Load-Balanced Data Collection through Opportunistic Routing

Mathieu Michel*, Simon Duquennoy†, Bruno Quoitin*, Thiemo Voigt†‡
*Computer Science Department, University of Mons, Belgium
firstname.lastname@umons.ac.be
†SICS Swedish ICT AB, Sweden
{simonduq,thiemo}@sics.se
‡Uppsala University, Sweden

***Abstract*—Wireless Sensor Networks performing low-power data collection often suffer from uneven load distribution among nodes. Nodes close to the network root typically face a higher load, see their battery deplete first, and become prematurely unable to operate (both sensing and relaying other nodes' data). We argue that opportunistic routing, by making forwarding decision on a per-packet basis and at the receiver rather than the sender, has the potential to better balance the load across nodes. We extend ORPL, an opportunistic version of the standard routing protocol RPL, with support for load-balancing. In our protocol, ORPL-LB, nodes continuously adapt their wake-up interval in order to adjust their availability and attain a deployment-specific target duty cycle. We implement our protocol in Contiki and present our experimental validation in Indriya, a 93-nodes testbed. Our results show that ORPL-LB reduces significantly (by approximately 40%) the worst node's duty cycle, with little or no impact on packet delivery ratio and latency.**

I. Introduction

Reducing the energy consumption is one of the main requirements of Wireless Sensors Networks (WSNs). Over the last years, many MAC and routing protocols have been proposed to address the trade-off between energy consumption and performance (*e.g.*, in delivery ratio and latency). State-of-the-art low-power routing solutions rely on a tree topology, with *e.g.*, CTP [1] for data collection networks or RPL [2], the standard IETF protocol for low-power IPv6 routing.

Tree-based routing protocols such as CTP do not spread the load evenly among all nodes in the network [3]. Nodes with a large sub-tree typically have a higher energy consumption, will see their battery deplete first, eventually leading to network partitions. Load balancing aims at spreading the load evenly among the nodes in order to postpone as much as possible the first occurrences of battery depletion. A properly balanced network will keep full connectivity longer, and will also result in lower maintenance costs, allowing to replace a maximum of batteries or nodes at the same time.

Load-balancing routing protocols such as LB-RPL [4] operate by building a topology that minimizes forwarding hotspots, distributing the traffic load as evenly as possible among the nodes. A problem with this approach is that depending on the current topology, every node is responsible for forwarding any traffic coming from its sub-tree. Variations in the traffic load, link quality, and other factors affecting energy consumption can therefore not be accommodated for.

We argue that opportunistic routing offers a number of benefits when it comes to load balancing. In opportunistic routing protocols such as ORPL [5], the next hop is selected *during* packet transmission based on receivers' availability, rather than *before* transmission based on the sender's routing table. Having the decision made at the receiver means the latter can easily decline traffic whenever it is too loaded, and vice versa. Additionally, because the next hop is selected on a per-packet basis, it becomes easy to spread traffic bursts towards the sink over multiple next hops.

In this paper, we extend the existing protocol ORPL with load-balancing capabilities. Our protocol ORPL-LB adapts the wake-up interval of the nodes depending on their current and past energy and traffic conditions. The wake-up interval directly affects the node's availability as a forwarder, a key factor in the node's future load energy consumption. Each node selects its wake-up interval adaptation based on the comparison of its own current duty cycle (used as a proxy for energy consumption) with a network-wide target duty-cycle. The target duty cycle is either fixed statically or at runtime, offering a balance between performance and lifetime.

Our experimental results in a 93-nodes testbed [6] show that ORPL-LB reduces on average the energy consumption by approximately 40% for the energy hotspots with no impact on the reliability and a negligible impact on the network average energy consumption and latency.

The paper is organized as follows. First we introduce in Sec II some background notions including ORPL and its concepts. Then we motivate the need of load balancing for ORPL in Sec III. Our load balanced version of ORPL is presented in Sec IV and then evaluated in Sec V. Sec VI presents related work before Sec VII concludes.

II. Background

A. Duty cycling and ContikiMAC

Several radio duty cycling MAC protocols [7] have been proposed in the literature to address specific WSN requirements and constraints such as low energy consumption of battery operated nodes or limited bandwidth. Duty-cycled MAC protocols aim at reducing the energy consumption by having nodes keep their *radio-transceiver* off most of the time. We refer to the portion of time where the node has its radio turned on as *duty cycle*. Typically, to reach lifetimes in the order of several months or a few years, radio duty cycled networks aim for duty cycles in the range of a few percents, sometimes below one percent.

ContikiMAC [8] is an asynchronous duty cycling protocol. Nodes running ContikiMAC sleep most of the time and wake up periodically for a short time to check the medium. The node quickly performs two successive Clear Channel Assessments (CCA) to determine, based on the radio signal strength, if there is an incoming transmission. If the CCAs succeed (channel clear), the node goes back to sleep immediately. If the CCAs fail (channel busy), the node stays awake and a procedure called a *fast sleep optimization* tries to determine if the received signal is due to noise or to an incoming frame. In case of noise the node goes back to sleep.

To send a packet, a node running ContikiMAC repeatedly sends the full data packet until an ACK is received or the transmission times out after a duration equal to a wake-up interval. The packet destination field allows non-targeted receiving nodes to go back to sleep early in the packet reception.

In addition, a mechanism called "phaselock" allows a node running ContikiMAC to learn the wake-up schedule of a neighbor. By allowing a node to achieve a transmission with on average only two packets (the first being used to announce the transmission), the phaselock mechanism decreases the time and the energy spent in TX mode. A side benefit of the phase-lock is a reduction in channel utilization and therefore in the risk of collisions.

B. RPL

Based on CTP [1], RPL is a distance vector protocol for low-power and lossy networks using a tree-like topology, named a DODAG (Destination Oriented Directed Acyclic Graph). This graph is built based on the notion of rank which expresses the cost needed by a node to reach the root (see Fig 1). This design makes RPL particularly efficient for *collect-only* traffic patterns. *Any-to-any* traffic can also be handled – albeit at higher cost – by routing via common ancestor (see Fig 2).

In a data collection scenario, nodes always transmit to the same destination: the network root which acts as sink. The simple rooted topology of RPL allows each node to obtain a preferred parent acting as a next-hop to reach the root. A node sends a data packet to the sink through its preferred parent. This parent will then forward the packet through its own preferred parent until the packet reaches the sink. Looking at Fig 1, a packet sent by the node **G** will reach the sink via the nodes **D** and then **B**.

C. ORPL

ORPL [5] is an opportunistic extension of RPL based on ContikiMAC. In ORPL, the forwarding decision is made during transmission rather than before, using anycast transmissions. Transmitting nodes send the data repeatedly until

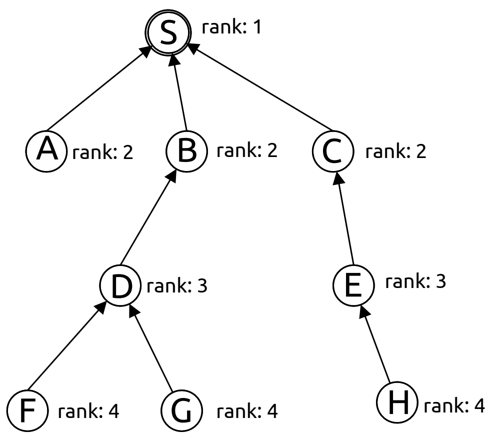

Fig. 1. RPL builds a DODAG topology

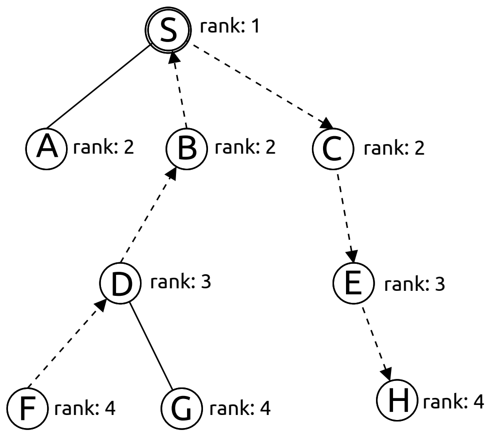

Fig. 2. RPL routes any-to-any traffic through a common ancestor: In this case node **F** sends its packets through the sink **S** to reach node **H**

receiving an ACK from any node closer to the destination. The set of the neighbors able to offer routing progress to a given node is called its "forwarder set". Therefore, the first node in the "forwarder set" that wakes up and successfully decodes the packet will acknowledge the packet and route it further. Like RPL, ORPL uses a DODAG topology, and supports both data collection and any-to-any traffic.

In data collection scenarios ORPL is similar to the ORW protocol [9], performing opportunistic routing along a gradient anchored at the root. Any-to-any transmission is supported by first routing the packet upwards to any common ancestor and then downwards to the destination.

III. Motivation: the Need for Load Balancing in ORPL

We argue that ORPL has the potential to better distribute the traffic among the nodes, because the routing decision is made at the receiver, on a per-packet basis. However, ORPL alone, without an explicit load balancing policy cannot avoid energy hotspots.

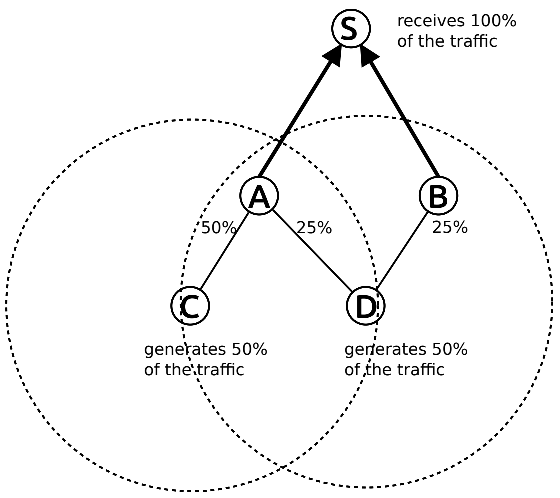

Fig. 3. Without LB and due to a higher number of descendants, node A would handle 75% of the traffic compared to only 25% for node B.

Fig 3 illustrates a case where the opportunistic nature of ORPL is not enough to spread the load evenly. In this topology, node **A** is a forwarder for **C** and **D** while **B** has **D** as its only child. By using ORPL, node **D** is free to use **A** or **B** as forwarders. We assume that the traffic is uniformly distributed among all parents. In this scenario, with ORPL, **A** is an energy hotspot, as it is a forwarder for 75% of the traffic.

A load-balanced extension of ORPL would aim at having **B** take more traffic from **D**, allowing node **A** to decrease its load.

There is a strong correlation between the energy consumption of a node and its network load. Indeed, with respect to their distance to the sink, not all nodes handle the same amount of traffic. While the leaves only have to transmit their own data packets, nodes closer to the sink also have to forward to the sink all the traffic originating from their descendants.

To characterize this behavior we have performed some ORPL experimentation in the Indriya testbed (93 Tmote Sky nodes). The results of a representative experiment are presented below. Fig 4 shows the packet load at every node, in an RPL topology, w.r.t. its average hop count to the sink. Given the nodes do not always used the same parent to reach the root, the hop count cannot be reduced to an absolute value and is then presented as the average number of hops required over time to reach the root. It can be observed that the nodes closest to the root/sink have the highest forwarding count, while the leaves have fewer packets to handle.

It is common to observe that nodes closer to the sink consume more energy than the leaves which handle less traffic [4]. Note that surprisingly, as illustrated by Fig 5, the nodes at a 1-hop distance from the sink, despite carrying a higher traffic load, consume less energy than the hops at a 2-hops distance [3]. This is explained by the fact that in most real-life deployment, the root/sink has a permanent power supply and then can stay always on. Hence, 1-hop nodes do not have to wait for their successor to wake up.

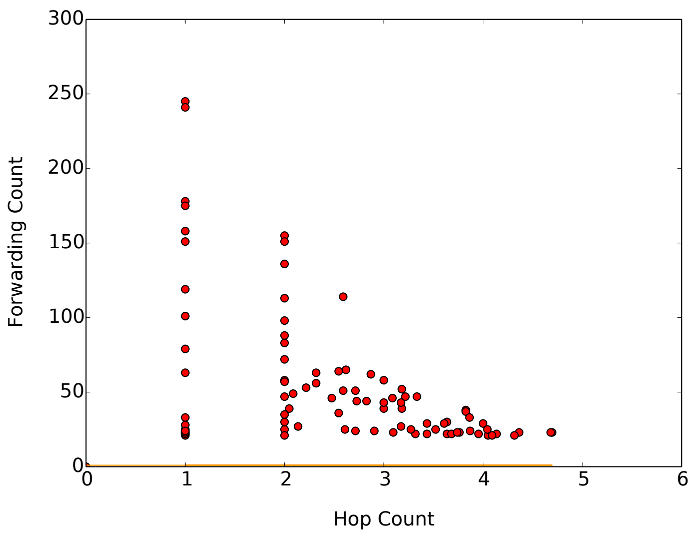

Fig. 4. Correlation between the average distance (in hops) from the sink and the network load: the lower the distance higher the load.

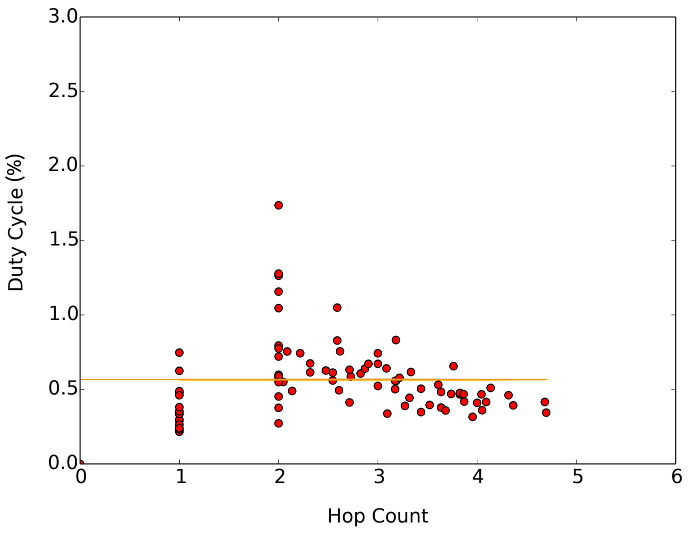

Fig. 5. Correlation between the average distance (in hops) from the sink and its energy consumption. Nodes closer to the sink have a higher energy consumption than the leaves, except for the sink's neighbor who benefit from an always-on parent.

IV. Design of ORPL-LB

We introduce ORPL-LB, a load-balanced, opportunistic version of RPL based on ORPL. The key idea in ORPL-LB is to have the nodes adjust their wake-up interval continuously to reach a target duty cycle. This target duty cycle can be either pre-defined in the configuration or determined at runtime. Increasing (resp. decreasing) the wake up interval allows nodes to reduce (resp. increase) their duty cycle to reach the target. By doing so, nodes that are lightly loaded will start waking up more frequently, hear and forward more traffic. As a result, their load will increase, alleviating overloaded neighbors. Conversely, nodes with a high load will react to their high duty cycle by increasing their wake-up interval. This leads to a reduction of both their traffic load and baseline idle consumption, helping them save energy in the future. This approach is illustrated on Fig 3: in this topology A handles 75% of the traffic while B only 25%. By using load balancing, node **B** would be able to decrease its wake-up interval, making itself more available for **D** and then more likely to handle **D**'s packets, allowing **A** to reduce its network load and duty cycle.

A. Duty Cycle Adaptation

In ORPL-LB, like in ORPL, all data-plane transmissions are anycast. The sender node transmits continuously until any of its neighbors wakes up, receives the packet, decides to forward it and acknowledges it. In such design, nodes can adjust their wake-up interval without the need to signal it to their neighbors. Waking up more often will simply result in more opportunities to forward; Sleeping more allows a node to reduce its load instantly.

The wake-up interval evaluation is done at regular interval of 2 minutes. To avoid excessive energy consumption, the wake-up interval is bounded by minimum (125 ms) and maximum (1000 ms) values. These values have been selected to avoid a waste of energy due on one hand to too frequent wake-ups and on the other to longer transmissions [5]. Indeed the maximum wake-up interval bounds the strobe length for anycast and broadcast transmissions, meaning that a node with a high wake-up interval will transmit longer strobes.

Every node starts with the ORPL default wake-up interval (500 ms) [5].

With the knowledge of a target duty cycle, nodes adjust their wake-up interval periodically by comparing the target with their own average duty cycle recorded since startup. The wake-up interval is only adapted if the distance to the target duty cycle exceeds a pre-defined threshold ht. In addition to this average duty cycle, the node computes its current periodic duty cycle to determine how to adapt the wake-up interval. More specifically, every node follows the following decision process periodically:

1) The node computes its current average duty cycle from the beginning DC_{avg}.
2) The node compares DC_{avg} to DC_{target} and makes one of the following decisions:
   - If $DC_{avg} < DC_{target} + ht$ the node will decrease its wake-up interval;
   - If $DC_{avg} > DC_{target} - ht$ the node will increase its wake-up interval;
   - Otherwise it keeps its wake-up interval unchanged.

With the purpose to reduce oscillations, a weighted moving average, accounting for both current periodic[1] (DC_{cur}) and past duty cycle values, is computed:

$$DC_{ewma} = \alpha \times DC_{cur} + (1 - \alpha) \times DC_{ewma}$$

[1]Over the last 2-minutes period

The DC_{ewma} value is used to make the decision on how much the wake-up interval must vary. By using a weighted moving average the nodes can track their recent duty cycle variations. The objective is, among others, to take into account cases where the average duty cycle DC_{avg} is higher (resp. lower) than DC_{target} but the node has already increased (resp. decreased) its wake-up interval to a value allowing the node to obtain current duty-cycle DC_{cur} close to the target.

Once the DC_{ewma} has been computed the node has to decide how much the wake-up interval has to be modified. This wake-up variation step WI_{step} is bounded by a maximum value WI_{max_step} (125 ms) to avoid excessive variation. A ratio $DC_{ratio\_bounded}$ is computed between the DC_{target} and the DC_{ewma} and used as a multiplier to WI_{max_step} to compute the increase:

$$\begin{aligned} DC_{ratio} &= \frac{DC_{ewma} - DC_{target}}{DC_{target}} \\ DC_{ratio\_bounded} &= \begin{cases} -1 & DC_{ratio} < -1 \\ DC_{ratio} & DC_{ratio} \in [-1, 1] \\ 1 & DC_{ratio} > 1 \end{cases} \\ WI_{step} &= DC_{ratio\_bounded} \times WI_{max\_step} \end{aligned}$$

This method is illustrated by the following example, where the following values are used:

- $DC_{target} = 1\%$
- $DC_{avg} = 0.90\%$
- $DC_{ewma} = 0.80\%$

Realizing that DC_{avg} is lower than DC_{target}, a node will decide to decrease its wake-up interval. To determine how to decrease the wake-up interval it will first compute the DC_{ewma} . With a value of -0.2 obtained for the $DC_{ratio\_bounded}$ (DC_{ratio} = -0.2) and considering a WI_{max_step} of 125 ms, the current wake-up interval would be decreased by 25 ms.

B. Wakeup Interval – Traffic Load Feedback Loop

The ORPL-LB load balancing mechanism could face two main problematic corner cases:

1) Despite waking up more frequently, a node with very few descendants is limited in its ability to increase its traffic load.
2) Despite waking up less frequently, a node located at a physical bottleneck (on the path towards the root for a large sub-tree) is limited in its ability to reduce its traffic load.

The first situation leads to a waste of energy, in a vain attempt to handle more traffic. The second situation causes an unnecessary increase of latency for the node's sub-tree due to a lower availability of their common ancestor.

To overcome these limitations, we consistently monitor the feedback loop between wake-up interval and traffic load. We aim at making sure that adjustments of the wake-up interval affect the node's traffic load as expected.

Every K (= 5) wake-up interval evaluation periods each node computes the following:

- the current average wake-up interval WI_{cur} from beginning.

- the actual average packet forwarding rate Pkt_{cur} from beginning.

These two values are compared with the results observed for the previous period (WI_{prev} and Pkt_{prev}) as follows:

- The node compares WI_{cur} with WI_{prev}:
  1) If $WI_{cur} > WI_{prev}$, we expect that given the node increases its wake-up interval its traffic load will decrease:
     - If $Pkt_{cur} < Pkt_{prev}$: everything keeps unchanged.
     - If $Pkt_{cur} \geq Pkt_{prev}$: the load balancing is not effective, the previous wake-up interval WI_{prev} is restored.
  2) If $WI_{cur} < WI_{prev}$, we expect that given the node decreases its wake-up interval its traffic load will increase:
     - If $Pkt_{cur} > Pkt_{prev}$: everything keeps unchanged.
     - If $Pkt_{cur} \leq Pkt_{prev}$: the load balancing is not effective, the previous wake-up interval WI_{prev} value is restored.

C. Target Duty Cycle Definition

In some deployment scenarios, we expect the target duty cycle to be fixed offline, at design time, reflecting the application lifetime requirement. Our experience with ORPL-LB is that from initial experiments without load-balancing it is possible to identify a realistic interval for the target duty cycle. However, we can also see the need for an automatic, runtime adaptation of the target duty cycle.

To address this issue we propose the following methodology to automatically and dynamically find the optimal value for the DC_{target}. The main intuition is to try to obtain the lowest objective keeping the balance between the energy spent for the transmission and for the wake-up procedure.

Fig 6 is a schematic illustration of the relationship between the energy consumption of the nodes within a network running ORPL and the length of their average wake-up intervals [5]. From this figure it appears that short average wake-up interval within the network involves a high average duty-cycle caused by too frequent wake-up forcing the nodes to spend a lot of energy for the wake-up procedure (side *A* of Fig 6). Conversely, a high average wake-up interval within the network results in nodes spending more energy to reach a specific destination (side *B* of Fig 6). Indeed, the number of strobes required by ContikiMAC[2] to reach a neighbor is bounded by the maximum wake-up interval.

We look for the optimal balance between the energy spent on wake-up and the energy spent for packets transmission. This optimal point, which is illustrated on Fig 6 can be found by the sink and then propagated through the network with the following methodology.

[2]The opportunistic anycast transmissions within ORPL prevent the use of the ContikiMAC phaselock mechanism to reduce the length of the transmission.

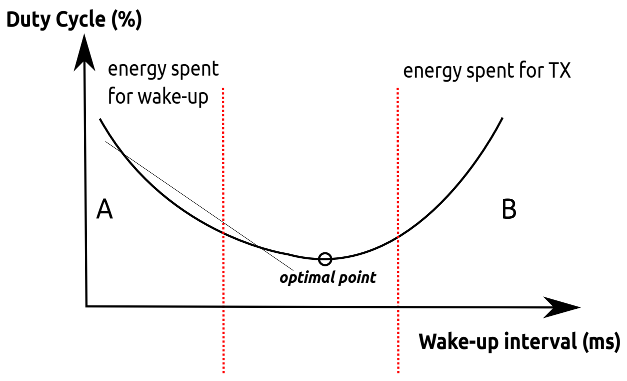

Fig. 6. With ContikiMAC the average duty cycle depends on the energy spent for transmission and wake-up. A higher wake-up interval results in more energy spent for transmissions while a lower one involves an increased energy consumption due to more frequent wake-ups.

The sink propagates the current duty cycle target DC_{target} through RPL DIO messages (periodic beacons). The sink starts by disseminating a reasonably high value as target: $DC_{target} = DC_{targetMAX}$. In our case $DC_{targetMAX}$ could be fixed at 1% which is almost twice the average duty cycle observed during previous ORPL experiments. With ORPL-LB, once a high DC_{target} received, a node will gradually reduce its wake-up interval as a reaction of thinking being under-loaded. The nodes will then spend more energy due too frequent wake-ups than for transmissions.

Periodically, the sink decreases the propagated DC_{target} by a fixed value $DC_{targetSTEP}$ (= 0.05%) until a significant increase of the energy spent for transmission is detected.

Each node can evaluate its transmission energy consumption via the Contiki's *energest* module, which measures the time spent by a node in the following states: LISTEN, RX and TX. By including these data in a field added through the data packet headers, the nodes are able to forward such information to the sink without requiring any additional traffic.

By evaluating this information the sink is able to detect traffic variation and in case of higher (resp. lower) transmission cost to increase (resp. decrease) the target duty cycle to better match the network situations.

The use of the DIOs and the fact that in collect-only traffic pattern all the data frames are sent to the sink makes this approach suitable for collect-only traffic pattern without any additional overhead. This methodology is less suitable for non-collect-only scenarios where the data packets are not necessarily targeting the root. Providing to the root the information about the energy spent would require a periodic transmission of additional control packets from each node.

D. High Load Optimization for ORPL

In addition to the load balancing optimization we brought the following enhancement to ORPL (-LB): Before sending an acknowledgement and then try to forward a packet, a node will first check the status of its transmission queues. If the queue occupation is higher than a certain threshold, the node will not acknowledge the packet, letting other, less saturated nodes handle this packet. This optimization allows ORPL(-LB) to be more efficient under high loads.

V. Evaluation

In this section we assess ORPL-LB's performance in periodic data collection scenarios. We evaluate its capacity to extend the network lifetime by reducing the energy consumption of hotspots. We compare these results with both RPL and original ORPL. Unfortunately we cannot compare our protocol against LB-RPL [4] which has only been evaluated through simulations in NS-2 [10].

The evaluation focuses on the following metrics: duty cycle (DC), latency and packet delivery ratio (PDR). The duty cycle represents the portion of time spent with the radio turned on. The latency measures the average end-to-end delay from the source sending a packet to the destination receiving it. The (end-to-end) packet delivery ratio is the ratio between the number of application data packets received by the sink over the number of application data packets sent.

A. *Methodology*

The Indriya testbed [6], which has already been used to evaluate ORPL, has been selected as test environment. Indriya is a testbed that currently consists of around 93 *Tmote Sky* nodes distributed over three floors in office environment. Given the unavailability of node 20 previously used as root to evaluate ORPL [5], we have selected node 1 as root, also located in the third floor, trying to maximize the network diameter. We use the maximum transmission power (0 dbm) of the cc2420 radio transceiver, which leads to a diameter of five hops in our experiments.

The application scenario is periodic data collection, where every node sends a packet every 2 minutes on average, resulting in a load of 0.76 packet per second at the sink. All experiments are run for one hour, and the metrics are computed after the first 10 minutes of the experiment, when the topology has fully converged. Whenever averages are shown, they are from at least three runs of the same experiment. In the case of ORPL and ORPL-LB, we run experiments for almost two hours, where the first hour runs the basic ORPL, and we enable load-balancing to evaluate ORPL-LB in the second half. By avoiding the randomization between two experiments due to the staggered starting times of the nodes or random temporary interference, this method allows us to get a fair comparison between ORPL and ORPL-LB.

B. *Dissection of an ORPL-LB Run*

We start by analysing a single, representative run to better understand the effect of load-balancing on ORPL.

Fig 7 shows the resulting per-node average duty cycle, with and without load-balancing. By dynamically adapting nodes wake-up interval, ORPL-LB allows to balance the duty cycle more evenly among nodes. As a result, the highest duty cycle drops from 1.73% to 1.18%; a decrease by 32%.

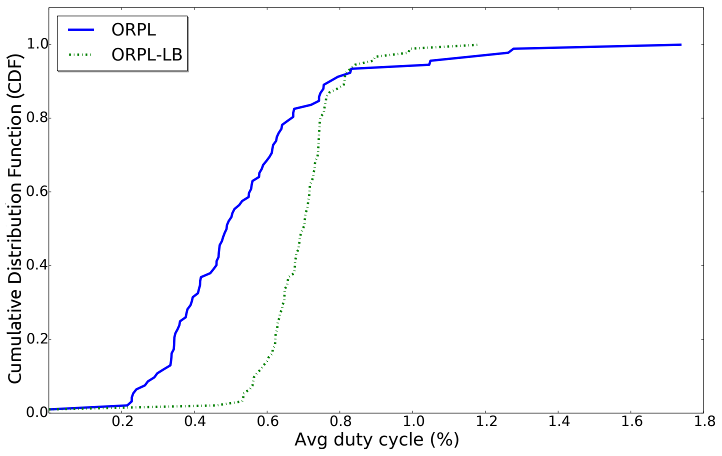

Fig. 7. ORPL-LB allows to reduce the consumption of energy hotspots compared to ORPL. While ORPL sees its hotspots reach a maximum duty cycle of $\sim$1.75%, this value is reduced to $\sim$1.2% with ORPL-LB.

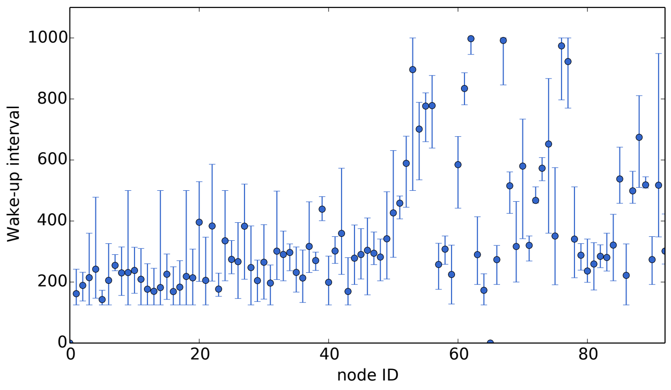

Fig. 8. Average wake-up interval (+min/max) for each node after enabling load balancing. The interval varies both between different nodes and with time.

Fig 8 shows the average and min/max (error bars) wake-up intervals obtained with ORPL-LB. As illustrated, the wake-ups profiles are very heterogeneous and vary over time. Nodes that are only one hop from the sink (mostly nodes with a low node ID) can transmit at very low cost as the sink is always on. The energy saved as a result is put by ORPL-LB into more frequent wake-ups, further increasing the forwarding capacity. Nodes further away from the sink have very heterogeneous profiles. Whether they are leaf nodes, have many children, a heavy or light traffic load affects their duty cycle, used by ORPL-LB to set the wake-up interval.

Fig 9 shows different metrics as a timeline, in a 110 minute experiment where load balancing is enabled around minute 55. The metrics are computed after 10 minutes to ensure that ORPL has converged. Overall, the system continues to operate with comparable global performance after load-balancing is enabled. Most importantly, we do not observe any oscillation due to the operation of ORPL-LB. This is due to use of the average duty cycle DC_{avg} and the weighted moving average duty cycle DC_{ewma} respectively to decide if a modification is needed and to quantify this modification.

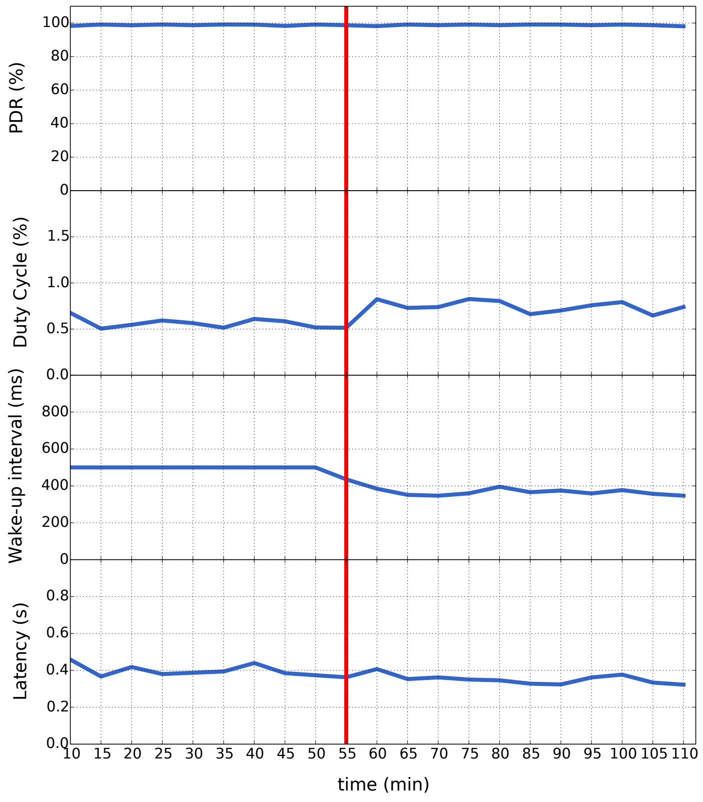

Fig. 9. Impact of the activation (vertical line) of ORPL-LB after almost 1 hour experiment on the following metrics: PDR, duty cycle, latency and wake-up interval. Load balancing activation shows little impact on performance metrics such as PDR and latency, but results in a slight increase in average duty cycle.

Using load-balancing leads to a slight increase of the average duty-cycle. This is the price to pay for more evenly distributed load, which helps reducing the worst case duty cycle. Moreover this higher average duty cycle can be explained by a non-optimal DC_{target} definition. Indeed the target duty cycle has been arbitrarily fixed to 0.60% which is the average duty cycle observed on previous ORPL experiments. One also has to consider the pre-defined threshold ht to explain this difference.

C. Comparison vs. ORPL and RPL

Tab. I summarizes the overall performance of RPL, ORPL, and ORPL-LB. For ORPL, we set the wake-up interval to 500 ms, which we found to give the best results. For RPL, we present results with an interval of 500 ms and 125 ms, as both lead to a different trade-off between energy and latency.

At a 500 ms interval, RPL offers the lowest average duty cycle of all, but also a latency 3 times higher than that of ORPL, ORPL-LB, or RPL with 125 ms interval. At a 125 ms interval, RPL offers comparable latency with ORPL and ORPL-LB, but with higher duty cycle and significantly lower delivery ratio. Overall, ORPL and ORPL-LB have an interesting best latency-energy balance and are more robust.

Metric	RPL (500ms)	RPL (125ms)	ORPL	ORPL-LB
Highest DC (%)	1.46	3.39	2.48	1.46
Avg DC (%)	0.47	0.92	0.57	0.63
Avg PDR (%)	97.95	97.45	99.67	99.78
Avg Lat (s)	1.42	0.52	0.41	0.46

TABLE I. EXPERIMENTS SUMMARY: *highest duty cycle, average duty cycle, average packet delivery ratio, average latency*. COMPARED TO ORPL, ORPL-LB IS ABLE TO REDUCE THE HIGHEST DUTY CYCLE BY MORE THAN 40%.

As intended, ORPL-LB reduces the highest duty cycle in comparison with ORPL, from 2.48% to 1.46%. Note that this decrease by more than 40% comes with the side effect of a slightly increased average duty cycle. Interestingly, the highest duty cycle with ORPL-LB is similar to that of RPL with a 500 ms interval – but in such settings all other metrics are clearly in favour of ORPL-LB, in particular a packet loss rate one order of magnitude lower, and latency three times lower (see Tab. I).

D. Network Lifetime

In this section we evaluate the impact of the reduction of the highest duty cycles by ORPL-LB on the network lifetime compared to classic ORPL. We consider the network lifetime as the time during which the network is able to maintain the PDR above a pre-defined threshold. The fact of the PDR dropping below this threshold would be considered as unacceptable in regards with the network obligations. For that purpose we have designed a specific experiment where the nodes have a pre-defined energy quota. When the energy spent (measured as radio-on time) by a node exceeds a given threshold, the node is considered as dead and then is unable to receive or send any data packets. Based on the power consumption of the most energy-efficient nodes across the different experiments, the threshold has been calibrated to ensure that no node has a more than 4 hours autonomy.

Fig 10 shows the gradual extinction of the nodes observed during one particular (representative) experiment. The upper part shows the number of nodes active at any time, while the lower part shows the resulting delivery ratio, i.e. the ratio of packets received at the sink over the total number of packets that would have been sent, should all nodes be active.

Both ORPL and ORPL-LB result in longer lifetime than RPL. RPL has been used with a wake-up interval of 125ms in order to keep similar performances (in terms of latency) to ORPL(-LB). ORPL-LB maintains the PDR above 90% for almost 140 min while ORPL goes below this threshold after 105 min of experiment, and RPL witnesses such drop in delivery as early as minute 80. Considering 90% as threshold under which the network could not maintain an adequate quality of service, ORPL-LB is then able to extend the network lifetime by 33% compare to ORPL.

It is worth noting that although ORPL-LB keeps most of the nodes operational for a long period of time, it eventually

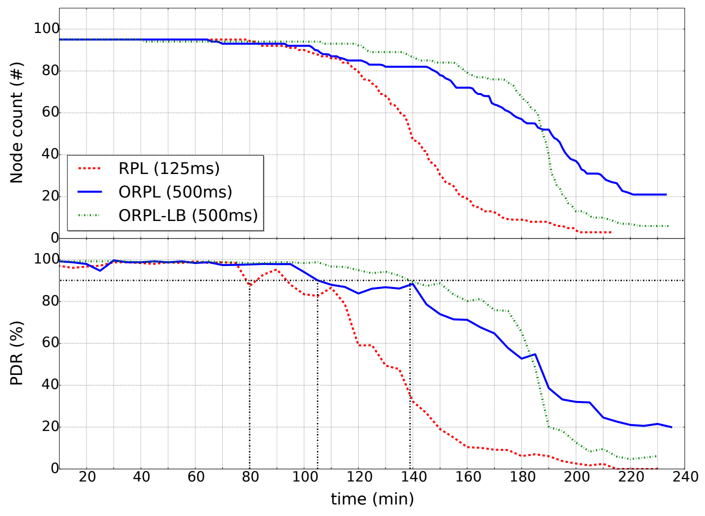

Fig. 10. ORPL-LB keeps a PDR of at least 90% for almost 140 min against ∼105 min for ORPL and ∼80 min for RPL.

reaches a point where many nodes deplete at once, crossing ORPL in the late life of the network. Due to the fact that the energy consumption is better balanced, the nodes tend to fail at the same time. This occurs at a point in time where more than 40% of the nodes are already out.

E. *Finding a Target Duty Cycle at Runtime*

We evaluate our methodology to automatically determine a target duty cycle at runtime (see Subsec IV-C). To evaluate this mechanism, we ran 5 ORPL-LB experiments within the Indriya testbed. Each experiment is a collect-only setup (see Section V.A) and lasts one hour.

The initial duty-cycle target advertised by the sink is $DC_{targetMAX} = 1\%$. This value is selected because it is almost twice the average duty-cycle observed in the same setup using ORPL. We observe that at the end of each experiment the target duty-cycle advertised by the sink has converged to a value which was at least 0.50% and at most 0.60%. This is in-line with the average duty-cycle of 0.57% observed for simulations with ORPL on the same setup (see Table 1). This means that ORPL-LB is able to achieve the same average duty-cycle as ORPL but with the additional benefit that the load is much better balanced than ORPL, leading to an increased network lifetime.

F. *Performance under Heavy Load*

We evaluate the ability of OPRL-LB to handle a heavy traffic load.

Using the high load optimization described in Sec IV-D we compare ORPL-LB to ORPL and RPL in Indriya with data collection at various intervals. For a fair comparison the wake-up interval for RPL has been fixed to 500 ms as for ORPL (see Sec V-C)[3]. The ContikiMAC "burst mode" is used for both RPL and ORPL(-LB). This mode allows a node to send several packets in a row to the same destination when an acknowledgement has been received.

We vary the data collection interval among 1 minute, 30 seconds and 15 seconds. Considering the 93 nodes in the Indriya testbed and the fact that the root does not send any data packets, these intervals result respectively in the following loads at the sink: 1,53 packets/s, 3.06 packets/s and 6.13 packets/s. The results of our experiments are presented in Tab II. For each transmission interval, these results have been averaged on at least four 1-hour runs.

Data tx delay	Protocol	PDR (%)	Lat (s)	DC (%)
60 sec	ORPL	99.65	0.94	0.92
	ORPL-LB	99.76	0.82	0.98
	RPL	96.91	1.75	0.66
30 sec	ORPL	99.3	1.79	1.48
	ORPL-LB	99.77	1.21	1.44
	RPL	96.23	2.27	1.02
15 sec	ORPL	96.48	5.77	3.36
	ORPL-LB	98.39	3.76	3.08
	RPL	89.14	5.24	2.11

TABLE II. HIGH LOAD EXPERIMENT SUMMARY: UNDER DIFFERENT NETWORK LOADS THE LOAD BALANCING MECHANISM OF ORPL-LB ALLOWS TO REACH A BETTER PDR THAN BOTH RPL AND ORPL.

The reduction of the PDR as the load increases is explained by a saturation of the nodes' transmit queues. At high loads, the medium saturates and collisions become more frequent. The high traffic load, combined with the deteriorated medium, put even more pressure on the node's queues, causing packet drop.

ORPL-LB outperforms ORPL and RPL thanks to load balancing and opportunistic routing combined. At the highest load, ORPL-LB increases the PDR by almost 2 percentage points (pp) compared to ORPL, and 9.5 pp compared to RPL. Compared to ORPL this is achieved with a lower latency and a slightly reduced duty cycle. Compared to RPL, the latency is lower but the duty cycle higher, as expected based on the observations made in Sec V-C.

VI. Related work

Load balancing aims at avoiding too great disparity in the energy consumption of the nodes by making the traffic as uniform as possible. Different approaches have been proposed to bring load balancing within WSNs: hierarchical, flat, multipath or tree.

Hierarchical protocols are mainly based on the creation of clusters where the nodes can directly communicate with each other but can only communicate with other clusters via their own cluster head. The formation of a cluster and the selection of the cluster head are the subject of much research [11]. The main limitation of such techniques is their centralized nature, which limits their scalability and applicability.

The flat model approach [12], on the other hand, allows the nodes to make their own routing decision locally. Multipath routing promotes the use of several concurrent paths to send data to the destination [13]. This particularity can be used for

[3]Using the 125 ms wake-up interval would result to RPL being able to reduce the time spent by the packets in the queues compared to ORPL(-LB)

load balancing purposes by splitting the traffic between the available paths [14] but requires more routing information to be maintained. Finally tree-based routing such as CTP [1] and RPL [2] by using different objective functions try to build a well-balanced tree.

LB-RPL [4] is a load-balanced version of RPL that addresses load balancing in a sender-oriented approach. Indeed with LB-RPL each node is responsible to select the best next hop in order to try to maintain a certain balance between the nodes. In LB-RPL, nodes maintain a list of k potential parents, and select one of them on a per-packet basis depending on the parent's load. The objective is to avoid to saturate a node with a high number of children.

This is achieved at the expense of a specific mechanism allowing a child to learn the load of all its followers. The load is quantified based on the buffer utilization of the parents, which is disseminated through RPL DIO control messages. Each node will use a delay proportional to its workload to offset a DIO transmission: the higher the workload, the higher the delay. Once a node has computed the load of its k parents, the two nodes with the lower load are selected and the packets are distributed between them in proportion to their link quality.

BCP [15] is another approach that uses back pressure from the nodes' queue in the path towards the sink. By computing for each neighbor a back-pressure weight based on this queue backlog and the link quality the routing decision can be achieved independently for each packet. The originality of this approach is to intentionally allow to send a packet backward to a node furthest from the destination. This makes BCP particularly efficient in case of high transmission rates. The main drawback of this protocol is very poor energy performance and the fact that, due to the use of a LIFO queue, some packets can remain in the queues for arbitrarily long times.

Another original proposal for reducing energy consumption is the Broadcast-Free Collection Protocol [3]. BCF offers to remove the broadcast transmissions in order to reduce the energy consumption of the nodes. To build a topology without broadcast BFC forces each node to eavesdrop on the unicast transmission within its range to select a parent with a reliable path to the sink. The main drawback of this broadcast free methodology using only unicast is to benefit essentially to the sink's neighbors and the leaves. While the unicast transmissions of the sink's neighbors are cheap due to the fact the always-on sink can directly acknowledged them, the (broadcast-free) traffic of the leaves is restricted solely to send their own packets given they do not relay anything. The restricted impact of BFC on the relay nodes makes this protocol unable to efficiently addresses the energy hotspots issue.

ORPL-LB differs with the above approaches by being the first to exploit opportunistic routing for load-balancing. Dynamic wake-up interval combined with opportunistic routing gives an enhanced control on the traffic load and resulting energy consumption. Moreover this approach does not require any load info propagation towards the children.

VII. Conclusion

We presented ORPL-LB, a load-balanced extension of ORPL. By allowing each node to adapt its wake-up interval based on a network-wide objective, ORPL-LB allows to reduce the energy consumption of the hotspots. This leads to an enhancement of the network lifetime.

Our testbed experiments have shown that ORPL-LB reduces by almost 40% the energy consumption for the hotspots. This is achieved with no impact (or a negligible one) on the performance in terms of reliability, energy consumption and latency. Considering the network lifetime as the time during which the network is able to maintain a high PDR (at least 90%), ORPL-LB is then able to extend the network lifetime by on average 33% compared to ORPL.

References

[1] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Lewis, "Collection Tree Protocol," in *Sensys' 09, Berkeley, USA*, November 2009.

[2] T. Winter *et al.*, "RPL: Ipv6 routing protocol for low-power and lossy networks," in *Internet Engineering Task Force, RFC 6550*, March 2012.

[3] D. Puccinelli, M. Zuniga, S. Giordano, and P. J. Marrn, "Broadcast-Free Collection protocol," in *SenSys'12, Toronto, ON, Canada*, November 2013.

[4] X. Liu, J. Guo, G. Bhatti, P. Orlik, and K. Parsons, "Load balanced routing for low power and lossy networks," in *WCNC' 13, Shangai, China*, April 2013.

[5] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: Scalable opportunistic routing with ORPL," in *Sensys' 13, Rome, Italy*, November 2013.

[6] M. Doddavenkatappa, M. C. Chan, and A. Amanda, "A low-cost, 3D wireless sensor network testbed," in *TridentCom*, 2011.

[7] C. Cano *et al.*, "Low energy operation in wsns: A survey of preamble sampling mac protocols," in *Comput. Netw, doi:10.1016/j.comnet.2011.06.022*, 2011.

[8] A. Dunkels, "The contikimac radio duty cycling protocol," in *SICS Technical Report T2011:13*, 2011.

[9] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *ISPN'12, Beijing, China*, April 2012.

[10] "ns-2, http://nsnam.isi.edu/nsnam/index.php/MainPage."

[11] D. W. abd B. V. Thakur, "Load balancing algorithms in wireless sensor network: A survey," in *IJCNWC ISSN:2250-3501 Vol2, No4*, August 2012.

[12] Q. Jiang and D.Manivannan, "Routing protocols for sensor networks," in *Consumer Communication and Networking Conference CCNC, First IEEE pp93-98*, 2004.

[13] J. Gallardo, A. Gonzalez, L. Villasenor-Gonzalez, and J. Sanchez, "Multipath routing using generalized load sharing for wireless sensor networks," in *Proceeding of the International Conferences on Wireless and Optical Communications, Montreal, QC, Canada*, 2007.

[14] E. Jones, M. Karsten, and P. Ward, "Multipath load balancing in multi-hop wireless networks," in *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 05), Montreal, QC, Canada*, 2005.

[15] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The Backpressure Collection Protocol," in *IPSN 2010*, 2010.