.

3.6.2. Coordination Algorithm. The coordination protocol in ORW fulfills two tasks: (1) it determines whether a packet was received by more than one potential forwarder, and (2) it ensures a unique forwarder in case of multiple receivers. Our goal is to achieve a duplicate rate similar to traditional unicast routing schemes. Due to the already low probability of multiple forwarders for a single packet at low wakeup intervals and traffic rates targeted by ORW (see Section 3.6.1), we aim for a practical, lightweight design. It relies on three mechanisms.

Demanding a Single Acknowledgment. Potentially, a sender receives multiple acknowledgments, one from each receiver[2]. This indicates multiple forwarders and hence the sending node resolves this as follows: It retransmits the packet and the forwarders will (potentially) receive the packet again. Receiving a link-layer duplicate, forwarders send a second acknowledgement only with 50% probability to reduce the number of duplicate acknowledgments. Eventually, only one of the nodes sends an acknowledgement. Upon receiving such a single acknowledgment, the sender determines that a unique forwarder was selected and does not initiate further retransmissions. Once a forwarding node does not receive further retransmissions of a packet it just acknowledged, it concludes that it is the sole forwarder for the packet. The same mechanism is applied if acknowledgments collide, that is, no acknowledgement is received (after a timeout) by the source. The calibration of acknowledgment probability in presence of link-layer duplicates to 50% is chosen based on the following observation: Usually, a packet is received by only one forwarder. However, if we have multiple forwarders, we expect two and not more forwarders (see Figure 5(b)). Thus, a 50% probability to acknowledgment helps to quickly select one forwarder. Nonetheless, in practice link dynamics and asymmetry can lead to the sender only receiving one of the acknowledgments sent by the multiple forwarders. In ORW, we address this with the following two mechanisms.

Data Transmission Overhearing. When one node overhears another node forwarding the same packet while waiting for a clear channel, it cancels its own transmission. This mechanism is common in opportunistic routing schemes: For example, ExOR [Biswas and Morris 2005] uses overhearing as a key mechanism to ensure a single forwarder. In ORW, we add it as an additional mechanism to further detect multiple forwarders and reduce the number of duplicates. Note that adding this mechanism comes at practically zero cost, as nodes waiting for a clear channel are in receive mode anyway.

Network-Layer Duplicate Detection. Finally, network-layer duplicate detection serves as fallback in case a packet slipped through the other mechanisms. Each node keeps

[2]The delay between a frame and its acknowledgement is bounded by the time for the receiver to take the forwarding decision, and an additional small random time we inject, to guarantee nonconstant timing and make acknowledgement collision more unlikely.

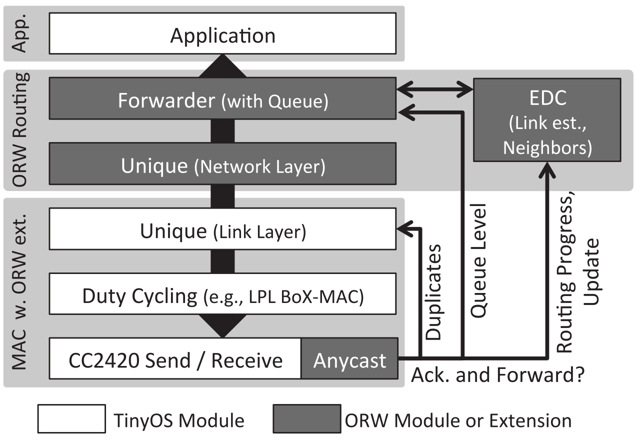


Fig. 6. Cross-layer control flow in ORW: Before acknowledging and forwarding a packet, ORW checks whether: (1) the node provides the requested routing progress, (2) has space in the queue, and (3) the packet is not a duplicate.

track of the sequence numbers and source addresses of recently forwarded packets and filters any duplicates. Assuming a sufficiently large history, this mechanism detects all duplicates. However, as packets in ORW potentially travel along different routes, duplicate detection is delayed until the forwarding path of two duplicates merges.

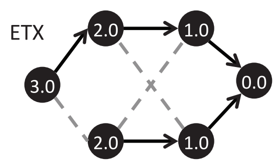
Finally, in the dynamic environment of low-power wireless with asymmetric or unstable links, our practical design cannot theoretically guarantee a unique forwarder and packets may slip through. If they take different routes, this duplicate might only be detected at the sink. However, our evaluation shows that the lightweight mechanisms of ORW are sufficient to keep the duplicate rate at a level similar to traditional unicast routing such as CTP [Gnawali et al. 2009].

3.7. System Integration

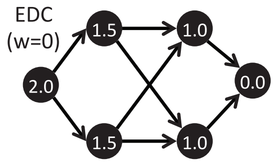
ORW acts as replacement of the unicast forwarding logic of WSN collection protocols. As a case study we integrated ORW into CTP, the de facto standard for collection in TinyOS. Moreover, we designed ORPL [Duquennoy et al. 2013] that applies the ideas of ORW to RPL [Winter et al. 2012].

In this section we the discuss system integration of ORW and the portability of our design choices. ORW provides the same interfaces as CTP to the application, uses its protocol headers, and reuses—in part—its TinyOS modules such as the forwarder module. Anycast routing in ORW relies on two headers: the EDC of a node and the required routing progress is stored as two 8-bit values in the 802.15.4 MAC header instead of the 16-bit destination address, which is not required for anycast routing. Thus, we allow EDC values from 0.0 to 25.5 at a granularity of 0.1. Overall, the integration into the 802.15.4 header allows a node to decide whether it provides the required routing progress after reading merely the header. Hence, it reduces energy consumption and ensures that 802.15.4 acknowledgments are triggered in a timely manner.

Additionally, we extend the CTP routing header with one field: we add a weighted average of the transmission rate to facilitate link estimation (see Section 3.5). ORW places a small interface between routing and MAC layer (see Figure 6): To decide whether to accept, that is, acknowledge and forward, a packet, it determines whether: (1) the node provides routing progress, (2) has space in its queue, and (3) the packet is not a duplicate (see Section 3.6.1). Hence, although ORW places functionality on



(a) Tree: Building a routing tree with ETX as metric



(b) DODAG: Building a DODAG with EDC as metric

Fig. 7. Topology I: We depict the resulting tree and DODAG when using ETX and EDC, respectively, to build the routing topology. The toy topology highlights that a DODAG built with EDC leads to a dense routing topology when compared to the tree built with ETX. For simplicity, all links are assumed to be reliable, that is, have a PRR of 1. The values depicted on the nodes represent the routing estimates computed by the respective metrics.

both the routing layer and the MAC layer, its design is not bound to a specific routing protocol, MAC layer, or duty-cycling scheme. ORW including link estimation requires slightly less RAM and ROM than CTP and its link estimator 4BitLE [Fonseca et al. 2007] which is mainly due to our simplified link estimator.

3.8. Examples: Building Topologies for Opportunistic Routing with EDC

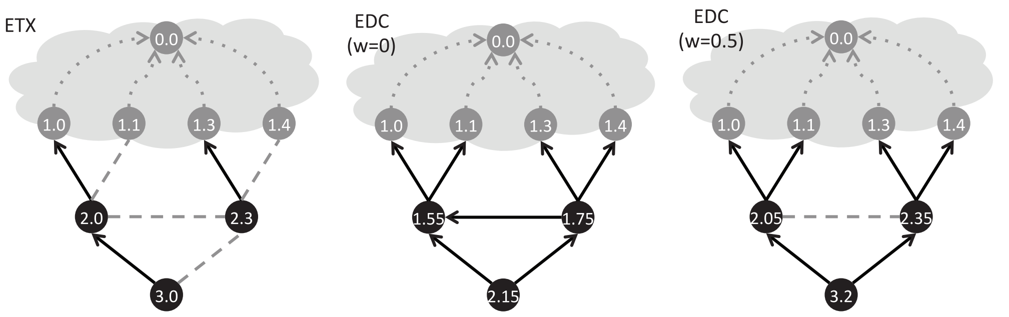
In this section we discuss how DODAGs built with EDC as the routing metric differs from trees formed with the ETX metric (see Section 3.3 and Eq. (6)). We show two sample topologies to highlight the following key differences between these two metrics. (1) By using EDC as the routing metric, nodes tend to utilize more links when compared to the ETX metric. (2) EDC, unlike ETX, is able to establish a trade-off between the number of hop counts and the robustness against link dynamics.

In principle, the ETX metric represents the expected number of (re)transmissions until a packet reaches its destination. In contrast, EDC represents the estimated end-to-end delay (in the unit of duty-cycled wakeups). Hence, while the resulting topologies from these metrics are comparable, for example, in terms of links utilized for routing, the values computed by the respective metrics are not directly comparable.

The routing topologies for ETX and EDC are depicted in Figures 7(a) and 7(b), respectively. They form a simple toy topology of 6 nodes. The figure highlights that a DODAG built with EDC utilizes more links than a tree constructed with ETX, leading to a dense routing topology. This translates into more stability against link dynamics and node failures. Additionally, it reduces the average end-to-end delay in duty-cycled networks by exploiting the spatial diversity of the different routes.

Figure 8 illustrates a subset of a larger network. It shows how EDC-based topologies can tune the number of active links when compared to ETX-based topologies. For example, if we set $w = 0$, all links in the sample topology are utilized (see Figure 8(b)) while a routing tree with ETX utilizes less than half of the links (see Figure 8(a)). As a consequence, the DODAG of EDC ensures a high stability to the dynamics of low-power wireless links and nodes. Additionally, the increased number of forwarding choices reduces the average end-to-end delay.

Using EDC as routing metric results in a large number of forwarding choices. Thus, it is not optimized in terms of hop counts and (re)transmissions when compared to the ETX metric. As a result, some packets in the DODAG may reach their destination via more hops than in the ETX tree. On the other hand, for values of w larger than 0 the degree of connectivity of the DODAG reduces (see Figure 8(c)) at the cost of an increased average end-to-end delay. Overall, theses examples show that the EDC metric is able to maintain a nice balance between the stability to wireless link dynamics and the number of hops required to reach a destination. Please note, for values of w larger than 0, the metric depicted in the figures shows the sum of the estimated end-to-end delay



(a) Tree: ETX as routing metric (b) DODAG: EDC with $w = 0$ as routing metric (c) DODAG: EDC with $w = 0.5$ as routing metric

Fig. 8. Topology II: For a given set of parent nodes (depicted in gray) offering a different routing progress each to the destination, we depict the resulting routing topology for ETX, EDC with $w = 0$, and EDC with $w = 0.5$. The figures highlight the impact of the forwarding cost w on the number of active links. For the values of w larger than zero, EDC is able to tune the number of utilized links to avoid the packets from veering across the network. For simplicity, all links are assumed to be reliable, that is, have a PRR of 1. The values depicted on the nodes represent the routing estimates computed by the respective metrics.

Table II. The Summary of Our Evaluations

Evaluation result	Type	Section
The analytical model is valid	Simulation (Monte-Carlo)	4.1.2
EDC matches the analytical model	Simulation (Traces)	4.1.3
EDC outperforms other metrics in a duty-cycled WSN	Simulation (Traces)	4.1.4
EDC strongly benefits from network density	Simulation (Random Top.)	4.1.5
Calibration of forwarding cost w	Testbed implementation	4.2.2
ORW strongly improves duty cycle and delay	Testbed implementation	4.2.3
ORW shows strong resilience node failures	Testbed implementation	4.2.4
ORW stabilizes quickly without expensive probing	Testbed implementation	4.2.5
ORW allows much lower wakeup intervals	Testbed implementation	4.2.6

and w and is not a direct representative of the delay. Therefore, it cannot be directly compared with ETX values.

4. EVALUATION

In this section we evaluate ORW in both simulations and testbed deployments (see Table II). First, we validate our analytical model via extensive Monte-Carlo simulations on an instance setup. Second, we focus on the EDC metric and evaluate its accuracy by comparing the resulting routing topology to an optimal anypath routing topology constructed by exhaustive search based on the analytical model. Third, we employ simulation to study the EDC metric and its differences to ETX and two alternative anypath routing metrics, namely EAX [Zhong and Nelakuditi 2007] and FRA [Dubois-Ferrié 2006].

Finally, we move to deployments and utilize two large testbeds for a detailed experimental comparison of ORW and the state-of-the-art unicast routing protocol CTP [Gnawali et al. 2009]. We focus on four key metrics: radio duty cycle, end-to-end delay, reliability, and transmission counts. Additionally, convergence time of the routing algorithm, impact of node failures, the choice of wakeup intervals, and spatial diversity of ORW are investigated. We discuss the advantages and limitations of our method compared to the state-of-the-art. Table II presents a summary of our evaluation results as well as pointers to the appropriate sections.

4.1. Simulation: Anycast EDC

In our simulation-based evaluation, we explore the potential of anycast routing with EDC in terms of delay and hops when compared to alternatives in unicast (ETX) and anycast routing (EAX, FRA) [Zhong and Nelakuditi 2007; Dubois-Ferrié 2006]. Also, we explore the impact of network density on the performance of EDC-based routing and we evaluate the influence of the transmission cost w on EDC.

4.1.1. Simulation Setup. In the simulations, we solely focus on the different routing metrics and not individual protocol implementations. Thus, we compare two idealized protocol families in this section: unicast routing with ETX and anycast routing with EDC, EAX, and FRA. We use three sets of simulation data: (1) extensive Monte-Carlo simulations (in Section 4.1.2), (2) PRR traces from testbeds (in Sections 4.1.3 and 4.1.4), and (3) PRR traces of randomly generated topologies (in Section 4.1.5).

In Section 4.1.2, we begin our simulation with extensive Monte-Carlo simulations and show that they match the analytical model presented in Section 3.2. Next, we compare the EDC metric to the state-of-the-art ETX, EAX, and FRA metrics (see Sections 4.1.3 and 4.1.4). To ensure a fair and realistic evaluation, we base our simulations on a set of network profiles derived from PRR traces of two testbeds: Twist [Handziski et al. 2006] and Motelab [Allen et al. 2005] with 96 and 123 nodes, respectively[3]. The Tx power for both testbeds is 0dBm. Moreover, we picked the nodes with ids 229 and 1 as sink nodes in Twist and Motelab, respectively. In addition, we use randomly generated topologies ranging from 100 to 1000 nodes to explore the impact of network density on the performance of EDC as a routing metric (see Section 4.1.5). Nodes are placed in a fixed area and we employ the Friis transmission model to determine link quality and PRR between nodes. The results presented for this simulation are averaged over 100 random topologies per data point.

On top of these topologies, we employ the routing metrics for numerical simulation in Matlab and Python. Hence, to ensure a fair comparison focusing only on the metrics we deliberately exclude (in this numerical simulation) protocol mechanisms outside of the routing metric itself such as link estimation or neighbor discovery. Overall, our goal is to evaluate the underlying performance of our routing metric EDC independent of a specific protocol implementation, before we compare EDC-based anycast routing in ORW to CTP in our testbed-based evaluation (see Section 4.2).

4.1.2. Validating the Analytical Model. We first compare the analytical model versus extensive Monte-Carlo simulations in a small example. The goal of this section is to show the match between the analytical model presented in Section 3.2 and a fine-grained numerical simulation that calculates the average of the number of duty-cycled wakeups for end-to-end delivery. We consider a toy example with a sender node that has 1 to 10 forwarders with fixed average wakeups and link reliabilities. We start the experiment with a sender and the first forwarder and calculate (both analytically and numerically via simulations) the average number of wakeups for the sender to deliver a packet to a hypothetical sink given the fixed average wakeups and link reliability of the forwarder. Then, we add the second forwarder while keeping the first one and recalculate the average wakeups for the sender. We repeat the same procedure until all 10 forwarders are added to the forwarding set.

The results of the experiment are depicted in Figure 9. Each data point in the curves corresponds to the updated y-axis values after adding a new forwarder. The y-value of each square in the plot is the average of one hundred thousand numerical simulation instances. The first observation is the accurate match between theoretical model and

[3]The traces for Motelab are available as part of the SING dataset (http://sing.stanford.edu/srikank/datasets.html).

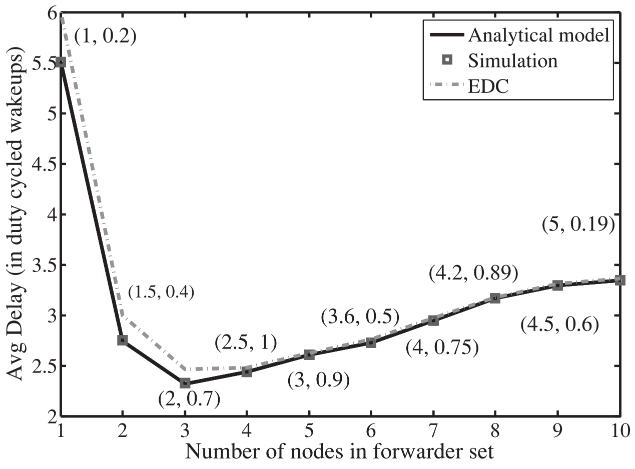


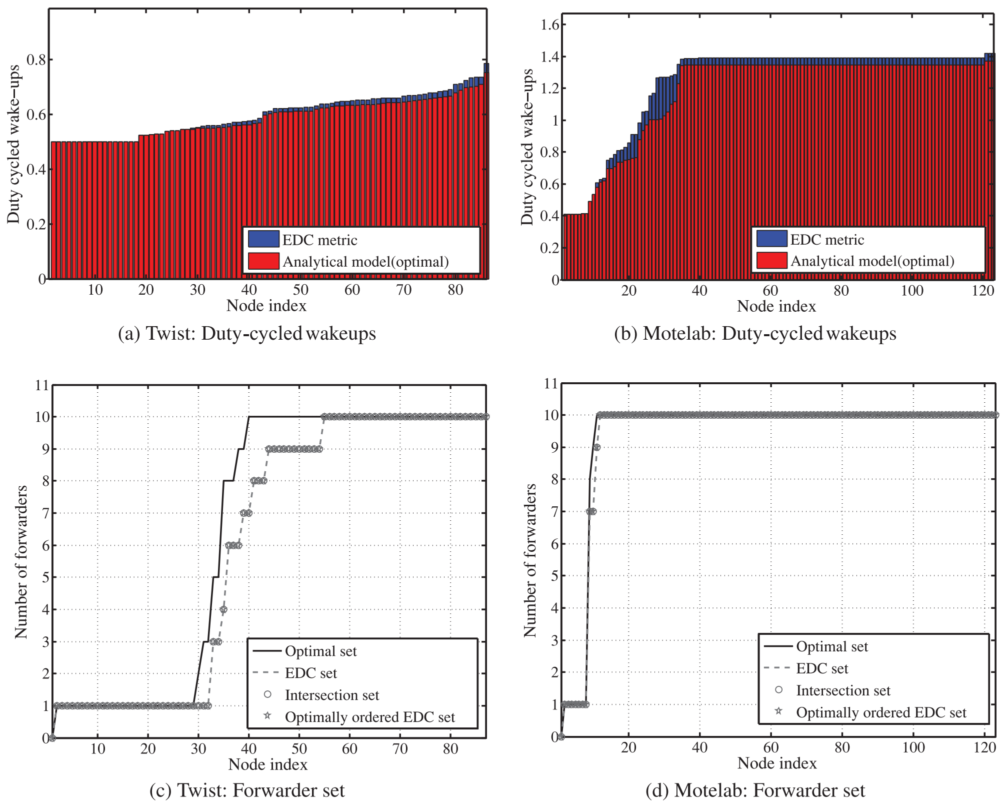
Fig. 9. Comparison of analytical model versus simulation and EDC metric. Forwarder set grows incrementally with new members shown by tuple (EDC, reliability).

the simulation of the $\mathbf{E}\{\mathrm{DC}\}$. Leveraging on this match, we will continue our evaluations while keeping the analytical model as a baseline. Second, we also included EDC metric values for the comparison. Note that with a growing number of forwarders, EDC values eventually tend to the theoretical values. We intentionally have sorted forwarders in increasing order of average wakeups. As a result, the plot represents the behavior of the forwarder selection algorithm. Starting from the forwarder with the minimum EDC, the sender inserts a new member into the forwarding set if the EDC of the new forwarder is less than the current value of the y-axis, which in this example happens until the third forwarder.

4.1.3. EDC versus Analytical Model. To evaluate the EDC metric versus the analytical model, we perform several simulations over Twist and Motelab data traces. Specifically, we run the forwarder set construction mechanism (Algorithm 1) that utilizes the EDC metric as its core and compare it against an exhaustive search based on our analytical model. The number of forwarders for each node is limited to up to 10 nodes. This restriction is made to ensure practical computability due to the exponential complexity of the analytical model as well as the exhaustive search over the forwarder candidates.

Figure 10(a) and Figure 10(b) show optimal duty cycles ($\mathbf{E}\{DC\}$) versus the EDC metric for each node in the network. We note that EDC values are very close to the analytical values. Figure 10(c) and Figure 10(d) illustrate the number of forwarders versus node index. For each node, the number of optimal forwarders (given by exhaustive search) and the numbers from the EDC metric are depicted. Moreover, the number of common forwarders in these two schemes as well as the number of truly ordered common forwarders are plotted. We observe that the last three curves coincide with each other. In other words, the EDC metric for each node picks a subset of forwarders with the same order as the optimal set. Roughly speaking, for each node, the best forwarder in the optimal set (the one with lowest $\mathbf{E}\{DC\}$) is also the first forwarder that EDC metric picks, and so on.

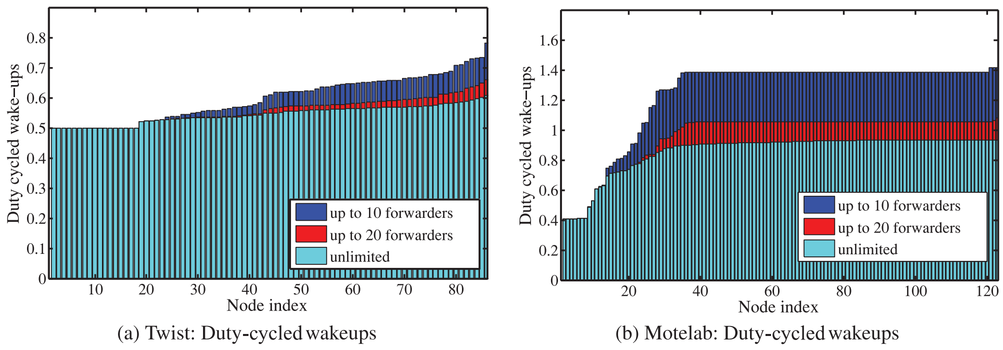
The EDC values of nodes based on different restrictions on the number of forwarders are illustrated in Figure 11. For some nodes (the ones closer to the sink) the EDC value does not change. The reason is that, due to good paths towards the sink, they do not add more forwarders. In contrast, the nodes farther from the sink (the ones with higher



(a) Twist: Duty-cycled wakeups (b) Motelab: Duty-cycled wakeups

(c) Twist: Forwarder set (d) Motelab: Forwarder set

Fig. 10. Per-node comparison of EDC values and forwarder set in Motelab and Twist profile. The number of forwarders is upper bounded by 10. The plots show high accuracy of EDC metric in both duty cycles and forwarder selection compared with optimal solution derived by exhaustive search.



(a) Twist: Duty-cycled wakeups (b) Motelab: Duty-cycled wakeups

Fig. 11. Per-node comparison of EDC values in Motelab and Twist profile. Each color represents a different upper bound on the number of forwarders.

EDC values) benefit from having more forwarders. We observe that the values for these profiles do not change significantly after 20 forwarders.

4.1.4. EDC versus Other Metrics. After evaluating the accuracy of EDC and the impact of the size of the routing table and the forwarder set, we now compare EDC with other metrics. We acknowledge that these metrics were devised in different design contexts (e.g., general-purpose wireless networks, networks without duty cycling or energy restrictions, etc.) and that a thorough comparison would require a full system-level implementation. However, this is not the aim of this section. Here, we conduct the comparison in the context of duty cycling, that is, all the nodes are battery powered and only activate for a short interval followed by a random wakeup. So the underlying assumption is to utilize ORW as a design core and then compare the performance of different routing metrics in the duty-cycled environment. The comparison is done in terms of three key quantities: the average number of wakeups required for end-to-end packet delivery (as an indicator of the delay); the average number of forwarders in the routing tables (as an indicator of the overhead of route computation); and the average number of hops required by a packet to reach the (single) destination.

For the baseline of the comparison, we opted for the expected number of duty-cycled wakeups, $\mathbf{E}\{\mathrm{DC}\}$, that has been developed in Section 3.1 and verified via Monte-Carlo-based simulations in Section 4.1.2. The first counterpart is the widespread ETX [De Couto et al. 2003]. Note that routing protocols using ETX aim to minimize the transmission count. In contrast to EDC, ETX does not take duty cycling into account. Hence, reducing transmission counts in ETX does not necessarily lead to low delays, nor does it reduce radio-on time. The second and third counterparts of EDC are selected from the literature of anycast routing in wireless networks.

To the best of our knowledge there is no metric next to EDC that considers the joint effects of both duty cycling and link failure probability in the routing-cost model. However, some metrics [Kim et al. 2008; Dubois-Ferrié 2006; Dubois-Ferrié et al. 2011; Zhong and Nelakuditi 2007] include either link reliability or duty cycling into their model. The popular EAX [Zhong and Nelakuditi 2007] metric accounts for link failure probabilities in its opportunistic routing while Dubois-Ferrié [2006] provides an opportunistic metric for the duty-cycled WSNs assuming fully reliable links. We note that the aforementioned metrics are not directly comparable with EDC in their original design format. The EAX metric assumes all the potential receivers are active while the sender transmits and it requires to maintain a priority list that dictates which parallel receiver has to relay the received packet. However, in a duty-cycled environment the assumption of EAX that forwarders are available concurrently does not hold. On the other hand, Dubois-Ferrié [2006] employs preambles in a duty-cycled link layer while ORW replaces preambles with the actual packet (re)transmissions to gain a higher performance [Buettner et al. 2006]. Hence, the original equations for the cost calculation in Dubois-Ferrié [2006] include preamble details and are not directly applicable for ORW implementations. Nevertheless, we believe that it is interesting to investigate whether the performance benefits of the EAX metric prevail compared to ETX once it is applied in the duty-cycled ORW, and whether it is important to consider both link reliabilities and wakeups in the EDC metric.

For this reason, based on the previous metrics, we have tailored two opportunistic alternatives for EDC. EAX* has the same metric values of EAX [Zhong and Nelakuditi 2007] but operates under the realm of ORW. The anycast routing with full reliability assumption (FRA) metric is the same as EDC with the assumption that all links are fully reliable. However, while weak links are either beneficiary or transparent for ETX, EDC, and EAX*, they are rather unfair for the FRA metric, especially because FRA assumes perfect link reliabilities. For this reason, only for FRA do we employ a minimum threshold of 50% reliability for links to be included in FRA pools.

Figure 12 illustrates the comparison between EDC, ETX, EAX*, and FRA for two reference testbeds. Several comments are in order. First, the plots show that EDC reduces the delivery delay when compared to alternatives. In both testbeds, FRA and ETX have the worst average delay. EDC with 20 maximum allowed forwarders achieves a better delay compared to EAX* in both testbeds while the average number of forwarders for

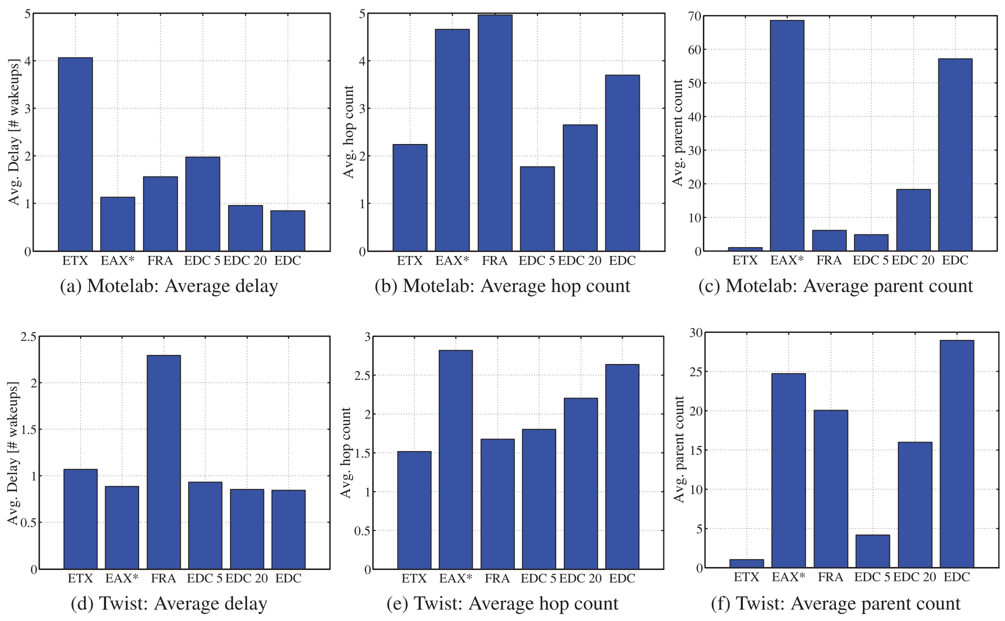
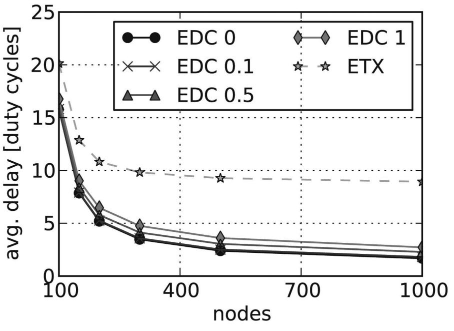


Fig. 12. Comparing EDC versus ETX, EAX*, and anycast routing with full reliability assumption (FRA) on Motelab and Twist traces. The EDC 5 and the EDC 20 data bars refer to the EDC metric when its number of forwarders is limited to 5 and 20, respectively. The EDC metric outperforms the three metrics in terms of delay. Additionally, EDC can be tuned so that it keeps the same average delay while reducing the hop count and the number of forwarders.

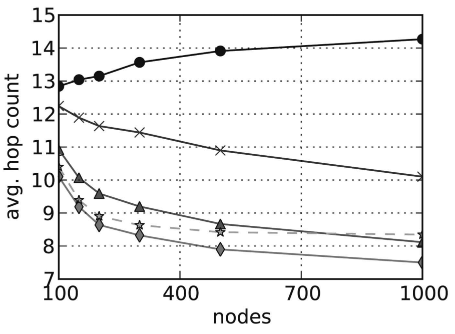
EAX* in Motelab and Twist are about 3.5 and 1.5 times more than EDC 20, respectively. Second, we show that, depending on the size of the forwarding table, EDC achieves hop counts similar to ETX (see Figure 12(b) and Figure 12(e)). In some situations it even outperforms ETX. Third, compared to ETX, EDC-based routing explicitly utilizes all neighbors: instead of waiting for one specific neighbor to wake up as in ETX-based routing, EDC utilizes the first neighbor that wakes up and provides routing progress. As a result, EDC outperforms ETX in terms of delay while leading to more hops. Fourth, ignoring either link reliability or radio-on time in the metric design of the duty-cycled WSNs causes a performance degradation. On the other hand, relying merely on the hop count as an indicator of delay is not efficient since a sender may waste both time and energy by waiting for a best neighbor to wake up. Overall, with EDC we are able to limit the aggressive forwarding and force nodes to select a small number of good parents (see Figure 12(c) and Figure 12(f)). In this way we trade delay for hop count and maintenance time of the forwarder set: by letting more candidates in the forwarder set we experience less delay while the routing algorithm requires more steps to stabilize. As a final remark, note that, even though EDC requires a larger forwarder set than ETX and FRA (but often less than EAX*), the overhead of routing table management remains unchanged. In practice, most metrics need to keep a similar number of neighbors as EDC in their neighbor table to keep track of the best routing option.

4.1.5. EDC: Network Density and Forward Cost. After comparing EDC to the state-of-the-art, we detail on EDC itself in this section. We evaluate how EDC benefits from network density when compared, for example, to traditional unicast routing metrics ETX. Also, we discuss the impact of the per-hop forwarding cost w on the routing performance.

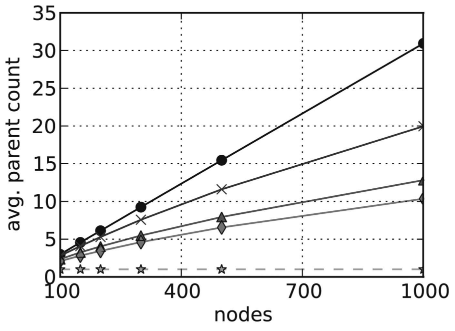
First, we investigate the impact of network density on the performance of the routing mechanisms. Figure 13 shows that anycast routing with EDC benefits from an



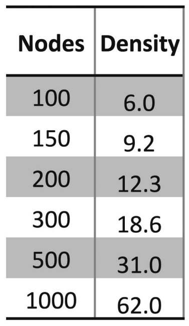
(a) Delay: EDC outperforms ETX by a factor of 1.3 to 6, depending on network size and choice of w. For w of 0, EDC shows the lowest delay



(b) Hops: For w of 1, EDC outperforms ETX in hops. For other configurations it trades lower delay for higher hop counts



(c) Parents: Increasing network size and density allows EDC to utilize more parents for forwarding, while increasing w reduces their number



Nodes	Density
100	6.0
150	9.2
200	12.3
300	18.6
500	31.0
1000	62.0

(d) Density: average node densities

Fig. 13. Simulation-based evaluation: The results show that anycast routing with EDC benefits from increased density much more than unicast routing with ETX. Nodes are placed in an area of fixed size, leading to increased network density when the number of nodes increases.

increased network density much more than unicast-based ETX. Compared to ETX, EDC reduces the average delay for packet delivery by factors of 1.3 and 6 for network sizes of 100 and 1000 nodes, respectively (see Figure 13(a)). We later show that this leads to energy savings on a similar scale (see Section 4.2). As before, EDC leads to hop counts larger than ETX (see Figure 13(b)).

Finally, we evaluate how the per-hop forwarding cost w impacts the hop count in the routing topology. Figure 13(c) and Figure 13(d) show that with $w = 0$, EDC utilizes about half of the neighboring nodes as parents. This decreases to 15% of the neighbors for $w = 1$. EDC with $w = 0$ utilizes all forwarders that provide even the smallest routing progress. Hence, while minimizing delay, an increased network density increases both the hop count and the number of parents of a node. Overall, these results show that a low value of w leads to the lowest routing delay, which directly translates to low radio duty cycles and energy consumption, as we show later. This is independent of the deployment and its characteristics such as traffic rates and wakeup intervals, as our simulations results underline. However, in practice such aggressive forwarding also makes this configuration of EDC sensitive to link dynamics and potentially leads temporary to routing loops, as we show in Section 4.2. In our experimental evaluation we next show that a configuration of w slightly above 0, such as 0.1, avoids these problems while utilizing many parents and providing low delay and high energy efficiency.

Table III. We Use Five Evaluation Scenarios

Testbed	Size	Sink	Tx Power	Diameter
	nodes, m^3	id	dBm	hops
Indriya	120,	1	0	5.6
	$50 \times 25 \times 20$	1	-10	9.1
Twist	96,	229	0	3.4
	$30 \times 13 \times 17$	229	-25	7.1

Each testbed contains about 100 nodes and the diameter ranges from 3 to 9 hops.

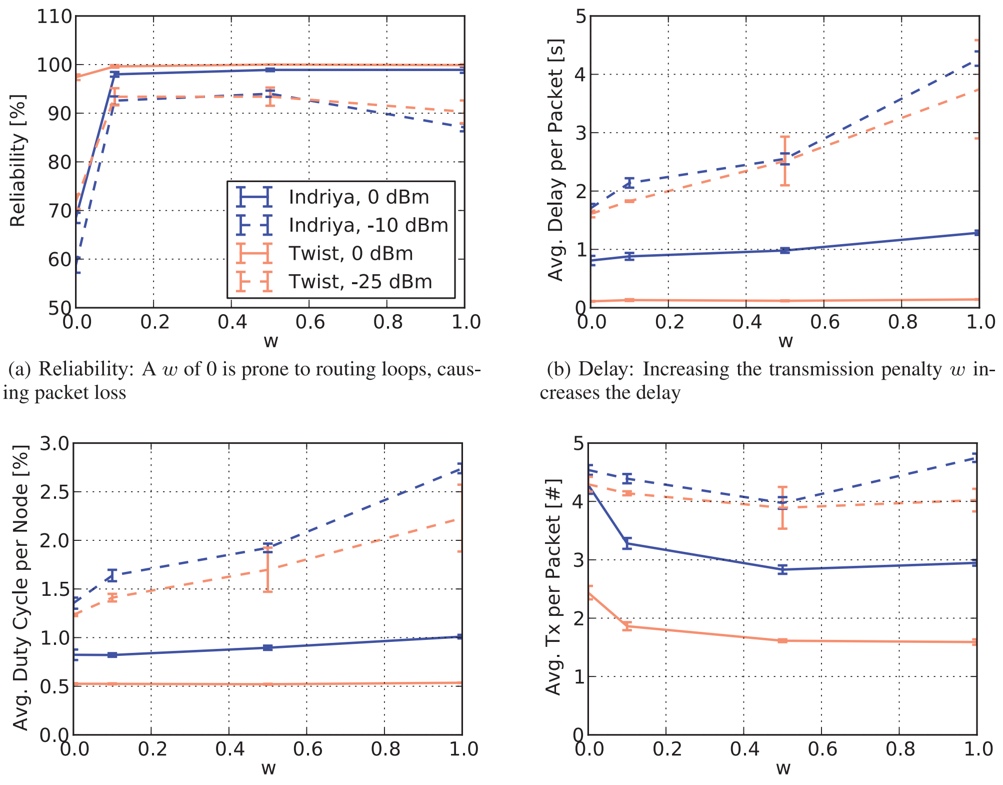
4.2. Testbed-Based Evaluation of ORW

In this section we evaluate ORW on real-world testbeds and compare its performance to CTP, the de facto standard collection protocol in TinyOS. Please note that the concepts in ORW are generic and independent of the chosen duty-cycling scheme: They apply to both asynchronous and synchronous (phase-locked) MAC schemes as well as receiver-initiated ones (see Section 2.3). However, to ensure a fair comparison with CTP, we use BoX-MAC in this evaluation. It is the default MAC in both TinyOS and CTP, and resembles a combination of X-MAC and B-MAC [Polastre et al. 2004]. We also show results for CTP with A-MAC [Dutta et al. 2010], a state-of-the art, receiver-initiated MAC. We do not compare to SCP-MAC [Ye et al. 2006], a synchronous MAC. Although it promises high energy efficiency, it is not available for current TinyOS releases, and recent work [Vanhie-Van Gerwen et al. 2010] indicates that its energy efficiency in multihop collection trees is well below BoX-MAC.

4.2.1. Testbeds and Metrics. We base our experimental evaluation on two testbeds: Indriya [Doddavenkatappa et al. 2011] and Twist [Handziski et al. 2006], with 120 and 96 nodes, respectively. For each testbed we use two levels of transmission power, resulting in four evaluation scenarios (see Table III). We use the following setup for both ORW and CTP. Every node generates a packet randomly with an average interval of 4 minutes, and the network forwards it to the sink. Unless explicitly mentioned, we use a wakeup interval of 2 seconds (with our settings, this leads to the optimal duty cycle in CTP for the given traffic load; see Section 4.2.6). As sink nodes we use corner nodes 1 for Indriya and 229 for Twist, respectively. We adopt the CTP default setting of having the sink node always on. For all experiments, we use channel 26 of 802.15.4 to avoid 802.11 impacting our results and potentially leading to an unfair comparison.

We evaluate energy consumption through the average duty cycle in the network, that is, the portion of time spent with the radio chip turned on. The average duty cycle is a good proxy for energy consumption because: (1) the radio chip consumes far more power than the other hardware components involved in our experiments and (2) low-power radio chips in sensor motes have a comparable power draw when transmitting or listening. This metric provides us with results that are independent from environmental conditions and hold across different hardware platforms, and are therefore reproducible. For a fair comparison we also skip the first two minutes when measuring duty cycles, as CTP shows a high duty cycle in this time due to its initial link probing.

For each data point, experiments are executed for a minimum of 30 minutes and are repeated three times; experiments are executed at random times of the day, but back-to-back to ensure fairness. We display average results and error bars show standard deviations. Overall, the results shown are based on more than 300 individual experiments, each between 30 minutes and 2 hours.



(a) Reliability: A w of 0 is prone to routing loops, causing packet loss

(b) Delay: Increasing the transmission penalty w increases the delay

(c) Duty Cycle: Increasing w increases the duty cycle

(d) Tx: The transmission count decreases when increasing w

Fig. 14. System calibration: A forwarding cost w of 0.1 is a good balance between energy efficiency, delay, and reliability.

4.2.2. System Calibration: w. We start the testbed-based evaluation by calibrating the forwarding cost w. Figure 14 shows that in all four scenarios, a low w leads to the best performance in terms of delay and duty cycle (as also indicated by our simulation results). A larger w reduces hop counts at the price of increased delay and duty cycle. However, reliability shows a sharp drop for w of 0. Our logs indicate that a w of 0 increases the risk of routing loops and duplicate packets, which increase packet drops and reduce reliability. Please note that integrating ORW into the address fields of 802.15.4 limits us to evaluate w at a granularity of 0.1, that is, restricts us to choose a w of 0, 0.1, 0.2, etc. (see Section 3.7). Overall, all of our four evaluation scenarios show that a value of 0.1 for w provides a stable balance between reliability, delay, and duty cycle. As these scenarios cover a wide range of network densities, network diameters, and forwarding loads per node (see Table III), we believe that a w of 0.1 is a good choice in general and use it as default in ORW. Later we further detail on this aspect in Section 4.2.6.

4.2.3. Per-Node Comparison of ORW to CTP. Next, we compare the performance of ORW and CTP in our four evaluation scenarios (see Table III). Figure 15 and Table IV show that ORW significantly improves duty cycles and delay. On average, ORW roughly doubles the energy efficiency; individual nodes show improvements up to 88%. The results show that ORW strongly benefits from network density: it shows the best results on the dense Twist deployment at a transmission power 0dBm. Additionally, it improves

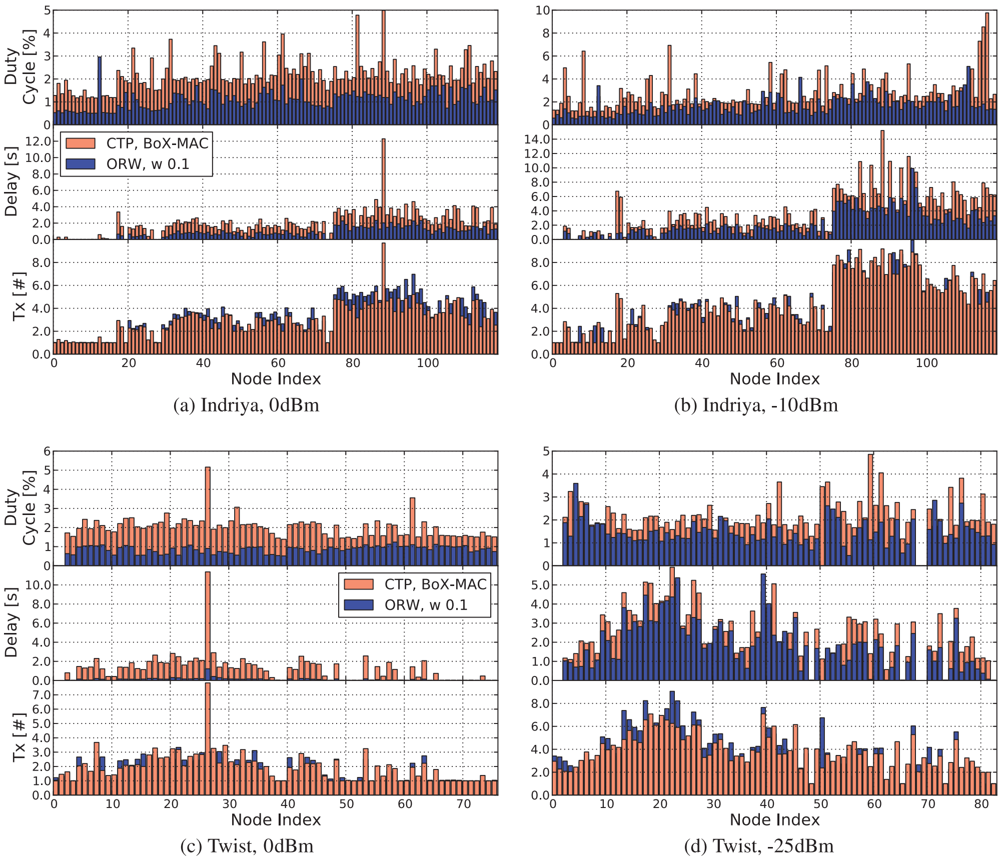


Fig. 15. Per-node comparison of ORW and CTP: ORW improves duty cycles and delays while achieving slightly higher hop counts than CTP.

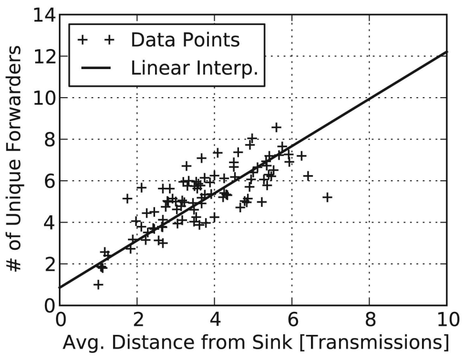
Table IV. Testbed Experiments Summary

Testbed	Duty Cycle					Delay			Tx		
	Trace [%]		Improve [%]			Trace [s]		Impr. [%]	Trace [#]		Impr. [%]
	ORW	CTP	Avg.	Max.	Min.	ORW	CTP	Avg.	ORW	CTP	Avg.
Indriya, 0 dBm	1.1	2.2	50	79	−19	0.8	2.0	58	3.3	3.0	−11
Indriya, −10 dBm	1.6	2.8	41	88	−30	2.1	3.8	44	4.4	4.5	0
Twist, 0dBm	0.8	2.0	57	82	0	0.1	1.2	91	1.8	2.0	10
Twist, −25 dBm	1.4	2.1	33	76	−39	1.8	2.4	29	4.1	3.8	−10

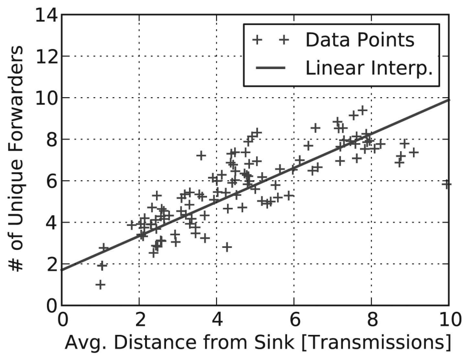
ORW decreases average duty cycles up to 57% and delays up to 90%, while achieving similar but slightly higher transmission count than CTP.

delay by 30% to 90% depending on network density and achieves (re)transmission counts (unicast or anycast) that are similar, but slightly higher compared to CTP.

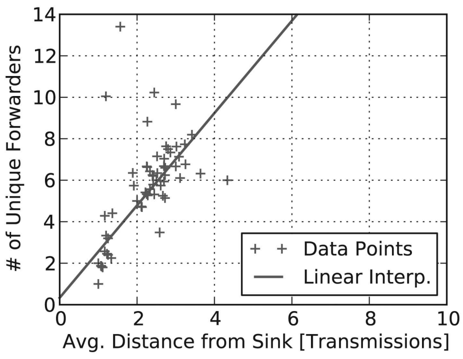
In ORW, nodes in dense networks and the ones further away from the sink benefit the most from spatial diversity in anycast forwarding (see Figure 16). We define spatial diversity as the number of different (i.e., unique) forwarders that are utilized per hop on the path from a node to the sink during the course of the experiment. As another benefit of anycast forwarding, ORW removes outliers both in terms of duty cycles and delay (see Figure 15). This has two key advantages: (1) It reduces the time until the



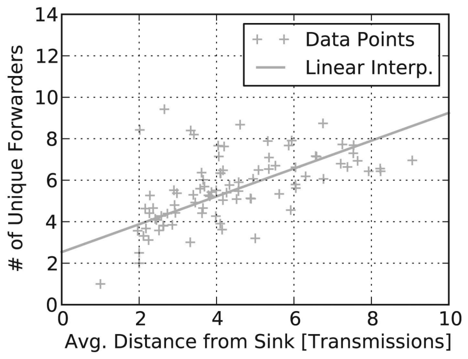
(a) Indriya, 0dBm



(b) Indriya, -10dBm



(c) Twist, 0dBm



(d) Twist, -25dBm

Fig. 16. Spatial diversity (number of unique forwarders): Nodes in dense networks and the ones further away from the sink exploit spatial diversity, that is, are able to utilize different forwarders, in anycast routing. The packets originating from these nodes use the largest number of different forwarders. We plot data points and their linear regression.

first node runs out of energy and hence ensures that sensor coverage can be maintained longer. (2) In terms of delay it allows us to switch to lower duty cycles in delay-sensitive applications, in turn reducing energy consumption even further.

4.2.4. Impact of Node Failures on ORW and CTP. We evaluate the impact of node failures on both ORW and CTP. Each 15 minutes, we remove on average 10 nodes from the network. Throughout the course of two hours this reduces the number of nodes from 120 to about 30 on the Indriya testbed.

Figure 17 depicts the impact of node failures on the key metrics of reliability, transmissions, duty cycle, and delay. Both protocols show spikes of reduced reliability in presence of node failures. For increased node failure rates, these spikes grow strongly for CTP; hence ORW maintains connectivity much longer in the resulting sparse network. Additionally, it shows the benefits of anycast routing in ORW over unicast routing in CTP: node failures have only minimal impact on the duty cycle, delay, or transmission counts of ORW, while these show sharp peaks in CTP.

4.2.5. Convergence of ORW. Earlier in Section 3.4.1 we demonstrated that the routing topology in ORW converges in finite time, assuming that link qualities did not change in this process. As ORW essentially operates without probing for link estimates (see Section 3.5), we evaluate its convergence in this section. We track the evolution of the average EDC as well as number of neighbors and parents per node. Figure 18

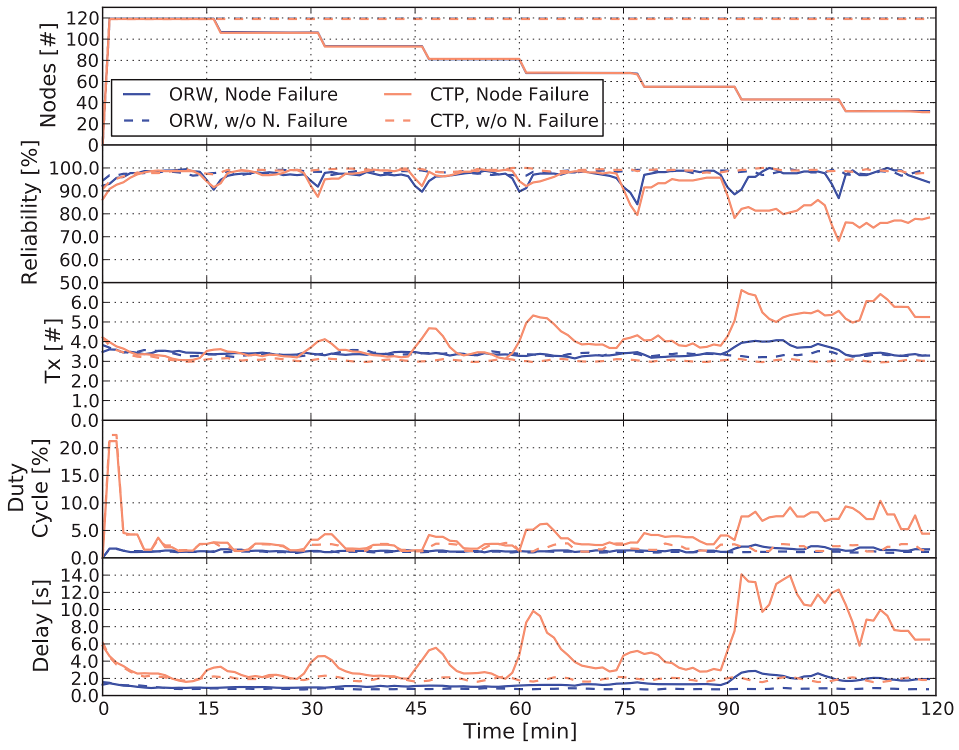


Fig. 17. Node failures (Indriya, 0dBm): While both ORW and CTP achieve similar reliability in presence of node failures, CTP pays a higher price in terms of energy, delay, and transmissions.

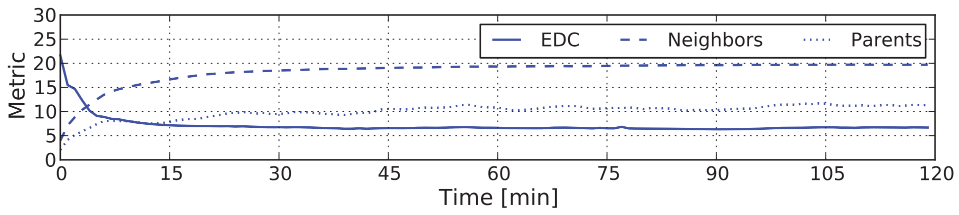
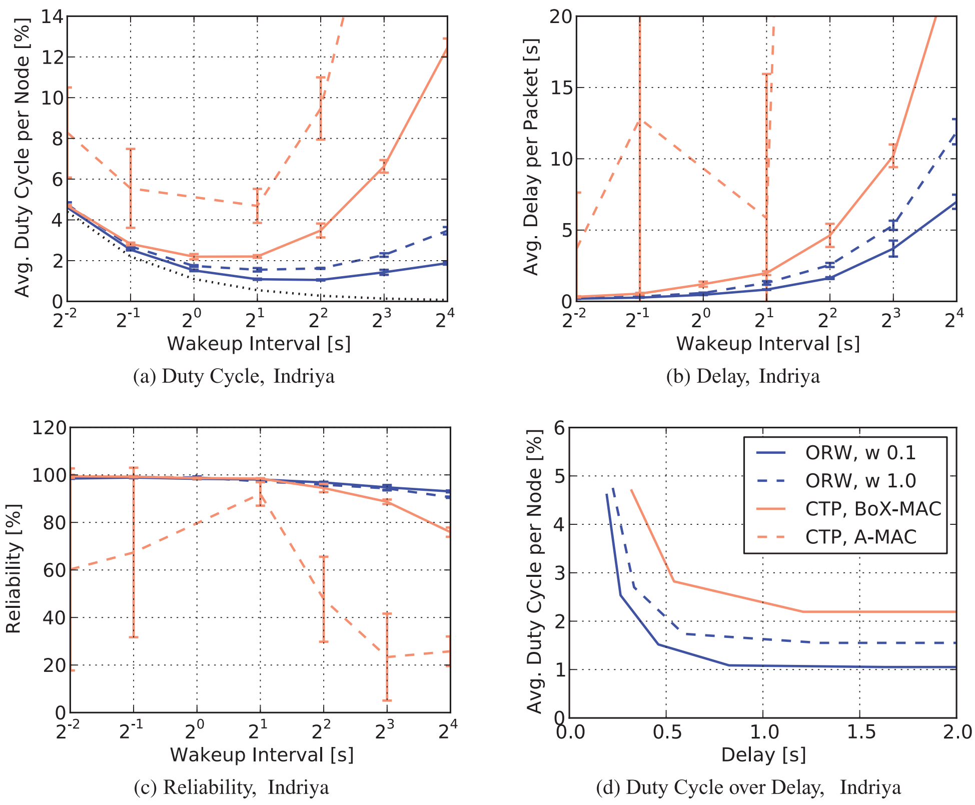


Fig. 18. Convergence of ORW (Indriya, 0dBm). Average per-node values for EDC, neighbors in the routing table, and parents selected in the forwarder set.

shows that the routing metric EDC reaches an initial stable point within the first five minutes without the need for extensive beaconing or detailed link estimation. Over time it optimizes slightly.

Similarly, nodes in ORW continue to add new neighbors for routing. However, these lead to only minimal improvements, as we see only minimal changes to the duty cycle and delay over time (see Figure 17). Overall, the results show that ORW stabilizes quickly without the need for expensive probing for link estimation and neighbor discovery. Additionally, Figure 17 shows that ORW maintains this stability even in presence of node failures.

4.2.6. Choice of Wakeup Interval. The previous experiments used a wakeup interval of 2 seconds, that is, a node wakes up every two seconds to receive data from neighboring nodes. We used this interval to ensure a fair comparison, as it leads to the optimal duty



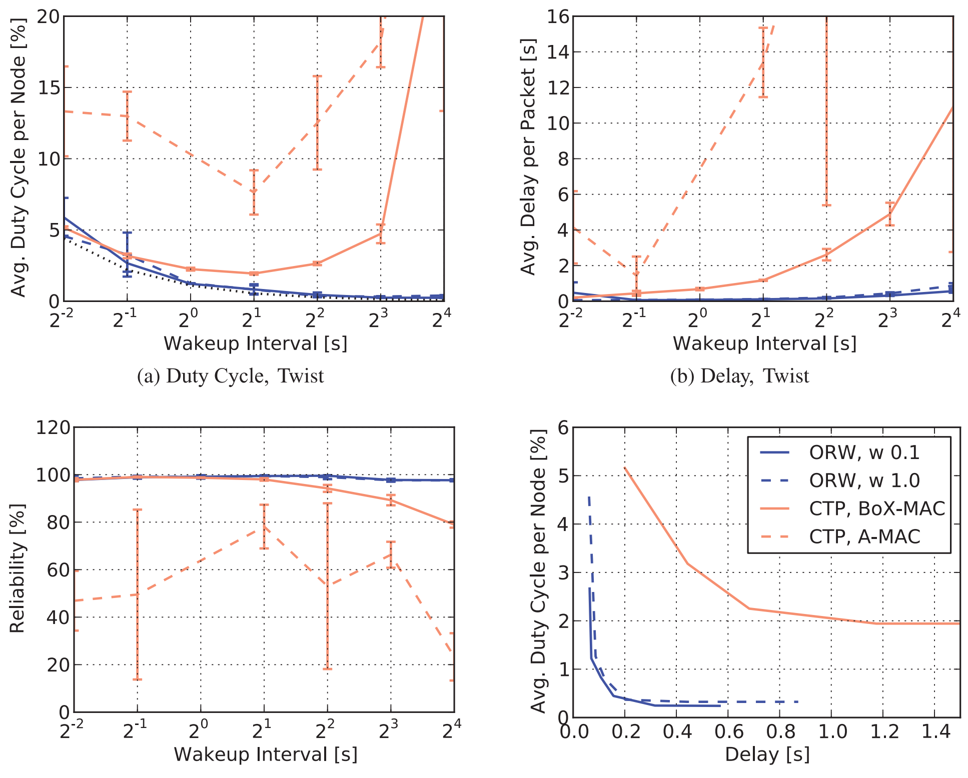
(a) Duty Cycle, Indriya (b) Delay, Indriya (c) Reliability, Indriya (d) Duty Cycle over Delay, Indriya

Fig. 19. Choice of wakeup intervals (Indriya testbed): Performance of ORW and CTP on Indriya (with TX power of 0dBm) for wakeup intervals between 0.25 and 16 seconds: ORW can operate at much lower wakeup intervals than CTP while achieving low delays and high reliability. The dotted line in (a) depicts the idle line of BoX-MAC. Some data points for A-MAC are omitted due to inconsistent results. For the same reason A-MAC is omitted from (d).

cycle in CTP and its default BoX-MAC at an inter-packet interval of 4 minutes (see Figure 19(a)).

In this section, we discuss the impact of the wakeup interval on duty cycle, delay, and reliability. Figure 19(a) and Figure 20(a) show that ORW benefits much more than CTP from reduced wakeup intervals. The figures also depict the idle duty cycle, namely, the energy that is consumed by just the wakeups of BoX-MAC without any data transfer (dotted lines in Figures 19(a) and 20(a)). This baseline defines the lower bound for the duty cycle. For Indriya (see Figure 19(a)) ORW stays closer to this line than CTP and in the dense Twist testbed (see Figure 20(a)) ORW is marginally above the baseline throughout all experiments. These results show that ORW efficiently exploits network density. Delay increases with increased wakeup intervals (see Figure 19(b) and 20(b)). However, the increase for ORW is significantly lower than for CTP with BoX-MAC. Figure 19(d) and Figure 20(d) show the resulting duty cycle for a given average delay. This underlines that ORW can operate at much lower duty cycles for a given average delay.

Both CTP and ORW show high reliability (see Figures 19(c) and 20(c)). However, at high wakeup intervals reliability of CTP decreases due to queue overflows on individual nodes. In contrast, ORW avoids this by using multiple forwarders. The results for other inter-packet intervals, radio channels, and testbeds show similar performance gains



(a) Duty Cycle, Twist (b) Delay, Twist

(c) Reliability, Twist (d) Duty Cycle over Delay, Twist

Fig. 20. Choice of wakeup intervals (Twist testbed): Performance of ORW and CTP on Twist (with TX power of 0dBm) for wakeup intervals between 0.25 and 16 seconds: ORW can operate at much lower wakeup intervals than CTP while achieving low delays and high reliability. The dotted line in (a) depicts the idle line of BoX-MAC. Some data points for A-MAC are omitted due to inconsistent results. For the same reason A-MAC is omitted from (d).

of ORW over CTP. As reference, the figures also depict results for CTP on A-MAC: However, it does not reach the performance of BoX-MAC for multihop routing with CTP. While we cannot conclude on the exact reasons, our traces indicate that its probes and loss of synchronization at low wakeup rates increase duty cycles and delays.

4.3. Discussion and Limitations

After evaluating our anycast routing scheme in simulation and comparing it to the state-of-the-art CTP collection protocol in testbeds, we comment on the results and discuss limitations of our work.

4.3.1. Discussion. ORW improves duty cycles and delays significantly while achieving similar reliability and transmission counts when compared to the state-of-the-art. Our results show an average decrease in duty cycle by about 50%; individual nodes improve up to 90%. Similarly, delay is decreased by 30% to 90% depending on network density.

Anycast forwarding allows ORW to forward a packet faster than traditional unicast routing. Overall, such a design performs the best at high network densities, as these give the most choices for forwarding. As a result, ORW shows the best results for dense topologies, that is, both in physical node density of the testbeds and at high transmission power. Similarly, its optimal duty cycle is at lower wakeup rates when compared to CTP. Hence, our results show that ORW can operate at much lower wakeup rates without major impact on reliability or delay (see Figures 19 and 20). At lower densities, ORW still outperforms CTP, but its benefits decrease (see Table IV). Similarly, at high wakeup rates, the delay and energy advantages of ORW decrease. We believe that ORW can strongly benefit from the integration of adaptive duty cycling [Jurdak et al. 2007; Meier et al. 2010; Hansen et al. 2012; Puccinelli et al. 2012]: exploiting its anycast forwarding, it could efficiently adapt wakeup rates to traffic load and network density.

Overall, the opportunistic nature of ORW allows to take the state of wireless links into account and delay the decision of selecting a forwarder until the packet has been received. As a result, it reflects temporal and spatial diversity of wireless links (see Figure 16) and increases the resilience of routing to link dynamics and node failures (see Figure 17).

4.3.2. Limitations. ORW targets applications with lifetime demands on the order of months or years, which are typical deployment scenarios in WSNs. Commonly, such applications rely on duty-cycled low-power networking with wakeup rates on the order of seconds. Our evaluation shows that ORW achieves the strongest improvement compared to CTP at such low wakeup rates (see Figures 19 and 20).

At high wakeup rates ORW loses some of its benefits. In this case the baseline cost of the MAC layer dominates the overall energy consumption, and both CTP and ORW show similar performance in terms of energy and delay (see Figures 19(a) and 20(a)). Moreover, at such high wakeup rates, we see an increased risk of multiple forwarders being awake concurrently (see Figure 5). This demands more coordination and increases the risk of duplicates and thus increases the forwarding cost of each packet. We believe that in such settings, that is, at very high wakeup rates leading to many nodes being awake concurrently or when no duty cycling is employed, traditional opportunistic routing schemes such as ExOR [Biswas and Morris 2005] or MORE [Chachulski et al. 2007] would be better suited.

ORW focuses on collection applications with low data rates. We believe that its design is not well suited for high-throughput settings such as bulk transfers: at high data rates nodes are transmitting bursts of packets from their forwarding buffers. Transmitting long bursts wakes up multiple neighbors, that in turn compete to be forwarders. As a result, long bursts increase the risk of duplicates. This demands for new coordination algorithms to ensure unique forwarders. In our ongoing work, we are extending ORW to temporarily lock onto a single forwarder (once it has found one) when transmitting a burst of packets. This prevents neighboring nodes from competing during such a burst and allows them to return to sleep immediately.

While ORW is agnostic to the underlying MAC scheme, it shows the strongest improvements for asynchronous MAC layers. For phase-locking MAC layers, we expect ORW to show similar improvements for delay but less benefits in terms of energy. However, in dense deployments such as Twist (see Figure 20(a)) the duty cycle in ORW closely approaches the idle baseline of the MAC layer. This is also the cost of an ideal phase-locked MAC, without considering its overhead in terms of time synchronization and guard times.

To ensure a fair comparison with CTP, our current implementation of ORW is tailored to collection applications with a single sink. Thus, we currently do not support mesh routing (or multiple sinks). However, the design of ORW is generic. When applications require it, this can be directly integrated by adding an extra header field that notes the intended destination next to the already existing requested routing progress (see Section 3.7).

5. RELATED WORK

In this section we discuss related work on opportunistic and adaptive routing in WSNs. Opportunistic routing itself is discussed in the preliminaries in Section 2.1.

GeRaF [Zorzi and Rao 2003a, 2003b] and RAW [Paruchuri et al. 2004] pioneered the concept of anycast routing in duty-cycled wireless sensor networks. They utilize geographic routing to determine the routing progress of its neighboring nodes and a busy tone protocol to ensure a unique forwarder. CMAC [Liu et al. 2009, 2007] combines the concepts of GeRaF and ExOR: It includes prioritized forwarders, slotted acknowledgments, and overhearing of acknowledgments to determine a unique forwarder as in ExOR. Relying solely on geographic routing, both do not address key challenges for opportunistic routing in duty-cycled WSNs such as anycast routing metrics and wireless link dynamics.

The application of opportunistic routing in WSNs also received great attention from a more theoretical perspective. Similar to our work, many [Kim et al. 2010; Dubois-Ferrié et al. 2011; Mao et al. 2011; Schaefer et al. 2009; Ashref et al. 2010; Basu and Chau 2008; Lu and Wu 2009; Xue et al. 2010; Zhong and Nelakuditi 2007; Kim and Liu 2008] consider anycast routing in WSNs. Their models and simulation results show that opportunistic routing can improve energy efficiency and delay when compared to traditional unicast routing. Their results strongly motivated our work. However, they omit the real-world challenges in terms of duty cycling and link dynamics that this article addresses.

For example, LCAR [Dubois-Ferrié et al. 2011] assigns a relay candidate set to each node in order to minimize the expected cost of forwarding a packet to the destination. The expected cost is recursively constructed by assuming that the relay nodes already know their own forwarding cost to the destination. Mao et al. [2011] study the selection and prioritizing of the forwarding list to minimize the overall energy consumption of WSNs. Similar to LCAR, it calculates an expected cost for every node to send a packet to a target. An optimal forwarder set that minimizes this cost is constructed with a greedy algorithm. In contrast to our work, these do not consider the duty cycling in their theoretical model.

Joint study of anycast forwarding and duty cycling has been done in Kim et al. [2010] and Dubois-Ferrié [2006]. Kim et al. [2010] investigate the optimal anycast forwarding policy for a Poisson wakeup model to minimize the expected end-to-end delay in event-driven WSNs. On the other hand, Dubois-Ferrié [2006] considers a preamble-based duty-cycled WSN and provides routing metrics for synchronous and asynchronous wakeups. However, in both solutions the forwarder selection merely depends on the wakeup process and does not take the probability of link failure into account: We consider this a key requirement, as (due to the low-power nature of the WSNs) wireless links are highly dynamic [Srinivasan et al. 2008]. Other approaches to opportunistic forwarding [Liu et al. 2010; Autenrieth and Frey 2011; Pavković et al. 2011; Unterschütz et al. 2012] are routing agnostic and do not include energy-efficient routing metrics nor tailor link estimation to anycast routing.

Commonly, routing protocols employ link estimation to identify long-term stable links [Fonseca et al. 2007; Woo et al. 2003] and to utilize these for routing. Recent approaches suggest to employ highly adaptive link estimation to identify dynamic links [Srinivasan et al. 2008, 2010] and to predict whether an intermediate link is temporarily reliable [Alizai et al. 2009; Liu and Cerpa 2011, 2012]. For example, BRE [Alizai et al. 2009] employs short-term link estimation [Becher et al. 2008] to reduce hop counts: when a far-ranging link of intermediate quality becomes temporarily available, BRE uses it to short-cut in the routing tree. In duty-cycled environments, BRE and related approaches such as 4C [Liu and Cerpa 2011] and Talent [Liu and Cerpa 2012] show two key limitations: (1) Routing short-cuts are only stable for a couple of milliseconds, making it difficult to exploit them in low-traffic scenarios. (2) Their link estimators need to overhear data traffic to determine possible short-cuts. In heavily duty-cycled systems where nodes are asleep most of the time, this is not practical. Similarly to these approaches, ORW exploits intermediate links for packet forwarding: Utilizing the first awoken neighbor that provides sufficient routing progress, ORW employs both stable and dynamic links for packet forwarding. However, due to its anycast nature, it does not demand for bursty traffic to make short-cuts in the routing tree. Thus, the benefits of ORW are also present in both heavily duty-cycled networks as well as systems with nonbursty and low-traffic rates.

Adaptive and low-power routing in WSNs proposes a dynamic change of parents in the routing topology. DSF [Gu and He 2007] selects the next hop of a packet based on the sleep schedule of neighboring nodes and different metrics such delay, reliability, and energy consumption. Similar to ORW, DSF shows strong improvements over unicast routing in these metrics. However, it focuses on synchronized networks. Furthermore, it requires iterative message exchanges to stabilize the forwarding schedules of all nodes, leading to control traffic overhead in the presence of dynamic links. The Backpressure Routing Protocol, BRP [Moeller et al. 2010], forwards packets to the neighbor with the lowest queue level. This improves throughput when compared to traditional unicast routing, while increasing delay. However, BRP can only be applied when the overall system is saturated, that is, nodes always have packets to forward. This is rare in WSN application scenarios as these commonly show low traffic rates. The Broadcast-Free Collection protocol, BFC [Puccinelli et al. 2012], presents a routing protocol that—similar to ORW—operates without link probing. BFC was developed independently of ORW and focuses on traditional unicast routing. As a result, it does not benefit from the energy efficiency and low delay that anycast routing in ORW provides. ORPL [Duquennoy et al. 2013] integrates the design concepts of ORW, namely anycast routing in duty-cycled WSNs and the routing metric EDC, into RPL [Winter et al. 2012].

6. CONCLUSIONS

This article introduces Opportunistic Routing for Wireless sensor networks (ORW), targeting applications with low-radio duty cycles. A packet in ORW is forwarded by the first awoken node that successfully receives it and offers routing progress. This provides two key benefits over traditional unicast routing: by utilizing all neighbors as possible next hops, ORW reduces delay and energy consumption significantly when compared to unicast routing; additionally, it improves the resilience to link dynamics and node failures.

Overall, ORW tailors the concept of opportunistic routing to the specific demands of WSNs and duty cycling. It integrates three key techniques: (1) EDC, a new anycast routing metric to reflect the multipath nature of opportunistic routing. EDC is based on precise analytical expressions for the expected number of wakeups required for opportunistic forwarding. (2) Mechanisms for selecting optimal forwarder sets in duty-cycled opportunistic routing are another mechanism. (3) Coarse-grained, long-term link estimation is the last. Our results show that ORW doubles energy efficiency in dense networks. It reduces duty cycles on average by 50% and delays by 30% to 90% while achieving reliability and transmission counts similar to the state-of-the-art.

We have made the source code of ORW publicly available at https://github.com/olafland/orw as a means to foster further research into opportunistic routing in duty-cycled WSNs.

APPENDIXES

A.1. Proof of Lemma 3.1

The first result can be verified by inspecting $f_i^{(1)}(\mathcal{A} \cup \{x\}) - f_i^{(1)}(\mathcal{A})$. For the second result one can check that $f_i^{(2)}(\mathcal{A} \cup \{k\}) - f_i^{(2)}(\mathcal{A}) > 0$ given that $c_k > c_j$ for all $j \in \mathcal{A}$. □

A.2. Proof of Lemma 3.2

The sign of $c_k - f_i(\mathcal{A}) + w$ is the same as the sign of $f_i(\mathcal{A} \cup \{k\}) - f_i(\mathcal{A})$, since

$$f_i(\mathcal{A} \cup \{k\}) - f_i(\mathcal{A}) = \frac{p(i,k) \cdot (c_k - f_i(\mathcal{A}) + w)}{\sum_{j \in \mathcal{A}} p(i,j) + p(i,k)}$$

and $p(i,k) > 0$. Our result follows. □

A.3. Proof of Lemma 3.3

Consider

$$f_i(\mathcal{A} \cup \{k\}) - c_k = \frac{f_i(\mathcal{A}) - c_k + w \frac{p(i,k)}{\sum_{j \in \mathcal{A}} p(i,j)}}{1 + \frac{p(i,k)}{\sum_{j \in \mathcal{A}} p(i,j)}}.$$

Our assumptions imply that the right-hand side is positive, hence $f_i(\mathcal{A} \cup \{k\}) > c_k$. □

A.4. Proof of Theorem 3.4

Assume that $\mathcal{F}^\star(i) = \{\pi(1), \ldots, \pi(k)\} \backslash \pi(m)$ for some $m < k$, namely, a node m with $c_m < c_k$ has been excluded from the forwarder set. Note that $c_j \leq f_i(\mathcal{F}^\star(i) \backslash \pi(j)) - w,\ \forall j \in \mathcal{F}^\star(i)$ because otherwise, according to Lemma 3.2, $f_i(\mathcal{F}^\star(i)) > f_i(\mathcal{F}^\star(i) \backslash \pi(j))$ resulting that $\mathcal{F}^\star(i)$ is not the optimum. Moreover, from $c_j \leq f_i(\mathcal{F}^\star(i) \backslash \pi(j)) - w$ and Lemma 3.3 we conclude $c_j \leq f_i(\mathcal{F}^\star(i)) - w,\ \forall j \in \mathcal{F}^\star(i)$. Lemma 3.2 ensures that adding m with $c_m < c_k$ will decrease $f_i(\mathcal{F}^\star(i))$, that is, $f_i(\mathcal{F}^\star(i) \cup \{c_m\}) < f_i(\mathcal{F}^\star(i))$ so $\mathcal{F}^\star(i)$ is not optimal and a contradiction is achieved. □

A.5. Proof of Lemma 3.5

The proof follows from the Lemma 3.2 and noting that

$$f_i(\mathcal{A} \cup \{k'\}) - f_i(\mathcal{A} \cup \{k\}) = \frac{(p(i,k) - p(i,k'))(f_i(\mathcal{A}) - c_k - w)}{\Big(p(i,k) + \sum_{j \in \mathcal{A}} p(i,j)\Big)\left(\frac{p(i,k')}{\sum_{j \in \mathcal{A}} p(i,j)} + 1\right)} > 0. \quad \square$$

A.6. Proof of Lemma 3.6

According to Lemma 3.2, at each iteration t of Algorithm 1 a new member $j \in \mathcal{N}(i)$ is added to the forwarder set $\mathcal{F}^\star(i)$ for node $i \in \mathcal{N}$ if $\mathrm{EDC}(j) < \mathrm{EDC}(i) - w$. By Lemma 3.3, this event can only happen since at iteration $t-1$ node $i \notin \mathcal{F}^\star(j)$, that is, otherwise $\mathrm{EDC}(j) > \mathrm{EDC}(i)$. Furthermore, upon adding j to $\mathcal{F}^\star(i)$ we have $\mathrm{EDC}(i) = f(\mathcal{F}^\star(i) \cup \{j\}) > \mathrm{EDC}(j)$ by Lemma 3.3, which ensures that node i cannot be added to the optimal forwarder set $\mathcal{F}^\star(j)$ of j in future iterations of the algorithm. Thus any two nodes $i, j \in \mathcal{N}$ can only be connected by either an arc (i,j) or (j,i) in each rooting topology $\mathcal{G}^{(t)}$.

Let us now consider a path of arbitrary length $P_i = \{i, j_1, j_2, \ldots, j_n\}$ from node i to the sink in $\mathcal{G}^{(t)}$, with $j_1 \in \mathcal{F}^\star(i)$, $j_2 \in \mathcal{F}^\star(j_1)$, and so on. Let's assume that P_i has a loop involving node i, namely there exists a $j_k \in P_i$ such that $i \in \mathcal{F}^\star(j_k)$. Hence, by Lemma 3.2 $\mathrm{EDC}(i) < \mathrm{EDC}(j_k) - w$. However, by applying Lemma 3.3 to each hop of the path P_i we have that $\mathrm{EDC}(i) > \mathrm{EDC}(j_n),\ \forall j_n \in P_i$ and a contradiction is achieved. □

A.7. Proof of Theorem 3.7

Let π be an ordered set of nodes with cardinality $|\mathcal{N}|$ in which nodes are sorted increasingly based on their EDC values. It is sufficient to prove that in the $m \geq 1$-th pass of the for-loop in Algorithm 1 at line 5, at least one node will find its optimal forwarder list and remains unchanged for the rest of the algorithm; in other words, the first m nodes with lower EDC values in π converge to their optimality (in the sense of the EDC metric) after $m-1$ executions of the for-loop. We prove this by induction. The inductive basis is trivial since the first node is the sink, which has the lowest EDC. Now, assume that our claim holds for the first $m-1$ steps. During the m-th pass, the first m node will not update their EDCs and the corresponding forwarders set since they converged to their optimality. However, the remaining $|\mathcal{N}| - m$ nodes still may insert beneficial forwarders in ascending order of their EDC. After the m-th iteration, there will be one (pair of) node(s) with the same minimum value of EDC between remaining $|\mathcal{N}| - m$ nodes. For the moment assume there is only one node with this condition denoted by i_{m+1}. We claim that i_{m+1} will converge to optimality in the m-th iteration. Because of the monotonically increasing property stated in Lemma 3.3, the EDC of the $|\mathcal{N}| - m$ remaining nodes will not be less than the first m nodes in π. It is easy to see that i_{m+1} after the m-th iteration selects forwarders only from the first m nodes according to the strict monotonicity property stated in Lemma 3.2. Hence, $\forall j \in \mathcal{F}(i_{m+1})$, EDC($j$) is optimal due to the inductive hypothesis. Since EDC(i_{m+1}) cannot be improved in upcoming iterations (due to the monotonically increasing property stated in Lemma 3.3) we conclude that i_{m+1} converges to the optimality of the metric. For the case of multiple nodes with minimum EDC values after m-th iteration, all of them converge to the optimality due to the fact that all of them are only utilizing the first m nodes in π for their forwarders set and, further, adding each other or the remaining $|\mathcal{N}| - m$ nodes will not decrease their EDC value. $\square$

REFERENCES

M. H. Alizai, O. Landsiedel, J. A. Bitsch Link, S. Gotz, and K. Wehrle. 2009. Bursty traffic over bursty links. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'09)*.

G. Allen, P. Swieskowski, and M. Welsh. 2005. Motelab: A wireless sensor network testbed. In *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'05)*.

F. Ashref, R. H. Kravets, and N. H. Vaidya. 2010. Exploiting routing redundancy using MAC layer anycast to improve delay in WSN. *SIGMOBILE Mobile Comput. Comm. Rev.* 14, 49–51.

M. Autenrieth and H. Frey. 2011. PaderMAC: A low-power, low-latency MAC layer with opportunistic forwarding support for wireless sensor networks. In *Proceedings of the 10th International Conference on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW'11)*.

P. Basu and C.-K. Chau. 2008. Opportunistic forwarding in wireless networks with duty cycling. In *Proceedings of the ACM Workshop on Challenged Networks (CHANTS'08)*.

A. Becher, O. Landsiedel, G. Kunz, and K. Wehrle. 2008. Towards short-term wireless link quality estimation. In *Proceedings of the 5th ACM Workshop on Embedded Networked Sensors (HotEmNetS'08)*.

S. Biswas and R. Morris. 2005. ExOR: Opportunistic multi-hop routing for wireless networks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SigComm'05)*.

M. Buettner, G. V. Yee, E. Anderson, and R. Han. 2006. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'06)*.

S. Chachulski, M. Jennings, S. Katti, and D. Katabi. 2007. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SigComm'07)*.

R. R. Choudhury and N. H. Vaidya. 2004. MAC-layer anycasting in ad hoc networks. *ACM Comput. Comm. Rev.* 34, 1, 75–80.

D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. 2003. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom'03)*.

M. Doddavenkatappa, M. C. Chan, and A. Ananda. 2011. Indriya: A low-cost, 3D wireless sensor network testbed. In *Proceedings of the International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom'11)*.

A. Dubois-Ferrie. 2006. Anypath routing. Ph.D. thesis, EPFL. http://infoscience.epfl.ch/record/89172.

A. Dubois-Ferrie, M. Grossglauser, and M. Vetterli. 2011. Valuable detours: Least-cost anypath routing. *IEEE/ACM Trans. Netw.* 19, 2, 333–346.

S. Duquennoy, O. Landsiedel, and T. Voigt. 2013. Let the tree bloom: Scalable opportunistic routing with orpl. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'13).*

P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. 2010. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'10).*

R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. 2007. Four bit wireless link estimation. In *Proceedings of the Workshop on Hot Topics in Networks (HotNets'07).*

E. Ghadimi, O. Landsiedel, P. Soldati, and M. Johansson. 2012. A metric for opportunistic routing in duty cycled wireless sensor networks. In *Proceedings of IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'12).*

O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. 2009. Collection tree protocol. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'09).*

Y. Gu and T. He. 2007. Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'07).*

V. Handziski, A. Kopke, A. Willig, and A. Wolisz. 2006. TWIST: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *Proceedings of the International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality (REALMAN'06).*

M. Hansen, B. Kusy, R. Jurdak, and K. Langendoen. 2012. AutoSync: Automatic duty-cycle control for synchronous low-power listening. In *Proceedings of the IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'12).*

R. Jurdak, P. Baldi, and Lopes, C. Videir. 2007. Adaptive low power listening for wireless sensor networks. *IEEE Trans. Mobile Comput.* 6, 8, 988–1004.

D. Kim and M. Liu. 2008. Optimal stochastic routing in low duty-cycled wireless sensor networks. In *Proceedings of the 4th Annual International Conference on Wireless Internet (WICON'08).*

J. Kim, X. Lin, N. B. Shroff, and P. Sinha. 2008. On maximizing the lifetime of delay-sensitive wireless sensor networks with anycast. In *Proceedings of the IEEE International Conference on Computer Communications (InfoCom'08).*

J. Kim, X. Lin, N. B. Shroff, and P. Sinha. 2010. Minimizing delay and maximizing lifetime for wireless sensor networks with anycast. *IEEE/ACM Trans. Netw.* 18, 515–528.

O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. 2012. Low power, low delay: Opportunistic routing meets duty cycling. In *Proceedings of the 11th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'12).*

P. Larsson. 2001. Selection diversity forwarding in a multihop packet radio network with fading channel and capture. *ACM Mobile Comput. Comm. Rev.* 5, 47–54.

S. Liu, K.-W. Fan, and P. Sinha. 2007. CMAC: An energy efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks. In *Proceedings of IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07).*

S. Liu, K.-W. Fan, and P. Sinha. 2009. CMAC: An energy-efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks. *ACM Trans. Sensor Netw.* 5, 29:1–29:34.

S. Liu, M. Sha, and L. Huang. 2010. ORAS: Opportunistic routing with asynchronous sleep in wireless sensor networks. In *Proceedings of the 2nd International Conference on Future Computer and Communication (ICFCC'10).*

T. Liu and A. Cerpa. 2011. Foresee (4C): Wireless link prediction using link features. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN'10).*

T. Liu and A. Cerpa. 2012. TALENT: Temporal adaptive link estimator with no training. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'12).*

M. Lu and J. Wu. 2009. Opportunistic routing algebra and its applications. In *Proceedings of the IEEE International Conference on Computer Communications (InfoCom'09).*

X. Mao, S. Tang, X. Xu, X. Li, and H. Ma. 2011. Energy-efficient opportunistic routing in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* 22, 11, 1934–1942.

A. F. Meier, M. Woehrle, M. Zimmerling, and L. Thiele. 2010. Zerocal: Automatic MAC protocol calibration. In *Proceedings of 6th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'10).*

S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. 2010. Routing without routes: The backpressure collection protocol. In *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'10).*

D. Moss and P. Levis. 2008. BoX-macs: Exploiting physical and link layer boundaries in low-power networking. Tech. rep, SING-08-00, Stanford.

V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S. Iyengar. 2004. Random asynchronous wakeup protocol for sensor networks. In *Proceedings of the 1st International Conference on Broadband Networks (BroadNets'04).*

B. Pavkovic, F. Theoleyre, and A. Duda. 2011. Multipath opportunistic rpl routing over IEEE 802.15.4. In *Proceedings of the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'11).*

J. Polastre, J. Hill, and D. Culler. 2004. Versatile low power media access for wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (Sen-Sys'04).*

D. Puccinelli, M. Zuniga, S. Giordano, and P. J. Marron. 2012. Broadcast-free collection protocol. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'12).*

G. Schaefer, F. Ingelrest, and M. Vetterli. 2009. Potentials of opportunistic routing in energy-constrained wireless sensor networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN'09).*

K. Srinivasan, M. Jain, J. I. Choi, T. Azim, E. S. Kim, P. Levis, and B. Rishnamachari. 2010. The factor: Inferring protocol performance using inter-link reception correlation. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom'10).*

K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. 2008. The β factor: Measuring wireless link burstiness. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'08).*

S. Unterschutz, C. Renner, and V. Turau. 2012. Opportunistic, receiver-initiated data-collection protocol. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN'12).*

J. Vanhie-Van Gerwen, E. De Poorter, B. Latre, I. Moerman, and P. Demeester. 2010. Real-life performance of protocol combinations for wireless sensor networks. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'10).*

T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. 2012. RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550 (proposed standard). http://tools.ietf.org/html/rfc6550.

A. Woo, T. Tong, and D. Culler. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'03).*

Y. Xue, M. Vuran, and B. Ramamurthy. 2010. Cost efficiency of anycast-based forwarding in duty-cycled WSNS with lossy channel. In *Proceedings of the IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'10).*

W. Ye, F. Silva, and J. Heidemann. 2006. Ultra-low duty cycle MAC with scheduled channel polling. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'06).*

Z. Zhong and S. Nelakuditi. 2007. On the efficacy of opportunistic routing. In *Proceedings of the IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07).*

M. Zorzi and R. R. Rao. 2003a. Geographic random forwarding (geraf) for ad hoc and sensor networks: Energy and latency performance. *IEEE Trans. Mobile Comput.* 2, 349–365.

M. Zorzi and R. R. Rao. 2003b. Geographic random forwarding (geraf) for ad hoc and sensor networks: Multihop performance. *IEEE Trans. Mobile Comput.* 2, 337–348.

Received January 2013; revised August 2013; accepted September 2013