EDAD: Energy-Centric Data Collection with Anycast in Duty-Cycled Wireless Sensor Networks

Mathew L. Wymore, Yang Peng, Xiaoyun Zhang, and Daji Qiao
Iowa State University, Ames, IA, USA
{mlwymore, yangpeng, zxydut, daji}@iastate.edu

***Abstract*—Recent efforts in applying anycast techniques to duty-cycled wireless sensor networks have shown promising results in terms of reduced delay and energy consumption. This paper further increases the energy savings by introducing EDAD, an energy-centric cross-layer data collection protocol designed for anycast communications in asynchronously duty-cycled wireless sensor networks. EDAD uses a new anycast routing metric, EEP, that minimizes the expected energy consumed along the path of a packet and automatically adapts to network settings. Simulation results show that EDAD consumes less energy than similar existing protocols, while maintaining a comparable delay and high delivery rate.**

I. Introduction

Emerging applications such as remote infrastructure health monitoring demand reliable and cost-efficient methods for collecting and reporting sensor data. Small, self-contained sensing nodes connected in a wireless sensor network (WSN) have the potential to fulfill this need. In WSNs, nodes communicate with each other using low-power wireless technology, usually over multi-hop routes. These nodes are powered by batteries or energy harvesting techniques and are tightly energy constrained. Since energy use in WSNs is known to be dominated by the radio hardware, routing and media access control (MAC) protocols for WSNs have focused on reducing radio on-time of nodes.

Primarily, this reduction in radio on-time is achieved with *duty-cycling*. In duty-cycling, nodes spend most of their time with their radios off, or *sleeping*, and only wake to communicate, as determined by the MAC protocol. MACs can align duty cycles synchronously, as in S-MAC [1] or T-MAC [2], but synchronous methods require significant overhead to propagate schedules and maintain clock synchronization.

Therefore, most recent research has focused on asynchronous MACs. These can be sender-initiated, such as X-MAC [3], in which a node with a packet to send repeatedly transmits short wakeup packets until a receiver wakes and responds. BoX-MAC-2 [4] further streamlines this process by repeatedly sending the full data packet, so a receiver only needs to respond with an acknowledgement. Asynchronous MACs can also be receiver-initiated, such as RI-MAC [5], in which a node with a packet to send wakes and politely listens for an advertisement beacon from the next hop receiver, leaving the channel open for other nodes to communicate in the meantime.

A cross-layer technique that has recently been shown to increase efficiency in WSNs is *anycast*, in which a packet only needs to be sent to one of a set of *potential forwarders*, neighbors that provide acceptable progress. Anycast has been shown to reduce energy consumption and delay and to increase reliability in duty-cycled networks by allowing nodes to dynamically take advantage of available links [6]–[8]. However, existing anycast data collection schemes do not specifically minimize energy consumption, and WSN nodes need as small an energy budget as possible to achieve sustainable operation.

To address this need, this paper presents EDAD: Energy-Centric Data Collection with Anycast in Duty-Cycled WSNs. EDAD proposes a new anycast routing metric called Expected Energy Consumed Along the Path (EEP) that, contrary to previous anycast routing metrics,

1) is designed to minimize energy, and
2) automatically adapts itself to network settings.

The next section investigates related work. EDAD's design, including EEP, is presented in Section III. Section IV presents simulation results that show EDAD reduces energy consumption compared to state-of-the-art alternatives, and Section V concludes this paper.

II. Related Work

Data collection is an application scenario in which source nodes generate data, such as sensor readings, and send the data through a network to a special node known as the *sink*, which typically transmits the data out of the network for processing. The Collection Tree Protocol (CTP) [9] is a classic unicast data collection protocol in which each node maintains a *routing metric* value that estimates the node's "distance" from the sink. In CTP, the distance is measured as the expected transmission attempts required to reach the sink (ETX). Nodes form a routing metric gradient that spreads outward from the sink, and packets are routed by forwarding to nodes with lower routing metric values, or nodes that provide *routing progress*. In CTP, each node chooses its neighbor with the lowest routing metric value as its parent, forming a tree topology, as shown in Fig. 1a. When anycast is applied to data collection, this tree becomes a destination-oriented directed acyclic graph (DODAG) [7], as shown in Fig. 1b.

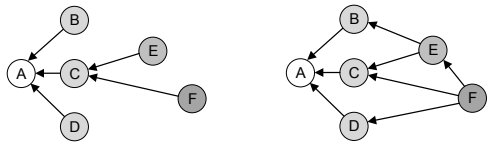


Fig. 1. The effect of anycast on a data collection topology. (a) shows a traditional gradient-based tree topology, where node A is the sink. Each node chooses one parent as its next hop forwarder. (b) shows anycast applied to the same network, creating a DODAG topology in which each node can have multiple potential forwarders.

Anycast for WSNs has been analytically studied in papers such as [10] and [6], and in protocols such as GeRaF [11], CMAC [12], and A^2-MAC [13]. These studies do not define a practical anycast routing metric, with most of them relying on geographical positions of nodes acquired by an external

This work is partially funded under the U.S. National Science Foundation Grant No. 1069283.

978-1-4799-8406-0/15/$31.00 ©2015 IEEE

process. In contrast, EDAD presents EEP, a new anycast metric that is calculated using only information from neighboring nodes. [14] does propose an anycast routing metric, but for a synchronous MAC protocol, while EDAD's design utilizes an asynchronous MAC protocol and the associated benefits.

ORiNoCo [8], a receiver-initiated anycast data collection protocol, avoids the anycast metric problem by forwarding a packet to any neighbor that advertises a path within a difference Θ of the best known path. ORiNoCo relies on the arbitrary choice of Θ to adjust the protocol to a particular network, whereas EDAD's EEP automatically adapts itself to given network settings, such as the average amount of time between node wakeups. Also, EDAD maintains a neighbor table and an explicit forwarder set to allow EEP fine-grained control over potential forwarders, allowing it to make routing decisions based on energy considerations.

ORW [7, 15] and the follow-up ORPL protocol [16] use an anycast metric called Expected Duty Cycled Wakeups (EDC), an approximation of the expected number of wakeups required to deliver a packet to the sink [15]. The EDC calculation factors in both link qualities and the number of potential forwarders available to a node. However, the physical meaning of EDC is not intuitive, whereas EDAD's EEP metric uses a clearly defined unit of energy, allowing EEP to minimize energy consumption. Additionally, similar to ORiNoCo, EDC relies on an arbitrary tunable parameter w intended to reflect the general cost of forwarding in a network, though the authors do provide $w = 0.1$ as a reasonable default for most scenarios.

III. EDAD Design

An overview of EDAD's components is shown in Fig. 2. At each node i, a neighbor table is kept updated with information from all of i's neighbors, as detailed in Section III-B4. A set of forwarders F_i is selected from the table, using the process described in Section III-A4, such that the minimum EEP_i is achieved. Data packets are then dynamically sent to the first available forwarder in F_i.

This work focuses on a receiver-initiated implementation, but the concepts of EDAD and EEP can be applied to any MAC protocol. The following sections explain the design of EEP, EEP forwarder set selection, anycast routing in EDAD, and some practical considerations, such as data packet retries, topology maintenance, and network initialization.

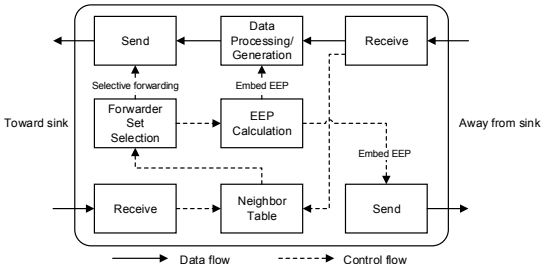


Fig. 2. Overview of EDAD's components at a node i. A neighbor table tracks the EEPs of i's neighbors, as well as estimates of the quality of their links with i. F_i, i's forwarder set, is selected from the neighbor table such that the minimum EEP_i is achieved. EEP_i is embedded in every outgoing packet. Data packets are forwarded to the first node in F_i that is available.

A. EEP: Expected Energy Consumed Along the Path

EEP is a new routing metric that estimates the expected energy consumed by nodes along the path of a packet as they relay it to the sink. EEP is designed to minimize energy for anycast on duty-cycled, asynchronous MAC protocols. The following sections describe the MAC model and simplifying assumptions on which EEP is based, as well as the observations that lead to EEP's energy-centric design. Then, EEP's calculation and forwarder set selection process will be presented.

1) System model: EEP works with any duty-cycled, asynchronous MAC protocol. An example of a receiver-initiated version of such a protocol is shown in Fig. 3. Between beacons, node i's potential forwarders sleep for a random amount of time in the interval $[0.5T_W, 1.5T_W]$, where T_W is a network setting known as the *wakeup interval*, and the expected sleep time for a node is T_W. A node i with a packet to send wakes and actively listens for an amount of time X_{min} before any potential forwarder $j \in F_i$ wakes and broadcasts a beacon to advertise its availability as a receiver. Assuming the wakeups of i's potential forwarders are uniformly distributed on T_W, the expected value of X_{min} is well-known, as follows:

$$E[X_{min}] = \frac{T_W}{|F_i| + 1}, \tag{1}$$

where $|F_i|$ is the number of nodes in i's forwarder set.

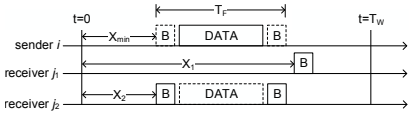


Fig. 3. Basic example of anycast with two potential forwarders and a receiver-initiated MAC. A sender i waits X_{min} before one of its forwarders, j_1 or j_2, wakes and sends a beacon. In this case, j_2 wakes first, so $X_{min} = X_2$, and i sends the data packet to j_2. Upon receipt, j_2 replies with an acknowledgement.

This model for X_{min} includes several simplifying assumptions. Because the sleep time for each node is random on an interval around T_W, X_{min} is actually expected to be larger than (1) suggests, due to a phenomenon resembling the *hitchhiker's paradox* [17]. However, as discussed in [8], (1) is a reasonable approximation and is used for its simplicity. Another assumption of (1) is that the beacon from the first $j \in F_i$ is successfully received by i. In reality, the beacon could be missed or garbled, meaning i would wait for the next j, which again suggests an expected wait longer than (1).

However, a counterbalancing assumption is that the wakeups of $j \in F_i$ are freshly uniformly distributed for each sender i in the path. Because the uniform distribution is not memoryless, this assumption pushes (1) toward overestimating expected wakeup delays. For example, suppose a node i can reach k either directly or through an intermediate node j. If j wakes first and receive's i's packet, then j likely does not need to wait as long as (1) suggests for k to wake, because part of the time until k wakes has already passed while the packet was queued at i and in transit to j.

Overall, these assumptions balance out well enough to use (1) as a simple model of the amount of time that a node with a packet to send has to wait until a forwarder wakes to receive it. This model provides part of the foundation for EEP calculation, described in Section III-A3.

2) Observations: EEP's design is based on the above model and the following observations of anycast routing in duty-cycled networks:

- Different forwarders provide different amounts of routing progress.

- Different forwarders require different numbers of expected transmission attempts before a successful transmission.
- A larger forwarder set for a node means less expected time for the node to wait for a forwarder to wake.
- A different forwarder set for a node may yield a different average routing progress per forwarder.

These observations reveal two tradeoffs in forwarder selection. The first is between end-to-end delay and the number of transmission attempts required to transmit the packet to the sink. Choosing only forwarders that provide low routing progress on reliable links will decrease the number of transmissions but increase the end-to-end delay. Choosing only forwarders with high routing progress on unreliable links will decrease end-to-end delay but increase the number of transmissions.

The second tradeoff is in regard to the size of the forwarder set. Adding more forwarders can decrease one-hop delay, but if the forwarders do not provide enough routing progress, the overall routing cost can be increased. EEP is designed to balance these tradeoffs.

In order to do so, both delay and transmissions are translated into units of energy. A unit of energy is defined as the amount of energy consumed by a node that is active (sending, receiving, or listening) for the length of time T_F required to transmit a data frame, as shown in Fig. 3. Sending power and listening or receiving power are assumed to be equal, a reasonable simplification for WSN radio hardware [18]. Along the path of a packet, one energy unit is consumed for each T_F of delay because the node with the packet currently queued is always active. Two energy units are consumed per transmission because both the sender and receiver are active, and each transmission lasts T_F.

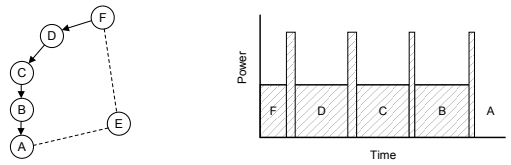
As shown in Fig. 4, this translation to energy allows the delay-transmissions tradeoff to be directly modeled. Fig. 4a shows an example of more delay over more hops on reliable links that require fewer transmissions, while Fig. 4b shows an example of less delay over fewer hops on unreliable links that require more transmissions. The shaded area is the energy consumed in each case. As discussed in Section III-A4, a forwarder set can be chosen that minimizes the expected shaded area, thus balancing the second tradeoff and minimizing energy consumed along the path of the packet.

As a final observation, because a packet is likely to spend much more time sitting in a queue than in transmission, the energy consumed is strongly affected by the delay. Therefore, this energy minimization model also tends to minimize delay.

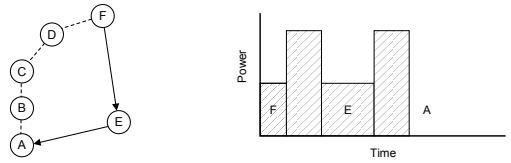
3) EEP calculation: Based on the above observations, EEP is calculated at each node i using the EEP of nodes $j \in F_i \subseteq N_i$, where F_i is i's forwarder set and N_i is i's set of communication neighbors. The EEP calculation also uses an estimate of the quality of the wireless link between i and j, measured as packet reception rate (PRR) and denoted as p_{ij}. The process for selecting F_i is detailed in the next section; for now, assume F_i is known. Then,

$$EEP_i = \frac{\sum_{j \in F_i} [EEP_j + 2/p_{ij}]}{|F_i|} + \frac{T_W/T_F}{|F_i| + 1}. \qquad (2)$$

The unit of EEP is the energy unit defined in the previous section. An example topology using EEP is shown in Fig. 5, where nodes B through F send to the sink node A. A solid arrow from i to j indicates that j is in F_i. The resulting EEP of each node is written below it.

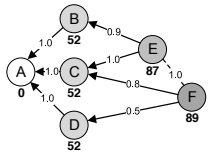


(a) Case when a path with many reliable hops is chosen. The end-to-end delay is larger, but less transmission attempts are required.



(b) Case when a path with fewer hops, utilizing unreliable links, is chosen. The end-to-end delay is less, but many transmission attempts are required.

Fig. 4. Illustration of the delay-transmissions tradeoff. The graphs show system-wide power usage over time as a packet is sent from F to A. The regions of lower power levels represent the times when a node along the path is listening, waiting for a forwarder to wake. The regions of higher power levels represent the time spent transmitting, when both sender and receiver are awake. The shaded area represents the total energy expended to route the packet from the source to the sink.



F's neighbor table

j	EEP_j	p_{Fj}
C	52	0.8
D	52	0.5
E	87	1.0

Fig. 5. Example of a topology using EEP when $T_W/T_F = 100$. Links are labeled with the p_{ij} of the link, and the EEP of each node is written below it. A solid arrow from i to j indicates that $j \in F_i$, while a dashed line indicates $j \notin F_i$. As shown in the table, even though F has a link to E, and E has a lower EEP than F, F does not include E in its forwarder set because adding E would increase EEP_F. In other words, with the given network settings, E does not provide enough routing progress for F to use it as a forwarder.

The EEP equation can be broken down into two distinct terms. The first term is the average expected energy expended along the multihop routes presented by the various $j \in F_i$. The energy for the route provided by a particular j is EEP_j, plus the expected number of transmission attempts required to reach j, multiplied by two because two energy units are consumed during each transmission. All $j \in F_i$ are assumed to have an equal probability of being the first to wake, so the average is obtained by dividing by $|F_i|$.

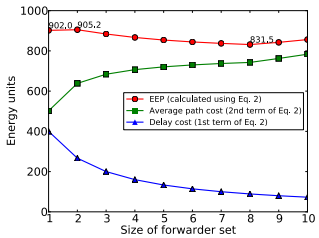
The second term of (2) is the expected energy consumed during the single-hop delay at i, meaning the time i spends actively waiting for some $j \in F_i$ to wake. Since one energy unit is consumed per T_F of activity, this quantity is (1) normalized to T_F.

4) EEP forwarder set selection: The forwarder set selection component shown in Fig. 2 determines which $k \in N_i$ are included in F_i to produce the minimum EEP_i. This process works as follows. All neighbors $k \in N_i$ are sorted in ascending order by a sorting key $c_k = EEP_k + 2/p_{ik}$. One at a time, the neighbors are added to F_i and EEP_i is calculated. When all neighbors have been added in order, and EEP_i has been calculated with each additional neighbor, the set of neighbors that yields the minimum EEP_i is used as F_i. As an example, note how, in Fig. 5, node F does not include

E in its forwarder set. Adding E would increase EEP_F, because the route provided by E would increase the expected average energy per route from $j \in F_i$, outweighing the decease in energy from the lower one-hop delay expected with one additional forwarder. This is how the forwarder set selection process balances the forwarder set size tradeoff discussed in Section III-A2.

This process guarantees the minimum EEP for a given N_i, without checking all possible permutations of F_i, because of the sorting of $k \in N_i$ by c_k. In more detail, the first term of (2) is the average c_j for $j \in F_i$. The second term of (2) decreases monotonically with the addition of more forwarders to F_i, regardless of the choice of forwarders added. Therefore, given an F_i and the set of remaining neighbors $k \in (N_i - F_i)$, the smallest EEP_i obtained from adding one more k to F_i will always be obtained by adding the k with the smallest c_k. Thus, only incremental F_i combinations, created by adding each k in order of c_k, need to be checked.

Some previous anycast works, such as [10], define a stopping condition for the forwarder set selection algorithm where the process can be stopped when the first ordered node that increases the routing metric value is found. In EEP, all incremental combinations, as described above, must be checked. As shown in Fig. 6, the EEP function is not necessarily convex, so a local minimum may occur, particularly when a group of forwarders have similar EEPs. However, even without an early stopping condition, forwarder selection for node i is efficient, with a computational complexity of $O(|N_i|)$.



j	c_j	EEP_i
1	502	902.0
2	775	905.2
3	775	884.0
4	775	866.8
5	776	853.9
6	778	844.5
7	780	837.3
8	780	831.5
9	920	842.3
10	974	856.2

Fig. 6. Example of an N_i with a non-convex EEP function when $T_W/T_F = 800$. The graph shows EEP_i as nodes are added to F_i, and also the separate cost terms of the EEP calculation. The addition of the second forwarder to F_i increases EEP_i, causing a local minimum at $|F_i| = 1$, while $|F_i| = 8$ yields the global minimum and therefore the optimal forwarder set. The table shows each node's c_j and the value of EEP_i if all nodes up to j are included in F_i.

B. EDAD Operation

1) Framework: EDAD's overall framework is similar to that of other anycast data collection schemes. All nodes in the network may be data sources. One sink is assumed, though the concepts can be applied to a multiple-sink scenario. EEP is used as the routing metric. The sink has an EEP value of zero. All other nodes calculate their own EEP using a neighbor table, as shown in Fig. 2.

2) Anycast forwarding: In EDAD, all nodes maintain an explicit forwarder set, described in Section III-A4. A sender uses this set to choose the next hop receiver, which is the first $j \in F_i$ from which i hears a beacon. This dynamic choice of forwarder is what makes EDAD anycast and creates the DODAG topology shown in Fig. 1. Any beacon from a $j \notin F_i$ is ignored, even if the beacon advertises a low EEP. This is because the potential cost of sending to that node could outweigh the gain, either because the node does not provide enough progress or because the number of transmission attempts to transmit a packet to that node could be excessive. This is a tradeoff that EDAD allows EEP to balance via the forwarder set selection process.

3) Retries: EDAD maintains a separate retry counter for each $j \in F_i$. Sender i increments the corresponding counter whenever a data packet transmission to j fails, meaning that an acknowledgement from j is not received by i. If the counter reaches a maximum, then i stops retrying to j and waits for the next forwarder. Thus, EDAD retries a single data transmission to a particular node to try to utilize the available connection, but if the transmission is continually unsuccessful, EDAD takes advantage of its anycast nature and sends to a different forwarder instead of dropping the packet.

4) Maintenance: EEP calculation and forwarder set selection require that a node maintain an updated neighbor table, as shown in Fig. 2, to track its neighbors' EEPs and link quality estimates. To this end, all beacons and data packets contain the EEP of the transmitting node, allowing EDAD to handle new nodes and changes in EEP. When a new neighbor is discovered, or the EEP of a current neighbor changes, the neighbor table is updated and the forwarder set is rebuilt. This potentially results in a new EEP value, which is embedded in all outgoing beacons and data packets.

Because of its multipath nature, anycast inherently mitigates node failure issues. In the extreme case, if i hears no beacons from any $j \in F_i$, all $j \in F_i$ are assumed to be at least temporarily unavailable, so the EEP for each j is set to infinity in i's neighbor table. This will in turn increase i's EEP, so upstream nodes will avoid forwarding to i until i reestablishes communication with its forwarders.

Failed transmission attempts have a similar, but slower, effect on i's forwarder set. Link qualities with forwarders can be estimated as a moving average of successes per number of attempts. A failed transmission attempt to j thus lowers p_{ij}. If enough transmissions to j fail, p_{ij} will become small enough that j will be excluded from i's forwarder set.

IV. Evaluation

EDAD was implemented in the ns-2 network simulator with an underlying MAC protocol based on RI-MAC and extended to anycast. For comparison, single-parent ETX data collection (as in CTP), ORW, and ORiNoCo were also implemented. To directly compare EEP, EDAD's main contribution, to other metrics, these protocols were implemented on EDAD's framework. This means they all used the same MAC layer and retry scheme, but their own respective routing metrics. ORW was tested with $w = 0.1$ and $w = 1.0$, the extremes of the w range tested in the ORW papers, where $w = 0.1$ was found to be the best default configuration [15]. ORiNoCo was tested with $\Theta = 0.1$ and $\Theta = 0.9$, the extremes of the allowable range of Θ, according to [8]. ORW with $w = 0.1$ is denoted as ORW-0.1, and the other protocols are denoted similarly.

Nodes were distributed in a 250 x 250 m space in a grid plus variance pattern, as seen in Fig. 10. A fully random distribution was also tested, with similar results, so the grid plus variance pattern was used for clarity of examples. The sink was located in one corner and all other nodes generated packets according to a Poisson process with rate λ. Each link quality p_{ij} was calculated based on the distance between i and j using the shadowing propagation model, a fixed noise level of -97 dBm, and a well-known function for estimating

packet reception rate based on signal-to-noise ratio [19]. The shadowing propagation model used a reference path loss of 61.4 dB at 2 m and a path loss exponent of 1.97, values that were experimentally determined for a real WSN deployment in [20]. No random variation was used in the propagation model; instead, 30 different node layouts were tested for one simulated hour each and averaged together for each data point.

A. General Performance

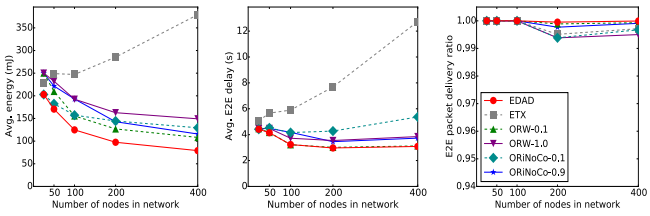
To test the general performance of EDAD relative to the other protocols, simulations were run at a variety of network densities, wakeup intervals, and data generation rates. The effect of each parameter is shown by fixing two of the three parameters at default values and varying the other. The default number of nodes in the network is $n = 100$, the default wakeup interval is $T_W = 2$ s, and the default data generation interval is $1/\lambda = 30$ s.

1) Network density: Performance under varying n, the number of nodes in the network, is shown in Fig. 7. Energy per packet for unicast ETX increases with density because, after a point, ETX's choice of forwarder remains similar, but the number of collisions increases as packets converge along similar routes to the sink, causing increased delay. However, for the anycast protocols, energy per packet decreases as n increases, because more nodes implies more forwarders, leading to lower delay and, due to the greater path diversity, fewer collisions. As expected given EEP's design goal, EDAD uses the least amount of energy, with a 25% decrease in energy per packet over ORW-0.1 when $n = 400$.

ORiNoCo performs unpredictably over the range of densities because of the way it uses the arbitrary Θ to determine its forwarder set. For example, at low densities, ORiNoCo-0.1 performs better than ORiNoCo-0.9, while the opposite is true at higher densities. ORW performs more consistently, with $w = 0.1$ generally providing better performance than $w = 1.0$, an observation that holds through most of the results and agrees with [15]. All protocols except EDAD show a small decrease in E2E reliability above 100 nodes, with protocols with smaller forwarder sets suffering more. This is likely because a smaller forwarder set means less path diversity, resulting in more collisions as routes from multiple nodes tend to converge on the way to the sink. At higher densities, the reliability of EDAD is expected to decrease as well.

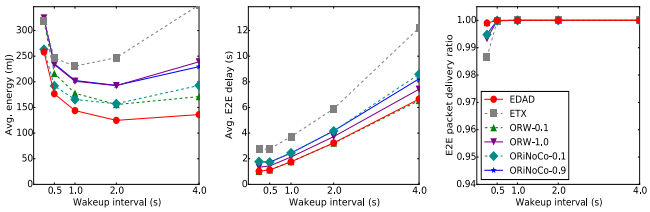
2) Wakeup interval: Fig. 8 shows average performance with different wakeup intervals. Most protocols reach an energy minimum at around $T_W = 2$ s. Below this, delay is shorter, implying less routing energy, but this decrease is outweighed by the energy of increased beaconing activity. EDAD again consumes the least energy per packet, around 20% less than ORW-0.1 at $T_W = 2$ s. EDAD also shows the slowest growth in energy per packet for $T_W > 2$ s.

These results also show how EDAD's automatic adjustment to T_W gives it an advantage over other protocols. At $T_W = 0.25$ s, ORW-0.1 shows energy performance similar to ORW-1.0, but as T_W increases, ORW-0.1 uses less and less energy than ORW-1.0. Also, at $T_W = 0.25$ s, ORiNoCo-0.1 shows energy performance similar to EDAD, but as T_W increases, the separation between ORiNoCo-0.1 and EDAD grows. This behavior is because a larger T_W leads to a greater delay, implying more energy consumption. A larger forwarder set counters that delay. But at a smaller T_W, one-hop delay matters less, so only forwarders that provide greater progress



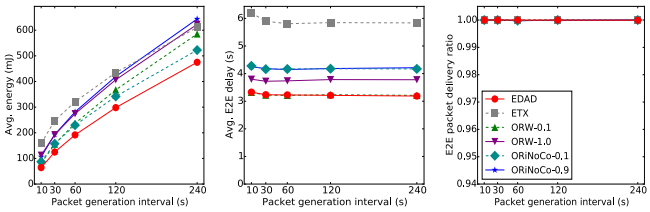
(a) Energy vs. n. (b) Delay vs. n. (c) Reliability vs. n.

Fig. 7. Performance with different network densities.



(a) Energy vs. T_W. (b) Delay vs. T_W. (c) Reliability vs. T_W.

Fig. 8. Performance with different wakeup intervals.



(a) Energy vs. $1/\lambda$. (b) Delay vs.$1/\lambda$. (c) Reliability vs. $1/\lambda$.

Fig. 9. Performance with different data generation rates.

should be used. Only EDAD balances this tradeoff in forwarder set size without parameter readjustment.

Above $T_W = 0.25$ s, all protocols show very high reliability. At $T_W = 0.25$ s, some packet loss occurs due to excessive collisions from beaconing activity.

3) Data generation rate: From Fig. 9, EDAD shows the least energy consumed per packet over the tested range of data generation intervals. In general, energy per packet increases as the data generation interval increases because when less packets are generated, relatively more of the network energy is consumed by beaconing activity. With data generation intervals above around 30 s, delay remains consistent. Below 30 s, collisions increase delay slightly. Reliability is high for all of the data generation intervals shown, but drops due to excessive collisions at higher data rates.

B. Detailed Behavior

1) Anycast route selection: Fig. 10 shows traces of two consecutive packets in a simulation using EDAD. At each hop, the packet can go to any one of a number of potential forwarders, resulting in widely different routes, even for back-to-back packets.

2) Path diversity: Fig. 11a shows a sample cumulative distribution function (CDF) for forwarder set sizes of four of the protocols with the default simulation settings. In this case, EDAD uses the most forwarders. Figs. 11b and 11c show traces of 100 packets in a particular topology for EDAD and ORW-0.1. EDAD shows a noticeably greater diversity in the paths taken by the packets, due to its larger forwarder set sizes. For this value of T_W, more forwarders leads to less energy

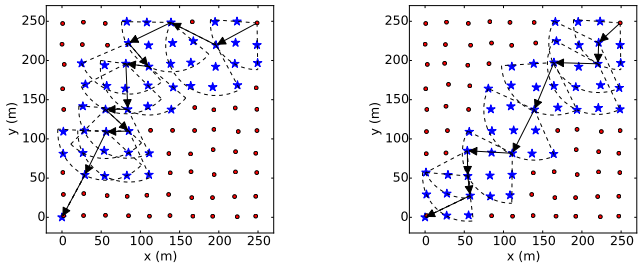
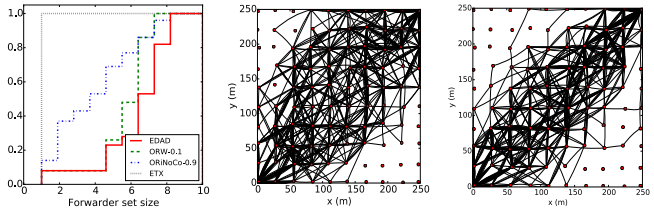


Fig. 10. Back-to-back traces of packets from a simulation using EDAD. The source is in the upper-right corner and the sink is in the lower-left corner. Arrows indicate the route taken by the packet. For each hop, the forwarder set of the node is shown enclosed in a dashed circle segment, and the potential forwarders are marked with blue stars.



(a) $|F|$ (b) EDAD (c) ORW-0.1

Fig. 11. Path diversity. (a) shows the CDF of the size of forwarder sets for a 100-node topology with default simulation settings. With these settings, EDAD selects the most forwarders. (b) and (c) show traces, for EDAD and ORW-0.1, of 100 packets sent diagonally across the simulated area. EDAD's path diversity spread is larger, leading to the energy savings seen for EDAD with these settings.

consumption, so EDAD yields the lowest energy per packet, as seen at $n = 100$ in Fig. 7. As previously discussed, at a smaller T_W, the lowest energy may be achieved with fewer forwarders. Since EEP automatically adapts itself to T_W, for this case, EDAD's forwarder set would be smaller.

3) Delay-transmissions tradeoff: Fig. 12 shows traces of two consecutive packets from a simulation using EDAD, in a format similar to Fig. 4, to illustrate a real example of the tradeoff between shorter E2E delay with more time spent in transmission and longer delay with less time spent in transmission. EEP estimates the shaded area in the figure, which allows the forwarder set selection process to balance the delay-transmissions tradeoff, and thus minimize energy, by minimizing the EEP for each node.

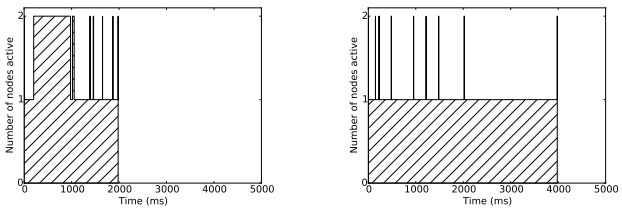


Fig. 12. The delay-transmissions tradeoff in real traces of consecutive packets routed from the same source node in a network using EDAD. The number of nodes along the path that are active is shown on the y-axis. Two nodes are active during each transmission. EEP estimates the shaded area, which allows the forwarder set selection process to balance the delay-transmissions tradeoff, and thus minimize energy, by minimizing the EEP for each node.

V. Conclusion

EDAD has been shown to achieve its design goal of further reduction of energy consumption in data-collection WSNs using anycast. This reduction comes primarily from EEP, a new anycast routing metric that judiciously selects the forwarder set that minimizes the expected energy consumed along the path of a packet. Future work includes implementation and testing of EDAD in a WSN testbed, including addressing implementation issues such as temporary routing loops that may arise from topology changes under dynamic channel conditions, and the possible increase in communication overhead per packet from the relatively large range of values that EEP can assume, compared to other metrics.

References

[1] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[2] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *ACM SenSys*, Nov. 2003.

[3] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *ACM SenSys*, Oct. 2006.

[4] D. Moss and P. Levis, "BoX-MACs: Exploiting physical and link layer boundaries in low-power networking," Stanford Computer Systems Laboratory, Tech. Rep., 2008.

[5] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *ACM SenSys*, Nov. 2008.

[6] G. Schaefer, F. Ingelrest, and M. Vetterli, "Potentials of Opportunistic Routing in Energy-Constrained Wireless Sensor Networks," in *EWSN*, Feb. 2009.

[7] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: opportunistic routing meets duty cycling," in *IEEE IPSN*, Apr. 2012.

[8] S. Unterschütz, C. Renner, and V. Turau, "Opportunistic, receiver-initiated data-collection protocol," in *EWSN*, Feb. 2012.

[9] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. The Collection Tree Protocol (CTP). [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html

[10] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "Minimizing Delay and Maximizing Lifetime for Wireless Sensor Networks With Anycast," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 515–528, 2009.

[11] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Multihop performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 337–348, 2003.

[12] S. Liu, K.-W. Fan, and P. Sinha, "CMAC: An energy-efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 4, pp. 1–34, Nov. 2009.

[13] H.-X. Tan and M. C. Chan, "A^2-MAC: An Adaptive, Anycast MAC Protocol for Wireless Sensor Networks," *IEEE WCNC*, Apr. 2010.

[14] Y. Gu and T. He, "Dynamic Switching-Based Data Forwarding for Low-Duty-Cycle Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 12, pp. 1741–1754, Dec. 2011.

[15] E. Ghadimi, O. Landsiedel, P. Soldati, S. Duquennoy, and M. Johansson, "Opportunistic Routing in Low Duty-Cycle Wireless Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 4, pp. 1–39, Jun. 2014.

[16] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree Bloom: scalable opportunistic routing with ORPL," in *ACM SenSys*, Nov. 2013.

[17] H.-J. Bungartz, S. Zimmer, M. Buchholz, and D. Pflüger, *Modeling and simulation: an application-oriented introduction*. Berlin: Springer, 2014.

[18] Texas Instruments, "CC2420 2.4 GHz IEEE 802.15.4/ZigBee-Ready RF Transceiver (Rev. C) Datasheet," 2007.

[19] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *ACM SenSys*, Nov. 2003.

[20] Y. Chen and A. Terzis, "On the implications of the log-normal path loss model: an efficient method to deploy and move sensor motes," in *ACM SenSys*, Nov. 2011.