Opportunistic Many-to-Many Multicasting in Duty-Cycled Wireless Sensor Networks

Mathew L. Wymore and Daji Qiao
Department of Electrical and Computer Engineering, Iowa State University, Ames, IA
{mlwymore, daji}@iastate.edu

***Abstract*—The technology of low-power wireless sensor networks (WSNs) needs to become more flexible to cater to emerging data-driven applications and the Internet of Things. For example, WSNs need to look beyond the traditional many-to-one data collection traffic model and begin to support multicast communications. However, efficient multicasting in WSNs is challenging. In this paper, we propose to apply the concept of opportunistic forwarding to create an opportunistic multicast framework for duty-cycled WSNs. Our framework allows for any node to directly and efficiently multicast to any subset of known potential destinations. We propose several variations of schemes to operate within this framework. We evaluate our framework and the proposed schemes using simulations.**

I. Introduction

With the rise of data-driven applications and the Internet of Things, the use-cases of wireless sensor network (WSN) technologies are becoming clearer and more abundant. However, to cater to these uses, WSNs need to become more flexible, including in terms of traffic pattern support. A traditional data-collection WSN supports an efficient many-to-one traffic pattern, where many *source* nodes send data to a single *destination* node, typically the root of a data collection tree. Some of these data-collection protocols also support data dissemination from the root back down the tree (one-to-many). Also, some of these protocols can be adapted to allow sources to send to any one of multiple potential destinations (many-to-any). However, WSNs lack efficient protocols that support multicasting with any node as the source and any set of nodes as the destinations (many-to-many).

One of the challenges with designing a multicast protocol for a WSN is that the wireless radios in a WSN are often *duty-cycled*, or turned on and off, in order to satisfy energy constraints for nodes. But for a node to receive a packet, the radios of the sender and receiver must both be on. A key challenge with WSNs, then, is to arrange a rendezvous between a sender and its receiver. The longer a sender waits for the receiver to turn its radio on, the more energy the sender consumes. If the sender must wait for multiple receivers, as in multicast, it will consume even more energy. Also, if a single packet must be sent to multiple forwarders to reach all destinations, each of these copies of the packet will consume additional precious energy as they are forwarded through the network. The combination of these considerations makes efficient multicasting in WSNs challenging.

However, we believe that efficient multicasting in WSNs will only become more desirable. Consider an application model based on data *publishers* and data *subscribers* [1]. Publishers are source nodes that periodically produce data from sensor readings. Subscribers are destination nodes that need this data to perform tasks. An example of this type of application could be an industrial monitoring and control network, where many control nodes need to be regularly updated on the conditions at other points in the network. Another example could be a smart building, where multiple actuators for lighting and HVAC may depend on readings from occupancy, temperature, and light sensors [2].

In recent years, the key metrics of energy, delay, and reliability of data collection have all been improved through the use of an *opportunistic* framework that allows nodes to dynamically make forwarding choices based on current network conditions and events. This gives the sender a large amount of flexibility in forwarding, allowing it to forward sooner and providing more options in case of failure. We propose to extend the concepts of opportunistic data collection to a full-fledged opportunistic multicast framework.

In our framework, detailed in Section IV, any node may directly and opportunistically forward a packet to any subset of potential destinations. Routes through the network are dynamically determined, along with decisions of when to send a copy of a packet to multiple forwarders (referred to hereafter as *splitting* the packet, for simplicity) so that it reaches all destinations efficiently. These processes are controlled by the *forwarder set selection* and *destination delegation* methods, also detailed in Section IV. We use simulations to evaluate our proposed framework in Section V. We find that our framework provides significant reductions in energy and delay compared to current multicasting methods for duty-cycled WSNs.

II. Related Work

Energy efficiency through duty-cycling has been a constant research topic for WSNs for well over a decade, as summarized in surveys such as [3]. At the link layer, protocols such as B-MAC [4], X-MAC [5], and ContikiMAC [6] have achieved progressively more efficient rendezvous without the need for synchronization or explicit scheduling between nodes, using a technique called *low-power listening*. Receiver-initiated protocols such as RI-MAC [7] use *low-power probing* to achieve similar results. At the network layer, protocols such as CTP [8] and IETF's RPL [9] natively support many-to-one data collection. More recently, *opportunistic* data collection for duty-cycled WSNs has been developed and tested in

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

protocols such as ORW [10] and our own EDAD [11]. These opportunistic data collection protocols use similar frameworks and differ mainly in the routing metric used for forwarding decisions, and also in the choice of either a sender- or receiver-initiated link-layer protocol. We note that opportunistic data collection has also been referred to as "anycast" data collection in the past, because it uses link-layer anycasting; however, to avoid confusion with our goal of network-layer multicasting, we use the term "opportunistic" here.

We classify approaches to multicasting in WSNs into four categories: optimal multicast trees (Steiner trees), flooding, down-tree routing, and multi-tree routing. Schemes that build (pseudo-)optimal multicast trees, such as [12]–[15], typically either do not consider duty cycling or rely on wakeup scheduling, and thus do not meet our energy efficiency and flexibility goals. In flooding schemes, such as IETF's MPL [16], each multicast packet is sent to every node in the domain. These schemes are inefficient at addressing smaller groups of nodes. A subset of protocols related to flooding, such as LWB [17] and Chaos [18], use the tightly synchronized transmissions of Glossy [19] to quickly disseminate packets to all participating nodes; however, these protocols are restrictive in terms of applications, scalability, and interoperability, making them unsuitable for our goal of flexible multicasting.

Down-tree routing is our name for a class of protocols that use an existing data collection tree structure [8] (or directed acyclic graph), rooted at the source, to support a one-to-many traffic pattern. To enable down-tree routing, each node stores the source addresses of the packets that it forwards up the tree in a *routing set*. The RPL standard defines a *storing mode* that builds these routing sets. Protocols such as SMRF [20] and BMRF [21] use RPL's storing mode to implement multicast. ORPL [22] implements an opportunistic version of down-tree routing that allows shortcuts across tree branches. REMI [23] builds clusters on top of the RPL tree and multicasts across clusters in addition to up and down the tree.

Even with these shortcuts, down-tree routing's reliance on the tree structure rooted at the source imposes limitations on its multicasting capabilities. In contrast, the *multi-tree routing* approach uses multiple tree structures, one rooted at each destination. These protocols then aggregate branches of the separate trees to form "multi-source, multi-sink" trees [2]. Examples of these protocols include MUSTER [2] and FROMS [24]. Our proposed multicasting framework combines this approach with opportunistic forwarding, producing a highly dynamic, flexible, and efficient multicasting process.

III. Preliminaries

Our opportunistic multicast framework is built on opportunistic data collection protocols such as ORW [10] and EDAD [11]. Our multicast framework could operate as either a sender-initiated or a receiver-initiated scheme; however, we prefer receiver-initiated schemes, in part because they mitigate the duplicate packet problem seen in opportunistic sender-initiated schemes [10]. Therefore, in this section, we briefly describe RI-MAC, a receiver-initiated duty-cycled MAC protocol, and EDAD, an opportunistic data collection scheme built on top of RI-MAC.

A. Receiver-initiated Asynchronously Duty-Cycled MAC

RI-MAC [7] is a receiver-initiated, asynchronously duty-cycled MAC protocol. In RI-MAC, nodes *sleep* (turn their radios off) when not engaged in communication. Periodically, nodes *wake* (turn their radios on) and advertise their presence as receivers by broadcasting a beacon packet and listening for a response. If no response arrives, the node goes back to sleep. The time between wakeups, or *wakeup interval* (T_W), is a network-wide parameter.

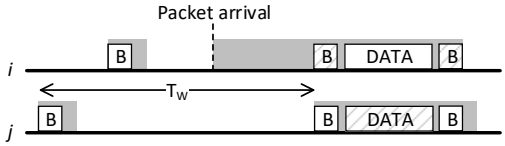


Fig. 1. Overview of RI-MAC, showing a timeline for sender i and receiver j. Both nodes broadcast beacon packets (B) at an interval T_W. Shaded areas indicate radio on-time, and hatched boxes indicate received packets.

As shown in Fig. 1, when a node has a packet to send, it wakes and *idly listens* for the next hop to wake and send a beacon. When this occurs, the sender unicasts the data packet in response to the beacon. The receiver responds with an acknowledgement beacon that also advertises the receiver's availability to receive additional packets. In case of a collision at the receiver, this beacon may also include a backoff window value to prevent competing senders from colliding again.

B. Opportunistic Data Collection

Traditionally, many-to-one data collection in WSNs is performed by building a tree with the destination (known as the *sink*) at the root [8]. This tree is built by defining a *routing metric* that models the "distance" from a node to the destination and then by choosing a forwarder that minimizes the routing metric's value. Traditional choices for routing metrics include hop count and the expected number of transmissions for a forwarded packet to reach the destination (ETX). For example, if p_{ij} is the probability of success of a TX attempt from i to j, then node i's ETX can be calculated as follows:

$$ETX_i = \min_{j \in \Gamma_i} \left(ETX_j + \frac{1}{p_{ij}} \right), \tag{1}$$

where Γ_i is the set of i's neighbors. As its forwarder, node i chooses the neighbor $j \in \Gamma_i$ that provides ETX_i. The destination's routing metric value is zero, and other nodes repeatedly calculate and propagate their own routing metric values as new information arrives. In a stable state, the network forms a routing metric *gradient*, and packets flow along the gradient to the destination.

In this single-forwarder framework, a sender must wait for its preferred forwarder to wake, which is expensive in terms of both delay and energy. This *forwarding delay* can be significantly reduced by selecting a *forwarder set*, composed

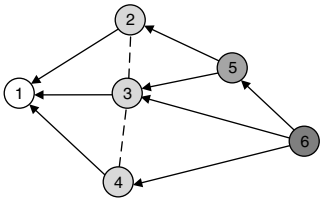


Fig. 2. Single-sink opportunistic data collection with a routing metric gradient to node 1.

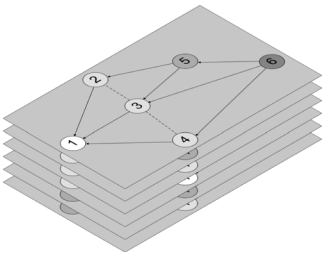


Fig. 3. Opportunistic multicast framework based on gradients to each possible destination.

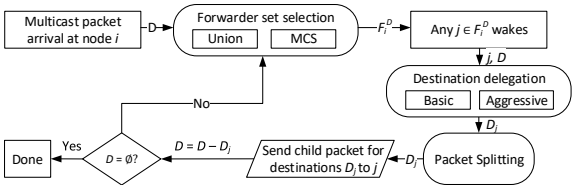


Fig. 4. Overview of our multicast framework. In each iteration, a child packet is sent to the first forwarder in the forwarder set (selected via either the Union or the MCS method) to wake. The packet is addressed to some subset (selected via either the Basic or Aggressive method) of the destinations. The process repeats until all destinations have been delegated.

of nodes that each provide "sufficient" progress toward the sink, instead of a single forwarder [10]. The sender can then opportunistically send to the first forwarder in the forwarder set that wakes, reducing forwarding delay and increasing robustness to link fluctuations and topology changes. This type of opportunistic data collection framework transforms the data collection tree of the single-forwarder framework into a destination-oriented, directed, acyclic graph (DODAG). An example of such a DODAG is shown in Fig. 2.

The challenge lies in selecting the forwarder set. In our previous work, we proposed a new opportunistic routing metric called EEP that models the expected energy consumed along the path of a packet. In short, the EEP for a node i (EEP_i) is the sum of the expected energy consumed while waiting for any forwarder in i's forwarder set F_i to wake, the expected energy consumed to transmit the packet to the forwarder, and the expected energy to send the packet from that forwarder to the destination. EEP_i thus depends on the number of forwarders in F_i, the EEP of those forwarders, and the quality of the links with those forwarders. The set F_i is selected from the set of i's neighbors such that EEP_i is minimized. We will use EEP in our evaluations in Section V; for more details, please refer to [11].

IV. Opportunistic Multicast Framework

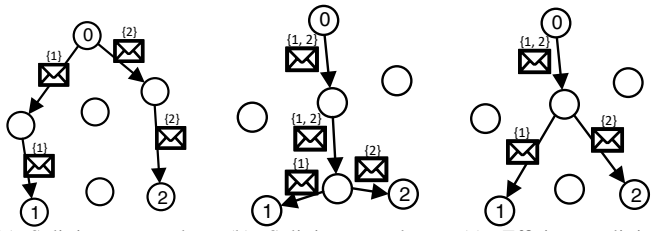
Our opportunistic multicast framework builds on the opportunistic data collection framework described in the previous section. Similar to [24], each node stores a routing metric value for each potential destination, creating a full-fledged *gradient stack*, shown in Fig. 3. The forwarding process for a multicast packet in our framework is shown in Fig. 4. The first step is *forwarder set selection*, which determines the next hops for the sender to consider. Once a forwarder in the forwarder set has awoken, the sender chooses which destinations will be handled by that forwarder, called *destination delegation*. A copy of the packet addressed to those destinations is sent to the forwarder, and the forwarding process repeats until all destinations have been delegated. The structure of the framework and the steps in the forwarding process are described in more detail in the following sections.

A. Gradient Stack

In our opportunistic multicast framework, each node stores a routing metric value for each potential destination, resulting in a routing metric gradient for each potential destination. We refer to these gradients as the *gradient stack*, conceptualized in Fig. 3. The gradient stack allows for highly dynamic, opportunistic multicasting, because each node has local information about how to reach each destination individually. This information can be dynamically merged, depending on the relevant destinations, to make multicasting decisions. The process of creating these gradients is the same as that of a traditional many-to-one gradient, with each potential destination acting as the root of a data collection DODAG.

B. Destination Grouping and Splitting

When a source generates a multicast packet, all of the multicast destinations are *grouped* into it. As shown in Fig. 5, as the packet is forwarded, it is *split* into child packets, each with a disjoint subset of the destinations, in order to reach all destinations. The choice of when to group and when to split destinations affects the efficiency of packet delivery and is a key consideration of our multicast framework. Splitting too early, as in Fig. 5a, creates excess copies of the packet that need to be separately forwarded through the network. Splitting too late, as in Fig. 5b, can result in unnecessary hops. In our framework, the decisions for when to group and when to split are made by the forwarding process, described below.



(a) Splitting too early creates excess copies of the packet.

(b) Splitting too late creates unnecessary hops.

(c) Efficient splitting results in less network activity.

Fig. 5. Example of how splitting behavior affects efficiency of packet delivery. In this example, node 0 has a multicast packet to deliver to nodes 1 and 2.

C. Forwarder Set Selection

As shown in Fig. 4, the first step of the forwarding process is forwarder set selection: given a multicast packet addressed to a set of destinations D, the sender must decide which neighbors to consider as forwarders. This set of neighbors is called the *forwarder set* and, for a node i, is denoted ${F_i}^D$. Each forwarder in the forwarder set provides enough

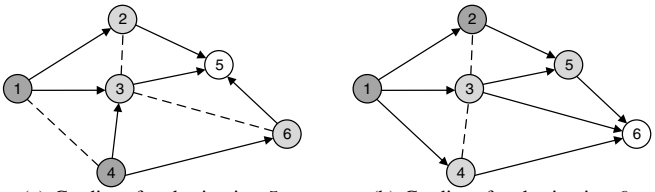
"progress" to warrant sending to that forwarder. If a forwarder provides progress toward a particular destination d, we say that forwarder *covers* d. For a multicast packet, multiple forwarders may be required to cover all $d \in D$, and at least one node in the forwarder set must cover each destination.

Defining "progress" in a multicast sense is difficult. We first considered extending the design of the single-sink gradient to a multicast scenario by defining a meta-metric that combines the metrics for each $d \in D$ to produce one multicast gradient value. However, this approach would require knowledge of the meta-metric for each neighbor, and since the meta-metric would be different for each possible D, this approach is impractical. We therefore propose two simpler, heuristic-based methods of choosing forwarders. These methods, Union and MCS, use only the information found in the gradient stack.

1) ***Union:*** The forwarder set ${F_i}^D$ for a node i, given a set of destinations D, is the union of the forwarder sets for each individual destination $d \in D$. The heuristic used here is that if a node is a good enough forwarder for at least one destination, then it is worth sending to that node.

2) ***MCS:*** The forwarder set is the union of all minimum covering sets (MCSs), where a minimum covering set is defined as a set with minimal members for which each $d \in D$ is covered by at least one member. This is a local attempt to minimize the number of packet splits. The union is taken to allow the wakeup order of the forwarders to determine the actual minimum covering set used.

As an example, we use the gradients shown in Fig. 6 and consider a multicast packet with $D = \{5, 6\}$. Tables I and II show the forwarder sets created using Union and MCS, respectively. In this example, using Union, Node 1's forwarder set for destination 5 is $\{2, 3\}$, and for destination 6, $\{2, 3, 4\}$. The forwarder set $F_1^{\{5,6\}}$ is the union of these two sets, $\{2, 3, 4\}$. Using MCS, node 1's forwarder sets for destinations 5 and 6 both contain node 2, so $\{2\}$ is an MCS for node 1, because no sets exist that can cover both destinations with fewer nodes. Likewise, $\{3\}$ is also an MCS. The forwarder set $F_1^{\{5,6\}}$ is the union of these MCSs, $\{2, 3\}$.



(a) Gradient for destination 5. (b) Gradient for destination 6.

Fig. 6. Example network with gradients shown for destinations 5 and 6.

Union naturally creates larger forwarder sets than MCS. This gives Union an advantage in sparser networks, because in this case, adding even one more forwarder can significantly reduce the forwarding delay. In dense networks with many potential forwarders, MCS is penalized less for being more selective. Additionally, MCS's selectivity decreases the number of splits and the amount of traffic in the network. This analysis is supported by our simulations in Section V.

TABLE I
FORWARDER SETS SELECTED FOR $D = \{5, 6\}$ USING UNION.

i	$F_i^{\{5\}}$	$F_i^{\{6\}}$	$F_i^{\{5,6\}}$
1	$\{2, 3\}$	$\{2, 3, 4\}$	$\{2, 3, 4\}$
2	$\{5\}$	$\{5\}$	$\{5\}$
3	$\{5\}$	$\{5, 6\}$	$\{5, 6\}$
4	$\{3, 6\}$	$\{6\}$	$\{3, 6\}$
5	$\{\}$	$\{6\}$	$\{6\}$
6	$\{5\}$	$\{\}$	$\{5\}$

TABLE II
FORWARDER SETS SELECTED FOR $D = \{5, 6\}$ USING MCS.

i	MCSs	$F_i^{\{5,6\}}$
1	$\{2\}$, $\{3\}$	$\{2, 3\}$
2	$\{5\}$	$\{5\}$
3	$\{5\}$	$\{5\}$
4	$\{6\}$	$\{6\}$
5	$\{6\}$	$\{6\}$
6	$\{5\}$	$\{5\}$

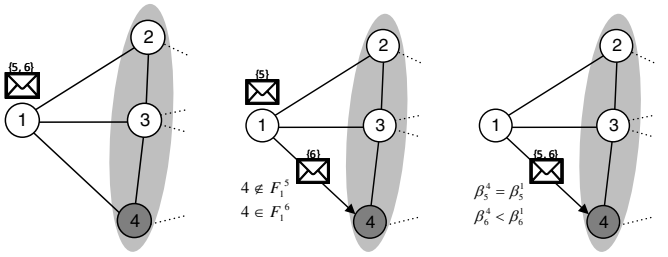
D. Destination Delegation

In our opportunistic framework, the sender i sends a packet to the first forwarder $j \in {F_i}^D$ that wakes. As shown in Fig. 4, the next step is *destination delegation*, the process of choosing which destinations to group into the packet to j. We propose the following two methods:

1) ***Basic:*** All destinations for which the forwarder provides *sufficient* progress, as determined by the gradient stack, are delegated. Formally, assume sender i has a multicast packet for a set of destinations D, and a forwarder $j \in {F_i}^D$ is available. Then D_j, the set of destinations delegated to j, is the set of all destinations d for which $j \in F_i^{\{d\}}$.

2) ***Aggressive:*** All destinations for which the forwarder provides *any* progress are delegated. Formally, if β_i^d is the routing metric value for destination d at node i, then D_j is the set of all destinations d for which $\beta_j^d \leq \beta_i^d$.

An example of the two delegation methods, using the topology of Fig. 6, is shown in Fig. 7. In this example, a packet for 5 and 6 has arrived at node 1. The forwarder set (using Union) is $F_1^{\{5,6\}} = \{2, 3, 4\}$, and node 4 wakes first. Using Basic delegation, the packet is split and only 6 is delegated to node 4. Node 1 must then start a second iteration of the cycle shown in Fig. 4, with $D = \{5\}$. But using Aggressive delegation, both destinations are delegated to node 4.



(a) Node 4 wakes first, and node 1 must determine D_4, the set of destinations to delegate to node 4.

(b) Basic delegation: $D_4 = \{6\}$ because node 4 is in 1's forwarder set for 6, but not for 5.

(c) Aggressive delegation: $D_4 = \{5, 6\}$ because 4's routing metric to both 5 and 6 is no larger than 1's.

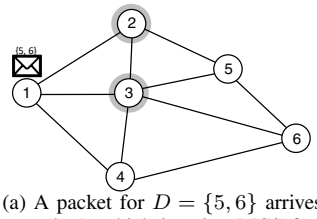
Fig. 7. Example of delegation mechanisms. Node 1 has a packet with $D = \{5, 6\}$. The shaded area indicates $F_i^{\{5,6\}}$, selected using Union.

From this example, we can see the motivation for Aggressive delegation. Node 4 does not provide significant progress toward node 5 from node 1, but there is a chance that one of node 4's forwarders does provide good progress to both 5 and 6. If such a forwarder wakes next, then Aggressive delegation has both allowed node 1 to finish sending earlier and prevented an excess packet split from occurring. If not, the drawback is

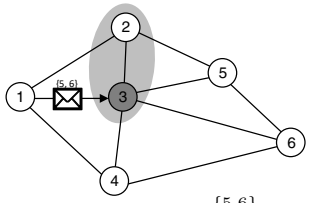
minimal, as the packet from 1 to 4 was being sent regardless, and the hop from node 1 to node 4 provides non-negative routing progress toward destination 5. Our simulation results have shown that Aggressive delegation is generally effective.

E. Summary and Example

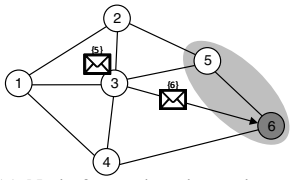
Returning to Fig. 4, after destination delegation, a child packet bound for D_j is sent to node j. D_j is then removed from D, and if D is now empty, node i is finished sending the packet. Otherwise, node i repeats the cycle, starting with forwarder set selection, using the updated D. We conclude the description of our opportunistic multicast framework with a small example, shown in Fig. 8. The gradient stack for this example is the same as Fig. 6.



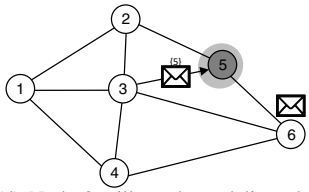
(a) A packet for $D = \{5, 6\}$ arrives at node 1, which is using MCS forwarder set selection. Both $\{2\}$ and $\{3\}$ are minimum covering sets.



(b) The forwarder set $F_1^{\{5,6\}}$ is the union of these sets. Node 3 wakes first and provides progress to 5 and 6, so node 1 delegates 5 and 6 to 3.



(c) Node 3 now has the packet with $D = \{5, 6\}$. The minimum covering set is $\{5, 6\}$. Node 6 wakes first, so node 3 splits the packet and sends to node 6 with $D_6 = \{6\}$.



(d) Node 3 still needs to deliver the packet to node 5, so it waits until node 5 wakes and then sends to node 5 with $D_5 = \{5\}$. All destinations have been reached.

Fig. 8. Example of delivery of a packet using MCS forwarder set selection and Basic delegation.

V. Evaluation

We evaluated our multicast scheme with an event-based simulator written in Python. The simulator uses a distance-based path loss model to determine received signal strength, which is combined with a noise floor to obtain a signal-to-noise ratio (SNR). We then use an empirically-derived function from TOSSIM [25] to obtain packet reception ratio (PRR) from the SNR. The simulated link layer is an opportunistic version of RI-MAC [7]. We simulate basic channel contention and packet retransmissions. The default simulation settings are shown in Table III. The frame duration settings were selected assuming a 250 kbps radio, as in IEEE 802.15.4, and packets of less than 100 bytes.

In each simulation topology, one source node is placed at each corner of the rectangular deployment, with a fifth source placed in the center. The other nodes, including the destinations, are randomly deployed in a connected topology. For each topology, each source node generates a total of 100 packets at a specified interval, with initial packets scheduled randomly.

TABLE III
Simulation default settings.

Parameter	Default
Path loss exponent	3.0
Noise floor	-100 dBm
Deployment area	100x100 m
# nodes (n)	100
# sources (s)	5
# destinations (d)	10
Data interval (T_D)	60 s
Wakeup interval (T_W)	500 ms
Wakeup duration (T_B)	1 ms
Frame duration (T_F)	3 ms

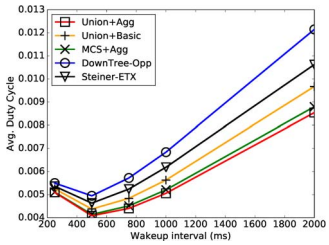


Fig. 9. Average node duty cycle vs. wakeup interval.

Each packet is addressed to the same set of destinations. This mimics a scenario in which the destinations are "subscribed" to data being generated by a group of sources. The results below are averaged over at least 50 topologies. All schemes use the same duty-cycled link layer. All tests are performed with the network at steady state.

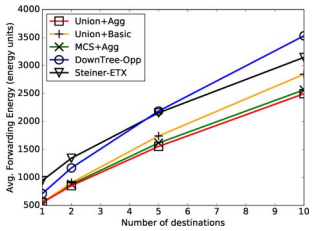
We evaluate three combinations of our forwarder set selection methods and destination delegation methods: Union with Aggressive delegation (Union+Agg), Union with Basic delegation (Union+Basic), and MCS with Aggressive delegation (MCS+Agg). We use the EEP routing metric [11] for these protocols. We do not include results for MCS with Basic delegation, which performs worse than MCS+Agg.

We implemented a down-tree routing scheme, which we call DownTree-Opp, to compare to our framework. DownTree-Opp uses an opportunistic framework, similar to ORPL [22] but with EEP as the routing metric. We use EEP to focus the comparison on frameworks and not routing metrics; however, we observed nearly identical performance when using EDC, which is ORPL's routing metric. In our simulations, each source is the root of a data collection DODAG, a best-case scenario for DownTree-Opp.

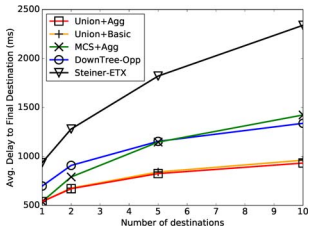
Finally, we implemented a scheme we call Steiner, which represents an optimal multicast solution for a single-forwarder framework, such as that employed by FROMS [24]. This omniscient, centralized scheme exhaustively searches for the minimum-cost subgraph that connects each source with all destinations (the minimum Steiner tree on the graph). Packets are then routed on this tree. We use ETX as the cost metric.

A. Wakeup Interval Calibration

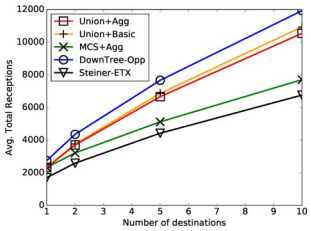
The energy performance strongly depends on the wakeup interval T_W and its relation to the traffic rate. Therefore, we first calibrate the network wakeup interval by testing a variety of wakeup intervals to find the optimal wakeup interval for our traffic settings. The average network duty cycle versus T_W is shown in Fig. 9. The duty cycle is calculated as the radio on-time divided by the total simulation time. If T_W is too small, excessive on-time is spent on wakeups. If T_W is too large, excessive on-time is spent on idle listening. From this figure, we see that the best T_W for our traffic load is around 500 ms, which we use for the remainder of our evaluations; however, we observed that trends in performance remain similar, regardless of wakeup interval optimality.



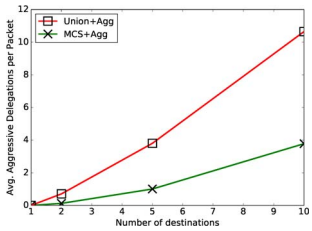
(a) Forwarding energy per packet.



(b) Delay.

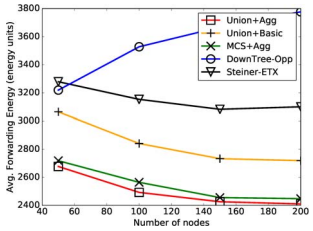


(c) Total RXs.

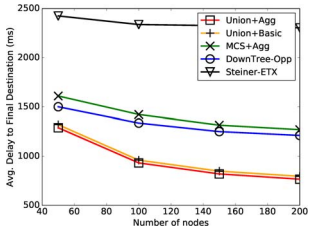


(d) Amount of Aggressive delegation.

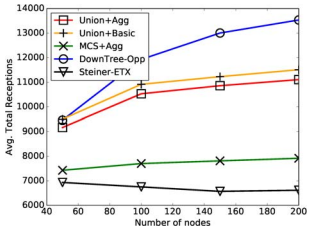
Fig. 10. Evaluation for different numbers of destinations.



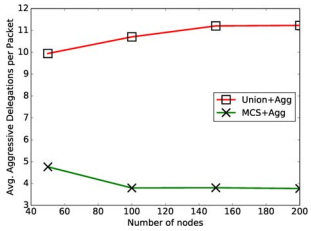
(a) Forwarding energy per packet.



(b) Delay.



(c) Total RXs.



(d) Amount of Aggressive delegation.

Fig. 11. Evaluation for different numbers of nodes in a fixed area.

B. *Effect of the Number of Destinations*

Fig. 10 shows the effect of the number of (randomly-deployed) destinations to which each source sends each packet. Fig. 10a shows the *forwarding energy*, which we define as the energy spent on idle listening, TXs, and RXs due to packet forwarding. This quantity does not include the baseline energy load of periodic wakeups. In short, forwarding energy represents the additional energy load imposed on the system from use of the multicast scheme. We measure forwarding energy in *energy units*, defined as the amount of energy consumed in one millisecond of radio on-time.

Forwarding energy per packet increases with the number of destinations for each packet, which is expected—more destinations means more work. Aggressive delegation helps energy performance for Union (and also MCS). Union+Agg uses the least energy for the numbers of destinations shown here. Steiner-ETX uniformly uses more energy more than Union+Agg, meaning that the benefits of Union+Agg's opportunistic framework outweigh the drawbacks of using non-optimal forwarders. Steiner-ETX does perform better than DownTree-Opp with five or more multicast destinations.

Fig. 10b shows delay, which we measure as the time required to deliver a packet from the source to all of the multicast destinations. Delay increases with the number of destinations, which makes sense, because more destinations increases the chance that at least one of them will be far from each source. The Union forwarder set selection method provides the best delay performance, because Union chooses more forwarders, sending to the first forwarder good enough for any one destination, whereas MCS waits for a forwarder that is good for some number of destinations. The Aggressive delegation mechanism provides a small delay performance gain. DownTree-Opp's delay is competitive with MCS+Agg, but not Union. Steiner-ETX has the worst delay performance, because its adherence to an optimal tree structure means that senders must always wait for a particular forwarder to wake.

Comparing the delay and energy figures, the energy performance of Union and MCS appears surprisingly close. Fig. 10c provides a clue as to how MCS compensates for its lackluster delay, showing the average total number of successful RXs over the course of the simulations, which is directly related to the number of times each packet is split on the way to its destinations. MCS splits packets considerably less often than Union or DownTree-Opp. We therefore note that MCS may perform better than Union in a high-traffic scenario where congestion is a factor. Steiner-ETX splits even less than MCS, due to its optimal tree structure. The tradeoff for our proposed opportunistic framework's superior energy and delay performance, then, is higher amounts of network traffic.

Fig. 10d shows the number of destinations "aggressively" delegated, meaning those destinations assigned to a forwarder by Aggressive delegation that would not be assigned by Basic delegation. We see that Union makes more use of Aggressive delegation than MCS, which is one way Union compensates for its use of lower-quality forwarders.

C. *Effect of the Network Density*

Fig. 11 shows results for performance with different numbers of nodes in the fixed simulation area, measuring the effect of network density. Fig. 11a shows forwarding energy. Our proposed schemes again have better energy performance than DownTree-Opp and Steiner-ETX. For our schemes, forwarding energy decreases with the number of nodes due to the increased options for forwarder sets. Delay (Fig. 11b) also decreases, for the same reason.

For our randomly-deployed networks, fewer nodes often leads to irregular topologies. The effects of this irregularity can be seen in the figures, with a "regularity threshold" emerging around 100 nodes. For example, in Fig. 11c, the number of splits stays fairly steady above 100 nodes. Below this point, route options tend to be limited, forcing destinations to be grouped together for longer. Interestingly, this forced grouping actually helps the DownTree schemes in energy performance

at 50 nodes. When more routes open up at 100 nodes, the DownTree schemes split more often. Splits in a DownTree scheme tend to be more costly because a packet from a source to a destination in a DownTree protocol must follow a route conforming to the existing data collection tree or DODAG structure that is rooted at the source. The splits therefore tend to happen earlier on the path of a packet, where more of the tree branches overlap, which means each child packet must be forwarded farther after the split.

D. Performance Summary

For our testing scenario, Union+Agg is the most energy-efficient variation of our framework. Aggressive delegation uniformly outperforms Basic delegation. Union forwarder set selection produces less delay than MCS forwarder set selection, but MCS produces less network activity. Our framework outperforms a representative state-of-the-art protocol, DownTree-Opp (which is operating in a best-case scenario here), showing the advantage of the gradient stack approach. Finally, our opportunistic framework outperforms Steiner-ETX, a single-forwarder solution operating on an optimal tree. We conclude that our dynamic and flexible approach is more suited to duty-cycled WSNs than rigid, "optimal" approaches.

E. Traces

To illustrate the dynamic nature of our multicast framework, we present two traces of Union+Agg in Fig. 12. All parameters, including routing metric values, are the same for these two traces, but the paths taken by the packets are different. This random-like behavior makes the framework robust, and also helps to balance the energy load between the nodes.

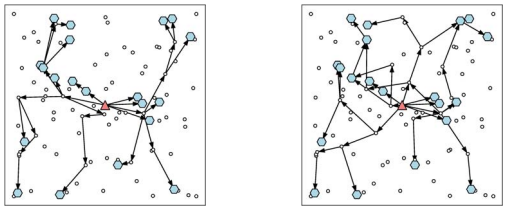


Fig. 12. Two traces of the paths taken to deliver a single multicast packet with Union+Agg. The red triangle in the center is the source, and the blue hexagons are destinations. Arrows show packet transmissions.

VI. Conclusion

We have proposed an opportunistic multicast framework for duty-cycled wireless sensor networks that provides flexible, efficient, and reliable multicasting. Our framework can be used to provide low-energy, low-delay multicasting for a variety of applications. Future work includes in-depth analysis of the overhead of our framework, methods for reducing memory consumption and network maintenance costs, and implementation of our framework in Contiki OS.

Acknowledgements

Funded in part by U.S. National Science Foundation Grants No. 1730275 and No. 1069283, the Integrative Graduate Education and Research Traineeship in Wind Energy Science, Engineering and Policy at Iowa State University.

References

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[2] L. Mottola and G. P. Picco, "MUSTER: adaptive energy-aware multisink routing in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 12, pp. 1694–1709, Dec 2011.

[3] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of mac protocols in wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 101–120, 2013.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. ACM SenSys*, 2004.

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

[6] A. Dunkels, "The ContikiMAC radio duty cycling protocol," SICS, Tech. Rep. 5128, Jan. 2012.

[7] 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 *Proc. ACM SenSys*, 2008.

[8] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. ACM SenSys*, 2009.

[9] A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, T. H. Clausen, and T. Winter, "RPL: IPv6 routing protocol for low-power and lossy networks," RFC 6550, Mar. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6550.txt

[10] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *Proc. ACM IPSN*, 2012.

[11] M. L. Wymore, Y. Peng, X. Zhang, and D. Qiao, "EDAD: energy-centric data collection with anycast in duty-cycled wireless sensor networks," in *Proc. IEEE WCNC*, 2015.

[12] K. Han, Y. Liu, and J. Luo, "Duty-cycle-aware minimum-energy multicasting in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 910–923, Jun. 2013.

[13] K. Han, J. Luo, L. Xiang, M. Xiao, and L. Huang, "Achieving energy efficiency and reliability for data dissemination in duty-cycled WSNs," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1041–1052, Aug. 2015.

[14] H. Gong, L. Fu, X. Fu, L. Zhao, K. Wang, and X. Wang, "Distributed multicast tree construction in wireless sensor networks," *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 280–296, Jan 2017.

[15] Q. Chen, H. Gao, S. Cheng, X. Fang, Z. Cai, and J. Li, "Centralized and distributed delay-bounded scheduling algorithms for multicast in duty-cycled wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3573–3586, Dec 2017.

[16] R. Kelsey and J. Hui, "Multicast protocol for low-power and lossy networks (MPL)," RFC 7731, Feb. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7731.txt

[17] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proc. ACM SenSys*, 2012.

[18] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale," in *Proc. ACM SenSys*, 2013.

[19] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. ACM/IEEE IPSN*, 2011.

[20] G. Oikonomou, I. Phillips, and T. Tryfonas, "IPv6 multicast forwarding in RPL-based wireless sensor networks," *Wireless Personal Communications*, vol. 73, no. 3, pp. 1089–1116, 2013.

[21] G. G. Lorente, B. Lemmens, M. Carlier, A. Braeken, and K. Steenhaut, "BMRF: bidirectional multicast RPL forwarding," *Ad Hoc Networks*, vol. 54, pp. 69–84, 2017.

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

[23] M. Conti, P. Kaliyar, and C. Lal, "REMI: a reliable and secure multicast routing protocol for IoT networks," in *Proc. ACM ARES*, 2017, pp. 1–8.

[24] A. Förster and A. L. Murphy, "FROMS: a failure tolerant and mobility enabled multicast routing paradigm with reinforcement learning for WSNs," *Ad Hoc Networks*, vol. 9, no. 5, pp. 940 – 965, 2011.

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