Staffetta: Smart Duty-Cycling for Opportunistic Data Collection

Marco Cattani† Andreas Loukas† Marco Zimmerling‡ Marco Zuniga† Koen Langendoen†
†Embedded Software Group, Delft University of Technology, The Netherlands
‡Networked Embedded Systems Group, TU Dresden, Germany
{m.cattani, a.loukas, m.zuniga, k.g.langendoen}@tudelft.nl, marco.zimmerling@tu-dresden.de

ABSTRACT

Opportunistic routing protocols tackle the problem of efficient data collection in *dynamic* wireless sensor networks, where the radio is duty-cycled to save energy and the topology changes unpredictably due to node mobility and/or link dynamics. Unlike protocols that maintain a routing structure, in opportunistic protocols nodes forward packets to *any* neighbor that wakes up first, reducing latency and energy costs and increasing the resilience to network dynamics.

We claim the performance of existing opportunistic routing protocols can be improved while retaining their resilience by harnessing the synergy between duty cycling and opportunistic forwarding. To prove this claim, we present *Staffetta*, the first practical duty-cycle adaptation scheme for opportunistic low-power wireless protocols. Staffetta dynamically adapts each node's wake-up frequency to its current forwarding cost, so nodes closer to the sink become more active than nodes farther away. In this way, Staffetta biases the forwarding choices toward the sink as the neighbor waking up first is also likely to offer high routing progress. Experiments on two testbeds with four different opportunistic routing mechanisms demonstrate that Staffetta achieves severalfold performance improvements compared with a fixed wake-up frequency. As a case a point, Staffetta enables ORW, the state-of-the-art opportunistic routing protocol, to reduce end-to-end packet latency by 79–452× and energy consumption by 2.75–9× while increasing packet delivery ratio compared with ORW's default link-layer settings.

CCS Concepts

•**Networks** → **Network protocol design;** *Transport protocols;*

Keywords

Duty Cycling; Data Collection; Opportunistic Routing

1. INTRODUCTION

Data collection is a solved problem—that is, from the packet-reception perspective, with protocols reported to collect more than 99.9 % of all data. However, data collection is still a challenge from the real-world perspective where latency and network lifetime are major concerns. In that light, ORW is the first protocol to effectively employ opportunistic routing in wireless sensor networks [21]. ORW shows that compared to traditional routing schemes, such as CTP [16], opportunistic forwarding decisions can significantly improve data collection performance in asynchronously duty-cycled networks in terms of end-to-end packet latency and energy efficiency, while being more resilient to topology changes.

The idea of opportunistic routing is simple: instead of first making the forwarding decision and then waiting for the destination to wake up, nodes forward packets opportunistically to the neighbor that wakes up first and offers enough routing progress toward the sink. The more potential forwarders a node has, the shorter the time it needs to wait on average before sending, and hence the more efficient it can operate [21]. Moreover, the more forwarding choices a node has, the more resilient it is to link fluctuations and other network dynamics.

Problem. Despite the many benefits over traditional routing schemes, existing opportunistic routing protocols run on top of duty-cycled link layers wherein all nodes have *the same* wake-up frequency. As a result, all nodes have the same forwarding costs with regard to latency and energy consumption. But nodes play *different* roles. The closer a node is to the sink, the more packets it has to forward, and hence, the lower its forwarding costs should be. The wake-up frequencies should match the role of each node. This is indeed the idea of various adaptive, duty-cycled link layers. However, those targeting opportunistic data collection protocols have only been studied analytically [18, 19], thus lacking validation against the real-world dynamics of low-power wireless, or have been designed for static networks and traditional tree-based routing schemes on top of the unicast primitive [17, 25, 32].

Contribution. We close this gap by presenting *Staffetta*, the first practical duty-cycle adaptation scheme for opportunistic low-power wireless protocols. Staffetta adapts the wake-up frequency of each individual node to its current forwarding costs. As a result, nodes closer to the sink are more active (wake up more often) than nodes at the edge of the network. This *activity gradient* automatically steers packets in the right direction: the probability of finding an awake neighbor is highest in the direction of the sink. This way, Staffetta improves the performance of opportunistic protocols while retaining their resilience to network dynamics.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Sensys 2016 November 14–16, 2016, Stanford, CA, USA

© 2016 ACM. ISBN 978-1-4503-4263-6…$15.00

DOI: xxxxxx

A key advantage of our approach is that it can be easily incorporated into existing duty-cycled link layers while incurring a minimal overhead. As such, Staffetta executes seamlessly underneath existing opportunistic protocols such as ORW. Moreover, due to its fully distributed and localized operation, Staffetta scales well to large and dense networks, and readily supports scenarios with mobile nodes.

We implement Staffetta atop the X-MAC [1] implementation in Contiki. In addition, we further implement four different opportunistic routing schemes, two of which are directly inspired by state-of-the-art protocols: ORW [21] and the Backpressure Collection Protocol (BCP) [26]. This allows us to quantify and compare the performance gains due to Staffetta across different opportunistic schemes irrespective of varying operating systems, protocol stacks, and other implementation details of the original protocols. Nevertheless, we also perform a head-to-head comparison against the original TinyOS-based implementation of ORW.

To this end, we run experiments on the FlockLab [23] and Indriya [8] testbeds with 28–139 TelosB nodes and different network diameters and node densities. For example, we find that compared to using a fixed wake-up frequency, Staffetta achieves superior performance in all metrics and across all opportunistic schemes and scenarios we consider. A case in point is the significant improvement Staffetta achieves compared to the original ORW with default link layer: 79–452× shorter median end-to-end packet latencies, 2.75–9× lower median energy consumption, and higher packet delivery ratios. Moreover, experiments with a mobile sink demonstrate that Staffetta further improves the high resilience of opportunistic protocols, delivering packets more reliably and consistently two orders of magnitude faster in the face of drastic network topology changes. These performance gains come with a negligible overhead in terms of memory and processing. From a broader perspective, we show that Staffetta's activity gradient can also be used as an implicit routing metric by itself, achieving performance similar or better than other mechanisms empowered by Staffetta, but without the need for maintaining up-to-date routing state for each neighbor.

In summary, this paper contributes the following:

- We present Staffetta, the first practical duty-cycle adaptation scheme for opportunistic wireless protocols. By analyzing the interplay between duty cycling and opportunistic forwarding, we explain how Staffetta can reduce latency while extending network lifetime.
- We demonstrate how to efficiently implement and combine Staffetta with existing opportunistic protocols.
- We perform extensive experiments and systematic comparisons on two testbeds, and show that Staffetta significantly improves the performance of opportunistic routing protocols while further helping their resilience.

2. PROBLEM STATEMENT

In duty-cycled networks, two main metrics determine the efficiency of data collection. First, the time until a packet is collected at the sink, known as *packet latency*. Second, the *routing overhead*, which accounts for the resources (e.g., energy, time) consumed by each node to forward messages. The goal of many data collection protocols (and their routing metric) is to *minimize* the packet latency Δ with the least possible amount of additional resources. More formally, consider a collection protocol and denote by $P = \{u_0, u_1, \ldots, u_h\}$ the sequence of nodes forming the path of length h (in hops) from node u_0 to the sink u_h. In an asynchronous duty-cycled network, the packet latency over path P is given by

$$\Delta_P = \sum_{i=0}^{h-1} \delta_F(i), \tag{1}$$

where the *forwarding delay* $\delta_F(i)$ is the time node u_i needs to send a packet to u_{i+1} (refer to Table 1 for an overview of the notation we use). This delay can be subdivided into two different components: the time node u_i waits for its destination to wake up, the *rendezvous time* $\delta_R(i)$, and the time required to forward the packet, the *transmission time* δ_{tx}:

$$\delta_F(i) = \delta_R(i) + \delta_{tx}. \tag{2}$$

When δ_R and δ_{tx} are constant across all nodes, which is the case for most tree-based unicast data collection protocols with fixed wake-up frequency, the efficiency of data collection is improved by minimizing the path length h. However, this is not the case for *opportunistic routing*, where $\delta_R(i)$ can vary significantly across nodes depending on the number of potential forwarders n_i and their wake-up frequencies. The more forwarders a node has, the sooner one of them will wake up. This is because in opportunistic routing, messages are routed dynamically to the *first* available neighbor that provides enough progress for the given routing metric[1]. Concretely, when all potential forwarders have the same wake-up frequency ω, it is known [3, 15] that the rendezvous time $\delta_R(i)$ has an expected value of

$$\mathbf{E}\left[\delta_R(i)\right] = \frac{1}{(n_i + 1)\,\omega}. \tag{3}$$

This rendezvous process suffers from two key problems. First, the following trade off arises: increasing the number of forwarders n_i reduces the expected $\delta_R(i)$, but increases the path length because of sub-optimal routing paths. On the other hand, with few forwarders the path length is reduced, at the cost of longer forwarding delays. Therefore, finding the right number of forwarders is a complex balancing act [9], yet key to efficient data collection using opportunistic protocols.[2] Second, the expected rendezvous is

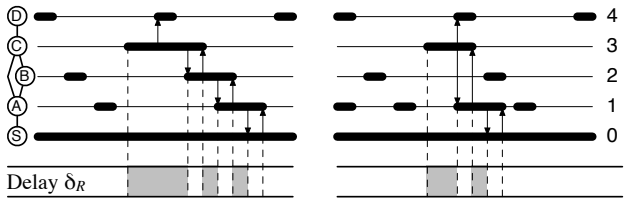
Table 1: Mathematical notations

Description	Symbol
Energy budget	DC_{max}
Number of forwarders	n
Wake-up frequency	ω
Hop distance to sink	h
Forwarding delay	δ_F
Rendezvous time	δ_R
Transmission time	δ_{tx}

When appropriate symbols will be indexed by node number (i) or hop distance from the sink (h).

[1]Not all routing metrics are strictly correlated with the hop-distance to the sink. Thus, it is possible that messages are forwarded to nodes whose hop-distance is equal or greater than the forwarding node. This is the case of EDC (cf. Section 4.3), which prefers faster paths to shorter ones.

[2]For example, Landsiedel et al. empirically found that the best trade-off in ORW is achieved with five potential forwarders [21].



(a) Fixed wake-up frequency (b) Staffetta

Figure 1: Opportunistic routing of a packet from node (C) to sink (S) – topology on the left, routing metric on the right (the lower, the better). By letting the node closer to the sink (A) wake up twice as often, the rendezvous times (in grey) and the number of hops are reduced.

load-agnostic cf. eq. 3. A node close to the sink has the same rendezvous cost as a leaf node, but it needs to rendezvous way more often because of its higher traffic load. Having the same expected rendezvous time across the whole network is like designing a city using only normal streets, without avenues or highways: latency and energy consumption increase unnecessarily.

To solve these two problems, we propose Staffetta, a low-overhead and fully distributed duty-cycle adaptation scheme that adjusts at runtime the rendezvous time of each node to its current local needs. *The basic idea behind our solution is very simple. Instead of setting a constant wake up frequency for all nodes, we set a constant energy budget. All nodes will die at the same time. This means that nodes need to make local decisions regarding their use of power: if their observed rendezvous time is long (short), they should wake up less (more) often.* As we shall see, this adaptive scheme is general enough to run underneath various opportunistic routing methods, and improves the performance of these methods on all fronts: the delivery rate is maintained or even improved, the network lifetime is increased, and the latency is reduced.

3. STAFFETTA

Staffetta builds upon the intuition that, by letting nodes closer to the sink wake up more frequently than those farther away, one can drastically shorten the forwarding delay while reducing the average path length. As an example, Figure 1 compares an opportunistic collection mechanism with fixed wake-up frequency (a) against one where the node closer to the sink (node A) wakes up twice as frequently (b). Since in Figure 1(b) A increases its activity, it wakes up before B and gets selected as forwarder by C. This not only reduces the forwarding delay of C, but also decreases the number of hops needed to deliver C's message to the sink.

In opportunistic routing, the forwarding choices of nodes depend on two aspects: i) the rendezvous time, and ii) the routing metric. A node forwards a packet to the neighbor that wakes up *first* and provides *enough* routing progress with respect to some metric, trading good routing choices for lower forwarding delays. Thus, by making nodes closer to the sink wake up more often, it is possible to improve opportunistic data collection in two ways: rendezvous with nodes closer to the sink becomes both *more probable* and *more efficient*.

3.1 Activity Gradient

Staffetta adjusts the wake-up frequency of nodes to their current forwarding costs. This creates an *activity gradient*, where nodes closer to the sink wake up more often than those farther away. Staffetta achieves this in a distributed fashion and with minimal runtime overhead, as explained next.

i) *Fix an energy budget.* Given a desired network lifetime as determined by the application requirements, Staffetta imposes a maximum duty cycle DC_{max}, $0 < DC_{max} < 1$, that is equal for all nodes in the network. This value determines the maximum fraction of time each node can keep its radio on. Since the radio is often the most power-hungry component on a typical low-power sensor node, DC_{max} effectively ensures that nodes stay within a fixed energy budget.

ii) *Adjust the wake-up frequency to stay within the energy budget.* Given the forwarding delay $\hat{\delta}_F(i)$ measured by each node at runtime, a node computes its wake-up frequency as

$$\omega_i = \begin{cases} \infty & \text{if it is the always-on sink} \\ DC_{max} \,/\, \hat{\delta}_F(i) & \text{otherwise.} \end{cases} \quad (4)$$

Note that unlike the analytical expression in eq. (2), $\hat{\delta}_F$ is based on each node's runtime observations, thus it takes into account implementation-specific delays, such as packet retransmission and channel sensing. Whenever the observed $\hat{\delta}_F$ changes, a node updates its wake-up frequency.

How does this policy create an activity gradient? In the presence of an always-on sink, the activity gradient emerges as follows. All nodes start with a fixed wake-up frequency, as in traditional opportunistic routing. However, since the sink does not duty-cycle its radio, its direct neighbors within radio range experience extremely short forwarding delays that are approximately equal to δ_{tx}—the rendezvous with the sink is almost instantaneous. Thus, any node that is a 1-hop neighbor of the sink adapts its wake-up frequency to

$$\omega_1 \approx DC_{max} \,/\, \delta_{tx}. \quad (5)$$

The higher activity of the sink's 1-hop neighbors, in turn, reduces the forwarding delay of the 2-hop neighbors, which then increase their wake-up frequency. This *cascading effect* propagates (with a decaying factor) throughout the network, creating what we call the *activity gradient* of Staffetta.

3.2 Analysis

While the behavior of a single node (e.g., adjusting the wake-up frequency) is fairly easy to describe and understand, the emergent behavior of the system (e.g., the resulting activity gradient) is more complex and difficult to predict. To gain a further understanding about Staffetta's gradient formation, we now present a numerical evaluation and a simple model. *This analysis is not meant to define the assumptions or guidelines used for the practical implementation of Stafetta, but to provide a clean setup for understanding Staffetta's macro properties.* A detailed description of Staffetta's practical implementation is presented in Section 4.

Figure 2 depicts a numerical simulation of opportunistic routing with and without Staffetta. The simulation is based on the topology shown at the bottom. Except for the sink's 1-hop neighbors, a node has 3 forwarders, an initial wake-up frequency $\omega = 5\,\text{Hz}$, a data rate of 1 Hz, and $DC_{max} = 15\,\%$.

Without Staffetta, the wake-up frequency, shown at the top, is constant across all nodes, and so is the forwarding de-

lay as shown below and captured by eq. (3). With Staffetta, the short rendezvous of the sink's 1-hop neighbors leads to a high wake-up frequency, as stipulated by eq. (4). The sink's 2-hop neighbors observe a higher forwarding delay than the 1-hop neighbors and adjust their wake-up frequency accordingly. This process further propagates throughout the network and has two important consequences. First, end-to-end packet latency is reduced significantly, because this metric is determined by the sum of forwarding delays on all hops, as captured by eq. (2). Moreover, the node's duty cycle, shown in the chart right above the topology, is also reduced drastically, because the nodes with the highest load spend less time and energy forwarding packets (the duty cycle can be roughly estimated as the forwarding delay times the load).

Note that, often, Staffetta nodes consume less energy than what it is in their budget. This is because Staffetta schedules the wake-up frequency assuming there is always a packet to forward (*worst-case budget allocation*). In case no packet needs to be forwarded, a node will only wake up for listening, thus saving the energy of the rendezvous process.

Another important observation is that the activity gradient of wake-up frequencies decreases exponentially, as visible in the top chart in Figure 2. We now derive a simple model to understand how the gradient's shape and steepness are controlled by the network topology (*width* and *depth*) and the energy budget. For mathematical tractability, we assume that i) no collisions or message retransmissions occur, and ii) all nodes have the same number of forwarders n. Given that the forwarding delay perceived by nodes at h hops from the sink is $\mathbf{E}\left[\delta_R(h)\right] + \delta_{tx}$, we can describe their activity gradient as

$$\omega_h \approx \frac{DC_{max}}{\mathbf{E}\left[\delta_R(h)\right] + \delta_{tx}} = \frac{DC_{max}}{1/((n+1)\omega_{h\text{-}1}) + \delta_{tx}}. \quad (6)$$

Since usually in duty-cycled networks $\delta_R \gg \delta_{tx}$, we can further simplify the above expression by setting $\delta_{tx} = 0$ for all nodes with $h > 1$. Thus, we obtain

$$\omega_h \approx DC_{max}(n+1)\omega_{h\text{-}1} = (DC_{max}(n+1))^{h\text{-}1}\omega_1. \quad (7)$$

Even though the resulting model is a simple approximation it captures the characteristics of the wake-up gradient.

According to this model, the activity gradient attains the maximum frequency at the sink's 1-hop neighbors (ω_1) and decreases with geometric rate $\mathrm{DC}_{max}(n+1)$. This geometric rate maps the trend that can be observed in Figure 2. In summary, eq. (7) conveys two important points: i) smaller energy budgets DC_{max} result in steeper activity gradients, and ii) increasing the number of forwarders n flattens the gradient.

In practice, this means that in dense and *wide* networks, where the number of forwarders increases, we could lower the energy budget DC_{max} without affecting the resulting activity gradient. Lowering the energy budget DC_{max} will extend the overall network lifetime, since in Staffetta the energy budget determines an upper bound on the maximum energy consumption of all nodes, as clearly visible in Figure 2. On the contrary, in *deep* networks (networks with a large diameter), DC_{max} should be increased so that the gradient could extend to the nodes furthest from the sink.

3.3 Explaining Staffetta's Performance Gains

Next, we analyze in more detail the benefits of Staffetta's activity gradient on packet latency, routing overhead, and network lifetime, the three key metrics of data collection applications [16].

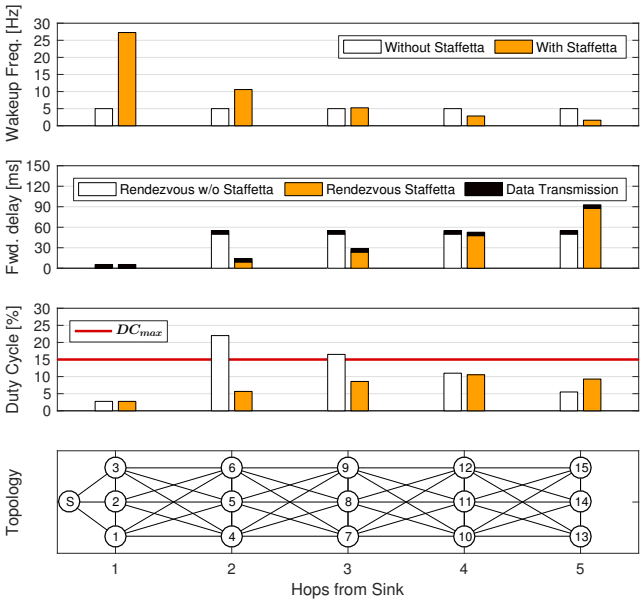


Figure 2: Numerical simulation of opportunistic routing with and without Staffetta. *With Staffetta, the closer nodes are to the sink (left side), the higher their wake-up frequency, and thus, the smaller their forwarding delay. This leads to lower latencies and duty cycles. The energy budget DC_{max} is an upper bound on the duty cycle of every node.*

Packet latency. The end-to-end latency depends on the *forwarding delays* and the *path length* between source and sink. Previously, we noted that by drastically reducing the forwarding delay of nodes closer to the sink, the overall end-to-end latency is also reduced, cf. eq. (2). Now we will discuss how Staffetta further reduces end-to-end latency by implicitly choosing short paths and exploiting temporal links.

While traditional collection protocols minimize latency by routing along a shortest-path routing tree, opportunistic protocols reduce the forwarding delay by selecting the first available node among a set of *candidates* that improve the routing metric over a certain threshold. Therefore, the wake-up frequency plays an important role in reducing packet latency of opportunistic mechanisms. First, it affects the forwarding delay of nodes; the higher the frequency, the lower the delay cf. eq. (3). Second, it biases the forwarding choices of nodes. In particular, given a node i with n_i forwarders, each with wake-up frequency $\omega_1 \ldots \omega_{n_i}$, the probability of selecting node j as a forwarder is

$$P_{ij} = \frac{\omega_j}{\omega_1 + \cdots + \omega_{n_i}}. \quad (8)$$

When forwarders wake up with a fixed frequency, they have the same probability of being selected. Thus, as mentioned in Section 2, the size of the set of potential forwarders set must be chosen carefully. Too many forwarders and the performance will be similar to a random walk (long paths, but short forwarding delays); too few forwarders and the mechanism will resemble deterministic routing (short paths, but long forwarding delays).

With Staffetta, on the other hand, the number of forwarders becomes a secondary concern: forwarding choices are inherently biased towards the nodes that are closer to the sink i.e., the ones that wake up more often cf. eq. (7) and (8). Therefore, *Staffetta improves opportunistic protocols by biasing the forwarding choices towards the candidate nodes that provide the highest routing progress, effectively reducing the path length of messages.*

Even if a "bad" node (i.e., a forwarder that offers little progress and long forwarding delays) is included in the forwarding set, the selection probability will be low as the wake-up frequency decreases with the distance to the sink. The effect is that Staffetta "routes" along paths whose length is comparable to mechanisms based on shortest-path routing trees, while maintaining the benefits of opportunistic protocols, namely, shorter forwarding delays, better load balancing, and exploitation of temporal links, that is, short-lived, long-distance links [29].

The exploitation of temporal-links is particularly interesting, because in Staffetta temporal links that provide better progress towards the sink, will have a higher probability of being selected. Consider for example node 8 in Figure 2, with three forwarders (nodes 4, 5 and 6) each waking up at frequency $\omega_4 = \omega_5 = \omega_6 = 11\,\text{Hz}$. According to eq. (8), the forwarding probabilities are $P_{84} = P_{85} = P_{86} = 0.\bar{3}$. Now assume that, for a short period of time, node 8 is able to communicate with node 2, waking up at frequency $\omega_2 = 27\,\text{Hz}$. The forwarding probabilities are now heavily biased towards the temporally-available new node, with $P_{82} = 0.45$, and $P_{84} = P_{85} = P_{86} = 0.18\bar{3}$. Note that, with a constant wake-up frequency (i.e., no Staffetta) the forwarding probabilities will be $P_{82} = P_{84} = P_{85} = P_{86} = 0.25$, almost halving the probability of exploiting the temporal link between nodes 8 and 2. Therefore, *Staffetta is able to drastically increase the probability of exploiting temporal links.*

Routing overhead. In static networks, routing overhead is usually a minor problem. Since nodes must build the routing structure only once, they can spend a significant amount of resources (energy, bandwidth) and amortize the costs over time. Unfortunately, in practice it is difficult to have real static conditions. Even if nodes do not physically move, the topology continuously changes due to link-quality fluctuations and uncontrolled sources of interference [5].

When the topology changes more often, nodes must constantly update their routing structures. Amortizing the overhead over time becomes less and less possible, up to the point that the mechanism is continuously using part of the channel bandwidth to maintain the routing structure. This itself limits the data rate of collection mechanisms, increases the chances of packet collisions, and incurs high energy overhead.

Staffetta overcomes these limitations as follows. First, Staffetta introduces only a minimal overhead to the existing protocol stack. Staffetta's mechanism requires just local observations (of the forwarding delay). The overhead thus consists of a few bytes of memory for storing the latest observations and trivial computations to determine the wake-up frequency, whereas it utilizes *no* additional bandwidth at all. Therefore, *empowering collection protocols with Staffetta comes with negligible overhead.*

Second, Staffetta allows data collection protocols to reduce their own overhead. Since Staffetta's activity gradient biases the opportunistic selection of a forwarder already towards the sink, it is possible to exploit Staffetta in a cross-layer fashion. In particular, the continuous neighbor discovery, needed by collection protocols to maintain their routing structure, can be avoided and substituted with the biased opportunistic forwarders provided by Staffetta. For example, to select the best forwarders in ORW [21], nodes must keep an up-to-date list of their neighbors' routing metric. While the current metric can often be piggybacked (e.g., on beacons and acknowledgments), link dynamics could force nodes to actively request their neighbors for updated metric information, increasing the routing overhead. Using Staffetta, the best forwarders are often the ones that wake up first, so nodes can avoid maintaining their neighbor's routing metric and simply select the first node provided by the opportunistic mechanism, which is simpler and more efficient.

Network lifetime. Network lifetime, often defined as the time until the first node (or critical group of nodes) in the network depletes its battery, is a very important metric for real-word deployments. This metric depends on the battery capacity of nodes and the average power consumption of the most energy-demanding node, the one that will deplete its energy first. To this end, it is more rewarding to reduce the maximum energy consumption among all nodes rather than the average energy consumption in the network. This is precisely what Staffetta achieves by imposing a fixed energy budget. However, note that although all nodes run with the same budget, they spend it differently. Nodes next to the sink can forward messages without any rendezvous delay, and spend the saved energy on waking up more frequently; nodes at the edge of the network do not need to forward any data and can spend their complete budget on rendezvous with parent nodes that wake up at a low frequency.

4. IMPLEMENTATION

We implement Staffetta within the default protocol stack in the Contiki operating system. As shown in Figure 3, our implementation of Staffetta and the protocols we compare it against has three layers. First, a *duty cycle* layer that uses either a fixed wake-up frequency or the adaptive Staffetta approach. Second, a baseline implementation of *opportunistic communication* that is inspired by the state-of-art ORW, but contains several enhancements to make opportunistic routing robust to packet duplicates, which is a common problem of opportunistic schemes. Third, we describe our implementations of three *routing metrics*, two of which are at the core of well-known protocols, namely ORW [21] and BCP [26].

The protocols, from which we take inspiration, are designed for different operating systems and platforms, with many implementation details that may hide the real benefits of their core mechanism, which is in our opinion the routing metric. Using a common implementation of the opportunistic data collection mechanism (baseline) allows us to perform a fair, controlled comparison, where differences in performance are solely due to the routing metric and the Staffetta mechanism. *Our goal is not to state that Staffetta is better than the state of the art, but rather that it makes the state of the art better.* That said, our implementations provide comparable or better performance than the original protocols. In the following, we describe the implementation of the three layers in Figure 3 in more detail.

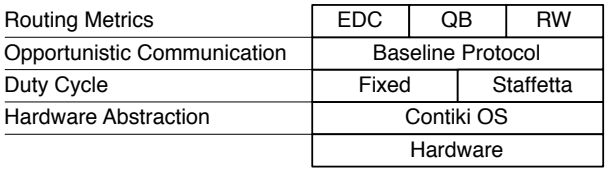


Figure 3: Our implementation consists of three distinct layers based on the default protocol stack in Contiki.

4.1 Staffetta

Staffetta's mechanism consists of adapting the wake-up frequency of nodes based on the observed forwarding delay $\hat{\delta}_F$. Thus, implementing Staffetta is rather straightforward.

Measuring the forwarding delay. We empirically measure the fixed communication delays δ_{tx}. For the baseline mechanism, depicted in Figure 4, δ_{tx} includes the time needed to listen to the channel to check if it is clear for transmission (δ_l), and the time needed to transmit the selection packet (δ_s).

To quantify the actual rendezvous time ($\hat{\delta}_R$), each node measures the period between the first beacon sent and the first acknowledgment received. Due to the stochastic nature of the rendezvous delay, our implementation averages the last 20 measurement using a simple moving average (SMA) to produce a stable, yet reasonably agile activity gradient that is robust to small disturbances such as fast link fluctuations.

Computing the wake-up frequency. Once δ_{tx} and $\hat{\delta}_R$ are determined, the wake up frequency is computed according to equation (4). In particular, after each successful message exchange, our implementation of Staffetta updates the average rendezvous time and the wake-up frequency ω, and schedules the next wake-up in $1/\omega$ seconds.

However, setting the wake-up frequency based on local information could lead to some complications. Different from traditional duty-cycling techniques, in which the activity rate depends on the application's traffic requirements, Staffetta bases the wake-up frequency solely on the forwarding delay, which depends on the network topology. In effect, the wake-up frequency is a function of the distance to the sink and the (average) number of forwarders at each hop. This could be problematic in case of weakly connected nodes, located at the edge of a large network. For these nodes, communication could be so expensive (due to long rendezvous times) that maintaining the required data rate will make the node consume more energy than budgeted. There are two approaches to handling this issue.

i) Give priority to the application requirements. A solution to this problem is capping the minimum frequency to the data rate of the application. Capping the minimum wake-up frequency has the consequence that equation (4) could be violated, causing the energy budget to be exceeded as more data could be sent than can be afforded (given the long rendezvous time with the forwarders). This effect is exacerbated by a poor selection of forwarders, as the gradient is effectively flattened (negligible bias), leading to longer paths and extra traffic, hence, raising energy consumption even more.

ii) Give priority to the energy budget. Using this approach, nodes can adapt their wake-up frequency to a value that is lower than the application requirements. The obvious consequence is that packets will be queued at the source, leading to longer latencies and eventually packet drops, likely violating latency and delivery requirements. However, if the application is delay-tolerant, the problem can be mitigated by aggregating packets at the source as the energy costs are mostly dictated by the rendezvous costs. An even more important observation is that only the nodes at the edge of the network suffer from this; most nodes will run on a higher wake-up frequency serving application data at the right pace.

In our implementation we decided to test Staffetta under the most stringent conditions, that of violating our core principle: equation (4). Thus, we adopt the first approach and use a minimum frequency (0.1 Hz by default), which can be set by the application.

4.2 Baseline

Inspired by opportunistic routing protocols such as ORW, our baseline mechanism is based on low-power listening [27] and opportunistic anycast, and works as follows (see Figure 4): Given a wake-up frequency as decided by the duty cycling mechanism, nodes wake up periodically to listen for incoming messages (*listening*). If no message is received (channel is clear), nodes start probing the channel with beacons (B) until a viable forwarder wakes up (*forwarding*). Each beacon contains the source's routing metric so that, whenever a node receives a beacon, it can compare the received metric with its own and send an acknowledgement (A) only in case of routing progress towards the sink. In case of multiple colliding ACKs, which is a known problem of opportunistic mechanisms [21], nodes may be able to decode only the stronger ACK due to power capture [31]. If capture does not occur, all receivers would forward the data packet, causing duplicates. To ameliorate this problem, ORW relies on overhearing and a probabilistic back-off ($P = 0.5$) that potential forwarders follow when retransmitting an ACK.

Even with these mechanisms, ORW suffers from duplicate packets, in particular in scenarios with high data rates and high densities. In the worst case the duplicates get duplicated themselves en route to the sink, leading to an exponential growth in the number of duplicates and posing serious scaling limitations. Considering that Staffetta works by (smartly) increasing the activity of nodes in the network, the 'duplicated packets' problem is expected to be exacerbated. To increase the resilience of our baseline protocol to packet duplicates, we implement a 3-way handshake. A third 'selection' packet (S) is added during the rendezvous process. If the sender cannot decode a single ACK, it does not send S. Potential forwarders will then re-transmit their ACKs but with a back off probability of 0.5. Upon decoding an ACK, the sender transmits S. As long as a selection packet is received, the duplication problem is completely avoided. If a selection packet is lost, the nodes that sent an ACK can proceed in two ways: i) they discard the received beacon containing the data payload, or ii) they proceed forwarding the packet. In the first case, a data packet is lost; in the second case, a duplicate is created. We observed that the number of duplicates due to a lost S was not that high, thus, to maintain a high delivery ratio, our implementation follows the second approach. Note that the 3-way handshake

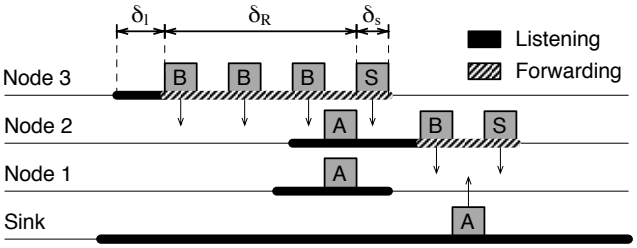


Figure 4: Baseline opportunistic mechanism with timing delays for listening (δ_l), rendezvousing (δ_R), and transfer of the select packet (δ_s).

(beacon-ack-select) is also useful to implicitly filter out low-quality links without the need for a link-quality estimator.

4.3 Routing Metrics

Routing metrics are at the core of data collection protocols and, together with the MAC protocol used, largely determine their performance. For this reason, we implemented three routing metrics from the state-of-the-art protocols on top of our baseline mechanism (see Figure 3).

Expected Duty Cycle (EDC). First proposed in ORW [21], EDC adapts ETX [7] to opportunistic, duty-cycled networks. In essence, EDC describes the expected duration until a packet reaches its intended destination (i.e., the packet latency) by summing the expected forwarding delays at each hop. In ORW, the forwarding delay (i.e., single-hop EDC) of a node i is computed as the inverse of the sum of the link quality of the forwarders plus a weight δ_{tx} that reflects the transmission costs

$$EDC_i = (1/\sum q_{ij}) + \delta_{tx}.$$

EDC_i approximates the expected forwarding delay of node i [15] by (partially) using probes. In our implementation, we avoid the overhead of neighbor discovery and link quality estimation and compute this metric directly using the average of the last observed delays

$$EDC_i = \hat{\delta}_F = \hat{\delta}_R + \delta_{tx},$$

where $\hat{\delta}_R$ is computed by averaging the last 20 observed rendezvous times.

Queue Backlog (QB). QB is the metric used by the Backpressure Collection Protocol (BCP) [26]. In BCP, nodes select the neighbor with the shortest queue backlog as the forwarder. Considering that the sink absorbs all packets, this simple mechanism leads to a gradient that decreases the size of nodes' queues towards the sink and can be used to make per-packet forwarding decisions. As demonstrated in the original paper, this agile metric shows superior delivery performance especially in dynamic networks experiences external interference or featuring a mobile sink. Different from the implementation in BCP, where the metric is mixed with ETX, our implementation solely uses the queue backlog for the opportunistic forwarding decisions. Specifically, unlike BCP, where the packet is forwarded to the neighbor with the shortest queue, we forward the packet to the neighbor that wake up first and has a queue backlog that is smaller than its own. Low-quality links, which in the original implementation are discarded by ETX, are filtered out by the baseline's 3-way handshake.

Table 2: Testbed characteristics and settings

Testbed	FlockLab	Indriya
Number of nodes	28	139
Tx power [dBm]	0	0
Network diameter [hops]	5	5
Node degree	8.5	22
Radio channel	26	26
Node ID of sink	1	2

Random Walk (RW). Using RW, nodes choose their forwarders randomly, independently of their routing progress. Since no routing metric is used, random walk mechanisms are lightweight and perform great in highly dynamic networks, where the overhead of maintaining routes is too high. On the other hand, in static networks, random walk mechanisms are too dispersive and suffer from long path lengths. Our implementation of RW takes inspiration from the opportunistic and duty-cycled method presented in [4]. In that mechanism, all nodes wake up with the same frequency, and thus have the same probability of being opportunistically selected. With Staffetta underneath, this probability will not be uniform anymore, but biased towards the better links.

Note that, for the first two routing metrics (EDC and QB), nodes forward their information to the first neighbor that wakes up *and* provides routing progress. For RW, instead, nodes forward their message to the first neighbor that wakes up, independently of its distance to the sink.

5. EVALUATION

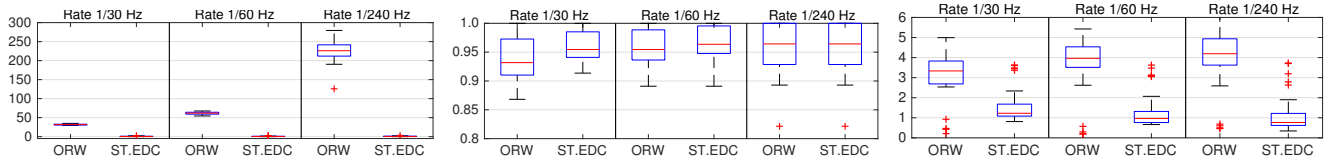
In this section, we use measurements from two testbeds to evaluate the performance of Staffetta and the effectiveness of using the activity gradient to guide the forwarding decisions.

5.1 Methodology

Testbeds. We use the FlockLab [23] and Indriya [8] testbeds. Out of the 28 TelosB on FlockLab, 25 nodes are deployed in several offices and hallways on one floor, while three nodes are deployed outside on the rooftop of an adjacent building. The resulting network is quite sparse: each node has on average 8.5 neighbors, and the diameter is 5 hops. The 139 TelosB on Indriya are spread across three floors in an office building. With 22 neighbors on average, the network is much denser than FlockLab; the diameter is 5 hops. In all experiments, nodes transmit at the highest power setting of 0 dBm, using channel 26 to limit the impact of external interference from co-located Wi-Fi networks. Table 2 lists all testbed settings, including the node ID of the sink.

Compared schemes. We compare the following schemes:

- **ORW:** This is the original TinyOS-based implementation of ORW [21], the current state-of-the-art opportunistic routing protocol for low-power wireless. ORW runs on top of the standard LPL layer in TinyOS with a wake-up frequency of 0.5 Hz, which matches the configuration used by Landsiedel et al. [21].
- **EDC, QB, RW:** These are implementations of different opportunistic schemes built on top of the baseline mechanism described in Section 4.2. The schemes are named after the specific routing metric they employ: Expected Duty Cycle (EDC), Queue Backlog (QB),



(a) Latency [s] - FlockLab (b) Delivery Ratio - FlockLab (c) Duty Cycle [%] - FlockLab

Figure 5: Latency, delivery ratio, and duty cycle of ST.EDC and ORW for different data rates. *Using Staffetta's dynamic duty-cycle adaption, ST.EDC outperforms ORW in all metrics, achieving several-fold improvements in latency and duty cycle.*

and Random Walk (RW). As detailed in Section 4.3, EDC is the metric underlying ORW [21], and QB is similar to the metric used by BCP [26], which is the state-of-the-art routing protocol for mobile sink settings. All three schemes run on top of the default LPL layer and opportunistic anycast in Contiki with a wake-up frequency of 1 Hz.

- **ST.EDC, ST.QB, ST.RW:** These schemes are the same as EDC, QB, and RW above except that instead of using a fixed wake-up frequency Staffetta's dynamic duty-cycle mechanism is employed to adjust the wake-up frequency at runtime, as described in Sections 3 and 4.1.
- **DIRECT:** This scheme, implemented atop the baseline mechanism, fully exploits Staffetta by using the activity gradient *directly* as a "routing metric:" a node forwards a packet to the neighbor that wakes up first and has a wake-up frequency higher than itself. In other words, instead of using an explicit routing metric, such as EDC or QB, to decide whether a neighbor provides routing progress, DIRECT makes this decision using the gradient of wake-up frequencies inherent in Staffetta.

Energy budget. When Staffetta is enabled, we set the energy budget DC_{max} to 7.5 % on FlockLab and to 6 % on Indriya.

Metrics. Our evaluation uses the following three key performance metrics of real-world applications [5, 6]:

- *Latency:* The time between when a packet is generated by the source node and when that packet is successfully received by the sink. To measure latency, we leverage the time synchronization among nodes on FlockLab and the serial logging of Indriya to timestamp both the generation and the successful reception of each packet.
- *Duty cycle:* The fraction of time the radio is turned on. We measure duty cycle in software using Contiki's accurate energy profiler [10].
- *Delivery ratio:* The fraction of packets generated by the source nodes that is delivered to the sink. We determine delivery ratio based on the unique sequence numbers of packets successfully received at the sink.

We also measure *path length*, the number of times a message is relayed before reaching the sink, to explain the behavior and performance of a given routing scheme. We compute these metrics based on three independent runs with the exact same settings. To allow each protocol to bootstrap its operation, we start measuring all metrics after an initial delay of 1 minute. We also give each protocol sufficient time (i.e., 1–5 minutes) to empty their packet queues before the end of each run. Unless stated otherwise, we report the results as median, 25th and 75th percentiles, and minimum and maximum values across all nodes in a testbed using box plots; statistical outliers are represented by crosses.

5.2 Comparison against the State of the Art

We begin by comparing ST.EDC against ORW to quantify the performance gains of Staffetta over the state of the art.

Settings. We perform 1-hour experiments on FlockLab. All nodes except the sink generate packets at the same rate. We test three different data rates in different runs: 1 message every 30 seconds (1/30 Hz), 1 message every minute (1/60 Hz), and 1 message every 4 minutes (1/240 Hz). The latter equals the data rate used in the ORW paper [21]. Nodes are given 5 minutes to empty their message queues at the end of a run.

Results. Figure 5 shows latency, delivery ratio, and duty cycle of ST.EDC and ORW for different data rates. We observe that ST.EDC outperforms ORW across all metrics regardless of the data rate. Using Staffetta to adapt the nodes' wake-up frequencies, ST.EDC delivers packets on average 79–452× faster than ORW, while achieving a 2.75–9× lower duty cycle. ST.EDC also provides a higher delivery ratio, especially as the traffic load increases. Moreover, ST.EDC reduces the variance in all metrics across nodes compared with ORW.

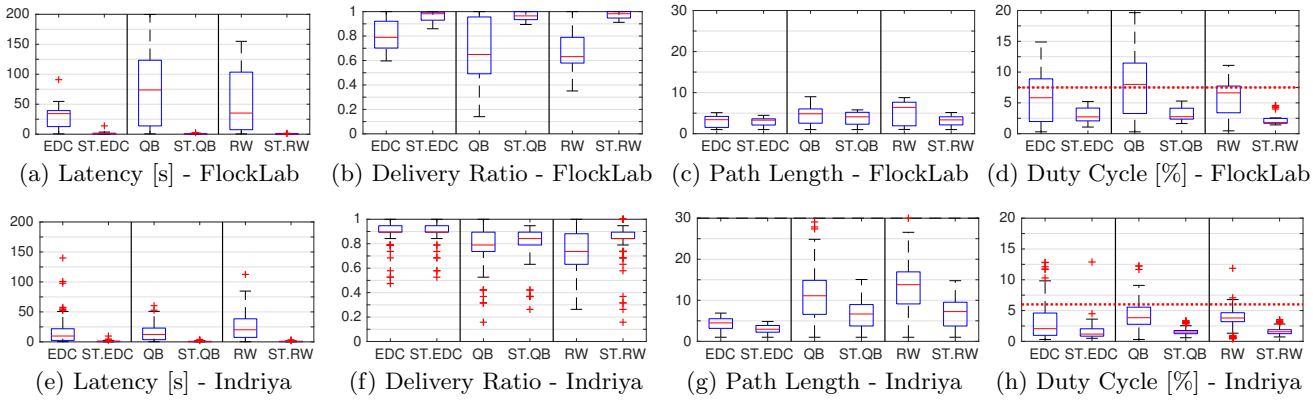
The results demonstrate the significant performance gains enabled by Staffetta's dynamic duty-cycle adaption. The basic mechanism is conceptually simple and lightweight to implement, but also highly effective in practice. The next section investigates the reasons for these improvements.

5.3 Benefits across Diverse Routing Metrics

We now explore in detail the benefits of Staffetta in terms of performance for different well-known routing metrics.

Settings. To this end, we conduct a set of 10-minute experiments on FlockLab and Indriya with EDC and ST.EDC, QB and ST.QB, RW and ST.RW. Except the sink, all nodes generate packets at the same fixed rate. We test two data rates: 1 packet every 10 seconds (1/10 Hz) on FlockLab and 1 packet every 30 seconds (1/30 Hz) on Indriya. Nodes have 1 minute to flush their packet queues at the end of each experiment.

Results. Figure 6 depicts for each scheme the measured performance and path length. Using Staffetta, instead of a fixed wake-up frequency, results in superior performance across the board, while reducing the variance among nodes.



(a) Latency [s] - FlockLab (b) Delivery Ratio - FlockLab (c) Path Length - FlockLab (d) Duty Cycle [%] - FlockLab

(e) Latency [s] - Indriya (f) Delivery Ratio - Indriya (g) Path Length - Indriya (h) Duty Cycle [%] - Indriya

Figure 6: Performance metrics and path length with and without Staffetta for different routing metrics on the FlockLab and Indriya testbeds. *Using Staffetta significantly boosts performance in all metrics compared with a fixed wake-up frequency.*

Table 3 lists the significant improvements in median performance enabled by Staffetta. Looking at both testbeds and all routing metrics, Staffetta reduces latency by 12.2–75.0× and increases delivery ratio by 1.0–1.5×, while reducing the energy costs, measured in terms of duty cycle, by 1.6–3.1×.

The improvements are generally higher on FlockLab than on Indriya. The reason is that opportunistic routing schemes naturally benefit from a higher node density: the larger number of potential forwarders on Indriya leads to, for example, a better load balancing and reduced forwarding delays. Thus, Staffetta is particularly beneficial in networks that are fairly sparse in general or contain low-density node clusters through which the bulk of the traffic must be funneled.

But how does Staffetta achieve these performance gains? The key lies in the activity gradient. By letting each node dynamically adapt its wake-up frequency to the observed forwarding delay, nodes closer to the sink are more active than those at the fringe of the network. This is visible in Figure 7, which plots the average wake-up frequency of nodes against their hop distance from the sink on both testbeds. We clearly see the geometrical decay for increasing hop distance. This way, the bottleneck typically found in duty-cycled networks around the sink is largely eliminated, allowing for faster and more reliable packet delivery. At the same time, the nodes' wake-up frequencies are proportional to their respective traffic loads, which results in a more effective use of energy. We now take a closer look at each performance metric.

Understanding lower latencies with Staffetta. Latency is mainly determined by three factors: forwarding delay, path length, and message backlog (i.e., the accumulation of packets in a node's queue). As discussed below, Staffetta reduces all of them, which explains the overall reduction in latency.

Looking at Figures 6(c) and 6(g), we can observe the reduction in path length for the different routing metrics. RW and QB tend to select diverse paths that are not necessarily directed towards the sink, whereas EDC chooses forwarders that definitely provide high routing progress. Thus, Staffetta reduces the path length significantly for RW and QB, while EDC leaves little room for further shortening the paths.

Table 3: Performance gains of Staffetta (cf. Figure 6)

Testbed	Latency	Delivery Radio	Duty Cycle
FlockLab	37.5–75.0×	1.3–1.5×	2.4–3.1×
Indriya	12.2–20.1×	1.0–1.2×	1.6–2.2×

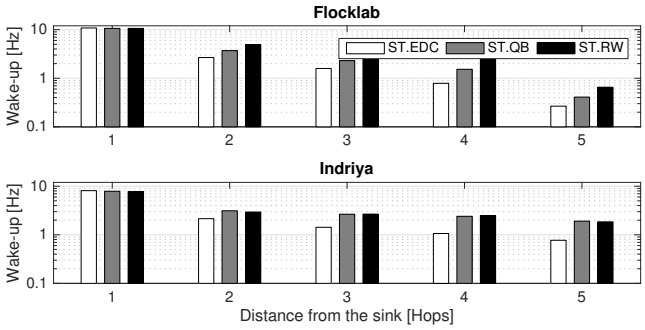
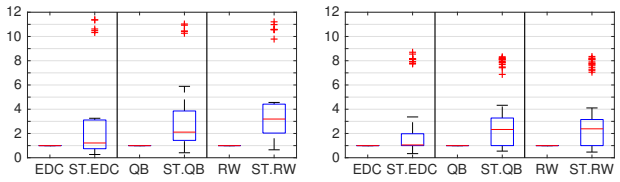


Figure 7: Activity gradient on FlockLab and Indriya.

As stipulated by eq. (3), the higher the wake-up frequency of forwarders, the lower the forwarding delay. Figure 8 shows the wake-up frequency of nodes on both testbeds for all routing metrics. We see, for example, that the median wake-up frequency with Staffetta is consistently above the fixed wake-up frequency of 1 Hz. This indicates that Staffetta reduces the forwarding delay for more than half of the nodes on the testbeds. A few nodes experience a higher forwarding delay due to a lower wake-up frequency, but this has a negligible impact on latency since these nodes are located at the fringe of the network and hence carry only very little traffic.

Finally, we note that the major reason for the high latency when using a fixed wake-up frequency is message backlog. This is because most packets are funneled through a small set of nodes actually delivering them to the sink. These nodes are effectively a bottleneck whenever the fixed wake-up frequency (here 1 Hz) is too low to sustain the aggregate load load around the sink (2.7 pkts/s on FlockLab and 4.5 pkts/s on Indriya). As a result, packets are backlogged



(a) FlockLab (b) Indriya

Figure 8: Wake-up frequency (in Hz) on the two testbeds.

at these bottleneck nodes, severely increasing latency. By contrast, with Staffetta nodes close to the sink are highly active, which increases the capacity and removes the bottleneck completely.

Understanding higher delivery ratios with Staffetta. Delivery ratio is affected by packets lost en route and packets stuck indefinitely in some node's queue. We tackle the former by the beacon-ack-select handshake of the baseline protocol, which ensures that a link is only used when communication (temporarily) succeeds in both directions. This 3-way handshake not only reduces packet loss, it also serves to keep packet duplicates under control, which is a common problem in opportunistic protocols. However, without Staffetta, duplicates still fill the nodes' queues, thereby delaying other packets. This is also confirmed by comparing latency in Figures 6(a) and 6(e) with delivery ratio in Figures 6(b) and 6(f): the longer the latency, the lower the delivery ratio. Staffetta, instead, reduces the message backlog, which improves delivery ratio except for a few badly connected nodes.

Understanding lower duty cycles with Staffetta. In a nutshell, Staffetta increases the activity of nodes where needed, while it reduces the energy consumption wherever possible. Figures 6(d) and 6(h) indicate that Staffetta reduces the duty cycle significantly, but also reduces the variance. This shows that Staffetta sets an appropriate wake-up frequency for every node. As mentioned in Section 3.2, the worst-case budget allocation of Staffetta cause nodes to overestimate their energy consumption and, thus, consume less than DC_{max}. This policy is beneficial in scenarios where *all* nodes need to be operational and lifetime is determined by the first node depleting its battery. Staffetta significantly boosts the network lifetime in these scenarios.

5.4 Adaptiveness to Mobile Sink Dynamics

Support for mobile sinks is a key requirement in participatory sensing and Internet of Things (IoT) applications [2]. We evaluate Staffetta's performance and ability to cope with the highly dynamic network topology in such scenarios.

Settings. To enable repeatable mobility patterns, we emulate a mobile sink that wanders about the FlockLab testbed area. Specifically, we select a sequence of three nodes, labeled 1, 2, and 3 in Figure 9, that passes through several offices, hallways, and even several buildings. One node at a time acts as the sink for 200 seconds. Every change in the sink designation triggers a drastic change in the forwarding decisions. We repeat the experiment multiple times, using different routing metrics both with and without Staffetta in different runs.

Figure 9: Nodes 1, 2, and 3 acting as a sink on FlockLab in the experiments of Section 5.4, and the corresponding clusters of nodes in their vicinity used to plot Figure 10.

Results. Figure 10 demonstrates that Staffetta adapts the activity gradient promptly and correctly as the sink designation changes. The figure charts the wake-up frequencies of nodes over time using ST.RW, grouped into three disjoint clusters: nodes in the vicinity of sink 1 (top), in the vicinity of sink 2 (middle), and in the vicinity of sink 3 (bottom). The results for the other Staffetta-enabled schemes are very similar. We see that Staffetta creates a distinct activity gradient with respect to the current sink designation. Depending on a node's hop distance to the preceding sink, it takes 5–30 seconds to adapt its wake-up frequency after a change. Staffetta quickly forms a new gradient, driving messages toward the new sink.

The net result is an improved resilience against network dynamics. This is confirmed by looking at Figures 11 and 12, which display latency and throughput (packets received per second) at the current sink with (ST.RW) and without (RW) Staffetta as the sink designation changes from 1 to 2. Despite a significant reduction in latency with Staffetta, we note that Staffetta provides a consistent and high throughput, whereas without Staffetta the throughput fluctuates widely, especially after the change in sink destination at time 200 seconds. The spikes are due to nodes with a considerable backlog that suddenly become neighbors of the (new) sink and hence get the chance to empty their queues. In contrast to this, due to its prompt and correct adaptation decisions, Staffetta does not suffer from this problem, delivering packets reliably and consistently two orders of magnitude faster as the sink moves.

5.5 Using Staffetta's Activity Gradient for Efficient Zero-overhead Routing

To forward messages efficiently and reliably, nodes typically maintain a list of their neighbors' routing costs with respect to some metric (e.g., ETX [7], EDC [21], or QB [26]). Keeping this state up to date requires to execute tasks such as neighbor discovery and information sharing via periodic beaconing that cost energy, bandwidth, and time. We show

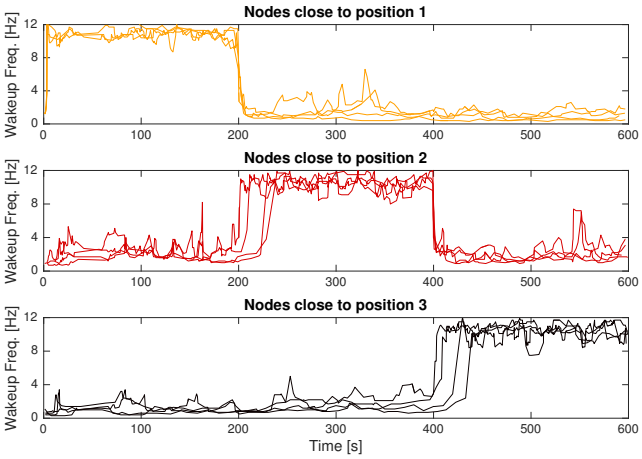


Figure 10: Wake-up frequency of nodes running ST.RW with an emulated mobile sink on FlockLab (cf. Figure 9). *Staffetta quickly adapts the activity gradient, which helps deliver messages efficiently and reliably as the sink moves.*

in the following that Staffetta avoids this overhead by using its inherent activity gradient to effectively guide the forwarding decisions without maintaining explicit neighbor tables.

Settings. We run another set of experiments with DIRECT, ST.EDC, ST.QB, and ST.RW on FlockLab. However, unlike previous experiments, we use a bursty traffic pattern typical of applications that respond to external events [12]. To this end, we initially let each node accumulate 20 packets in its local queue before transmitting. Afterwards, we measure until all nodes have transmitted their 20 packets, plus 1 minute to allow route-thru packets to leave the network. This results in a throughput of 2.4–4.6 packets per second at the sink.

Results. Figure 13 shows for each scheme the measured performance and path length. We see that DIRECT achieves a performance that is comparable or even superior to the other three schemes. This occurs without the additional overhead with regards to implementation complexity and resource usage incurred in maintaining an explicit routing metric. Using DIRECT, a node simply forwards to the neighbor that wakes up first and has a wake-up frequency higher than itself. In this way, packets essentially "surf on Staffetta's activity gradient," which is both efficient and highly effective.

6. RELATED WORK

The amount of work related to energy-efficient communication protocols that duty cycle the radio is vast. In this section, we focus on the two categories that are most related to our work on Staffetta: protocols for data collection over duty-cycled networks, and mechanisms that dynamically schedule the duty cycle to improve performance.

Data collection protocols. The first generation of data collection protocols for sensor networks saw duty-cycling as an issue that had to be overcome. The Collection Tree Protocol (CTP) [16], for example, was built on top of the fact that broadcast is a primitive that should be used rarely, because it consumes a lot of energy under duty-cycling techniques. Therefore, CTP uses unicast for all data transmissions and broadcast only for route discovery [22]. The Broadcast Free Collection protocol [28] took this strategy one step further, and avoids the use of broadcast altogether; routes are discovered by eavesdropping unicast transmissions of neighboring nodes. Although this approach saves significant amounts of energy, the price of overhearing in duty-cycled networks is rather high leaving room for further improvements.

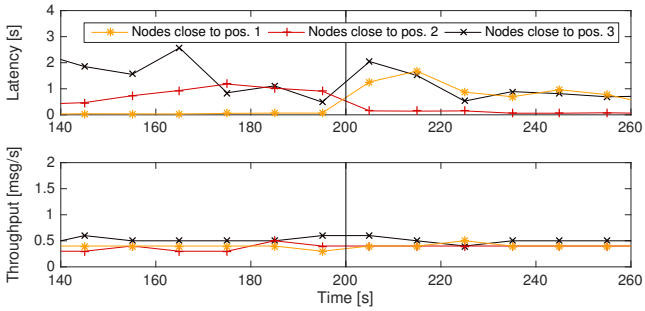


Figure 11: Latency and throughput when Staffetta is enabled as the sink designation changes from 1 to 2 (cf. Figure 10). *Staffetta reduces latency by two orders of magnitude, while providing a consistently, high throughput.*

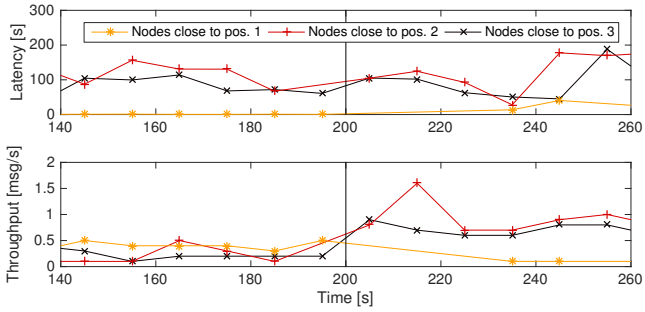
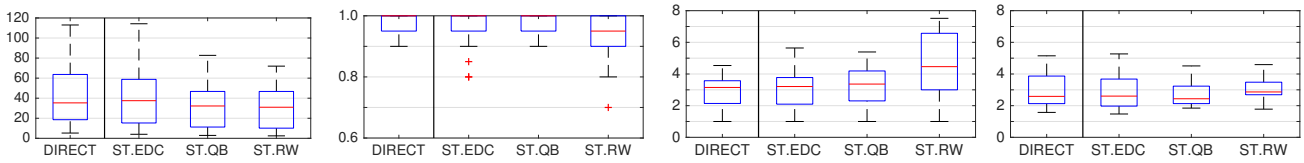


Figure 12: Latency and throughput without Staffetta as the sink designation changes from 1 to 2 (cf. Figure 10).

Glossy [14] and Low-power Wireless Bus (LWB) [13] remove the need for route discovery by using an efficient flooding mechanism based on constructive interference. Every message can be received by every node in the network, making data collection trivial. Duty cycling happens between synchronized active periods, in which every node uses its radio. The downsides are the need of a central network coordinator and the dependency on tight time synchronization.

Opportunistic protocols such as ORPL [11], ORW [21], and ORiNoCo [30] enhance the efficiency of routing in duty-cycled networks by exploiting *opportunistic anycast* to reduce forwarding delays, balance load more evenly, and take advantage of temporal links. As demonstrated in the previous section, Staffetta improves the performance of these kind of protocols by biasing the forwarder selection to nodes closer to the sink. Staffetta can also be regarded as a generalization of SOFA [4], the first protocol to exploit opportunistic anycast for efficient random walks in duty cycled networks. In this case, Staffetta allows SOFA to perform biased random walks that are directed towards a sink.



(a) Latency [s] (b) Delivery Ratio (c) Path Length [hop] (d) Duty Cycle [%]

Figure 13: Performance metrics and path length with Staffetta for DIRECT and different explicit routing metrics on the FlockLab. *DIRECT uses Staffetta's activity gradient to effectively guide its forwarding decisions without the additional overhead in term of implementation complexity and resource usage incurrent when maintaining explicit routing costs.*

Finally, a long body of literature explores the problem of route-aware sleep scheduling. DMAC [20], for example, was the first to explore the use of a phased duty-cycled schedule, in which forwarders wake up one after the other to reduce the overall forwarding delay. Contrary to Staffetta, DMAC requires explicit notifications ("more-to-send" packets) to cope with dynamic traffic and involves an initial overhead that can be considered negligible only for quasi-static networks.

Dynamic duty-cycling mechanisms. No matter how efficient data collection protocols strive to be, their performance will always be bounded by the intermittent activity of the radio underneath, usually with a fixed frequency. Therefore, several works adapt the duty cycle to values that still meet the requirements of (traditional) data collection.

Theoretical works [18, 19], for example, have analyzed optimal duty-cycling parameters (like the link-layer wake-up frequency) to minimize the packet latency of opportunistic collection protocols in energy-constrained networks. However, the results are derived based on assumptions that are made more for mathematical tractability than to resemble typical WSN settings (e.g., Poisson arrivals vs. strict periodic reporting), thereby ignoring the intricacies and overheads of practical implementations.

As a more realistic approach, pTunes [32] models communication efficiency as an optimization problem with multiple application-level objectives, such as end-to-end latency, network lifetime, and end-to-end reliability. Through an efficient network dissemination technique (based on Glossy), pTunes periodically collects (centrally) network information and, based on the application requirements, computes the optimal duty-cycling parameters and disseminates them to the nodes. Staffetta complements this centralized approach by providing a good but sub-optimal solution that is fully distributed, has minimal overhead, and addresses the unique characteristics of opportunistic protocols.

Similarly, ZeroCal [25] balances the energy consumption of nodes by adapting their wake-up frequency. Compared to ZeroCal, we believe that Staffetta has three main differences. First, ZeroCal requires radio and data traffic parameters to solve a distributed optimization problem, while Staffetta requires only one parameter (the energy budget) and a simple operation to adapt the wake-up frequency locally. Second, ZeroCal incurs a higher overhead since it requires to monitor the state of nodes' children, while Staffetta does not require any such explicit input. Furthermore, ZeroCal focuses on the unicast primitive, while Staffetta addresses opportunistic protocols and their anycast primitive. Finally, Staffetta's energy budget is similar in spirit to TinyDB's "lifetime-based query" [24], which automatically sets the sampling rate to match a given lifetime requirement.

7. CONCLUSIONS

In this paper we addressed the fundamental issue of efficient data collection in wireless sensor networks, where protocol designers must balance communication performance and energy efficiency. While opportunistic collection protocols successfully trade off these two antithetical goals, they do not exploit the full potential of the network. We have demonstrated that, instead of *coping* with the challenges of low-power communication, collection protocols can *tame* the duty-cycle mechanism and use it to their advantage, raising performance and reducing (average) energy consumption at the same time.

Our low-level Staffetta mechanism sets the wake-up frequency according to the forwarding latency observed by the nodes; the less time is spent on forwarding, the more time can be spent on servicing incoming traffic (i.e. waking up more frequently). The net effect is that Staffetta sets up a gradient with nodes close to the sink waking up more frequently than nodes at the edge of the network. This gradient automatically steers opportunistic anycast traffic towards the sink as the probability of finding an awake neighbor is highest in the direction of the sink.

We implemented Staffetta in Contiki, and evaluated it with three opportunistic collection protocols on two different testbeds. The extensive set of experiments showed that Staffetta significantly reduces both packet latency and energy consumption, while maintaining high – or even improving – packet delivery ratios. As Staffetta does not need to maintain complicated routing metrics spanning the entire network, it can also handle network dynamics like link-quality fluctuations and node mobility really well. We found that Staffetta adapts its gradient in just a matter of seconds.

Acknowledgments

We would like to thank the anonymous reviewers and our shepherd Jie Liu for providing us detailed feedback on draft versions of this paper. This work was supported by the Dutch national program COMMIT/ (project P09 EWiDS), by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (CFAED), and, partially, by TU Graz.

8. REFERENCES

[1] M. Buettner, G. Yee, and E. Anderson. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *4th ACM conf. on Embedded networked sensor systems (Sensys)*, pages 307–320, 2006.

[2] J. Burke et al. Participatory sensing. In *1st Workshop on World-Sensor-Web (WSW '06)*.

[3] M. Cattani, M. Zuniga, A. Loukas, and K. Langendoen. Lightweight Cardinality Estimation in Dynamic Wireless Networks. In *13th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 179–189, 2014.

[4] M. Cattani, M. Zuniga, W. Matthias, and K. Langendoen. SOFA: Communication in Extreme Wireless Sensor Networks. In *11th European Conf. on Wireless Sensor Networks (EWSN)*, pages 100–115, 2014.

[5] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *8th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 277–288, 2009.

[6] S. Dawson-Haggerty, S. Lanzisera, J. Taneja, R. Brown, and D. Culler. @scale: Insights from a large, long-lived appliance energy wsn. In *11th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 37–48, 2012.

[7] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11:419–434, 2005.

[8] M. Doddavenkatappa, M. C. Chan, and a. L. Ananda. Indriya: A low-cost, 3D wireless sensor network testbed. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 90 LNICST:302–316, 2012.

[9] H. Dubois-Ferrière, M. Grossglauser, and M. Vetterli. Valuable detours: Least-cost anypath routing. *IEEE/ACM Transactions on Networking*, 19(2):333–346, 2011.

[10] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *EmNets*, 2007.

[11] S. Duquennoy, O. Landsiedel, and T. Voigt. Let the Tree Bloom : Scalable Opportunistic Routing with ORPL. In *11th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 1–14, 2013.

[12] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *4th Int. Symp. on Information Processing in Sensor Networks (IPSN)*, 2005.

[13] F. Ferrari, M. Zimmerling, L. Thiele, and L. Mottola. The low-power wireless bus. In *10th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 1–14, 2012.

[14] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient Network Flooding and Time Synchronization with Glossy. In *10th Int. Conf. on Information Processing in Sensor Networks*, pages 73–84, 2011.

[15] E. Ghadimi, O. Landsiedel, P. Soldati, S. Duquennoy, and M. Johansson. Opportunistic Routing in Low Duty-Cycled Wireless Sensor Networks. *ACM Trans. on Sensor Networks (TOSN)*, 10(4):67, 2014.

[16] O. Gnawali, R. Fonseca, and K. Jamieson. Collection Tree Protocol. In *7th ACM Conf. on Embedded Networked Sensor Systems (Sensys)*, pages 1–14, 2009.

[17] R. Jurdak, P. Baldi, and C. V. Lopes. Adaptive Low Power Listening. *IEEE Transactions on Mobile Computing*, 6(8):988–1004, 2007.

[18] J. Kim, X. Lin, and N. Shroff. Optimal anycast technique for delay-sensitive energy-constrained asynchronous sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 19(2):484–497, 2011.

[19] 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 (TON)*, 18(2):515–528, 2010.

[20] B. Krishnamachari and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *18th Int. Parallel Distrib. Process. Symp.*, pages 224–231, 2004.

[21] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. In *11th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 185–196, 2012.

[22] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *1st Symp. on Networked Systems Design and Implementation (NSDI)*, pages 15–28, 2004.

[23] R. Lim, F. Ferrari, and M. Zimmerling. FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *12th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 153–165, 2013.

[24] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[25] A. Meier, M. Woehrle, and M. Zimmerling. ZeroCal: Automatic MAC protocol calibration. In *6th Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, volume 1, pages 31–44, 2010.

[26] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. In *9th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 279–290, 2010.

[27] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *2nd Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 95–107, 2004.

[28] D. Puccinelli, S. Giordano, M. Zuniga, and M. Pedro Jose. Broadcast-free collection protocol. In *10th ACM Conf. on Embedded Network Sensor Systems (SenSys)*, pages 29–42, 2012.

[29] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *ACM Trans. on Sensor Networks*, 6(2):1–49, 2010.

[30] S. Untersch, C. Renner, and V. Turau. Opportunistic, Receiver-Initiated Data-Collection Protocol. In *In 9th European Conf. on Wireless Sensor Networks (EWSN)*, pages 1–16, 2012.

[31] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the Capture Effect for Collision Detection and Recovery. *2nd Workshop on Embedded Networked Sensors (EmNetS)*, pages 45–52, 2005.

[32] M. Zimmerling, F. Ferrari, L. Mottola, and T. Voigt. pTunes: Runtime Parameter Adaptation for Low-power MAC Protocols. In *11th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 36–38, 2012.