A Wave-Based Request-Response Protocol for Latency Minimization in WSNs

Riccardo Monica, *Member, IEEE*, Luca Davoli, *Member, IEEE*, and Gianluigi Ferrari, *Senior Member, IEEE*

***Abstract*—Transmission latency is a key performance metrics in most wireless sensor network (WSN) applications. Nodes in a WSN often keep their radio transceivers off, and turn them on periodically using a duty cycling mechanism. The latter is a major source of delay in the network, because transmissions must wait for the next receiver wake-up. In this paper, we present a cross-layer approach to minimize latency of a request-response (RR) protocol adopted in an IEEE 802.15.4-based WSN where the IPv6 routing protocol for low-power and lossy networks (RPLs) is used. Extra wake-ups are generated dynamically to match the predicted arrival time of the response packet, in order to reduce the duty cycling delay. The proposed approach is verified with the Cooja simulator, relying on the Contiki operating system (OS). The observed experimental results show a shorter RR delay with respect to a phase alignment (PA) approach.**

***Index Terms*—Network latency, request-response (RR) protocol, routing protocol for low-power and lossy network (RPL), wireless sensor networks (WSNs).**

I. Introduction

ENERGY efficiency and latency are key performance metrics in most wireless sensor network (WSN) applications. The radio transceiver is one of the components with the highest power consumption on a low-power wireless sensor node. Therefore, nodes of a WSN often keep their radio transceivers off as much as possible, to prolong their batteries' lifetimes. Since a node cannot receive any data when the transceiver is turned off, a duty cycling mechanism must be used at the medium access control (MAC) layer, to periodically turn the radio on. Hence, duty cycling is a major source of delay, because packets must wait at the sender node for the receiver's wake-up before they can be sent.

The IPv6 routing protocol for low-power and lossy networks (RPLs) [1] has been proposed to provide IP(v6) connectivity on low-power radios. RPL leads to a tree-like network topology, anchored at a sink node. In a typical WSN, nodes proactively send periodic sensor reading updates to the sink, complying with an update frequency that is limited by power constraints and network throughput. However, specific applications require sensor nodes to be queried aperiodically, using a request-response (RR) protocol, when sensor data are urgently needed in response to external unpredictable events (e.g., alarms, user interaction, etc.). Hence, the query should be answered as quickly as possible, to increase the responsiveness.

The main contribution of this paper is a cross-layer approach for delay optimization of an RR protocol. The approach requires the transmission of a single pair of IP packets: a request by the sink node and the response by the sensor node. The duty cycling delay is reduced by making the nodes wake up as if they were "hit" by (upward and downward) "waves," i.e., sequentially according to their depths. Namely, we configure the nodes wake-up phases so that: a wave of wake-ups carries downward the request packet from the sink to the required sensor node with a minimal delay; and a dynamic second wave is generated symmetrically for the response packet moving upward in the network toward the sink. The proposed approach is tested with the Contiki operating system (OS) in the Cooja simulator [2], showing a shorter RR delay with respect to a simpler phase alignment (PA) approach, as the one used in the RAWMAC low-latency harvesting protocol [3] and based on the use of static waves. Our analysis highlights interesting tradeoffs between RR delay and energy consumption.

The rest of this paper is organized as follows. In Section II, an overview of related work on duty cycling mechanisms is provided. Section III briefly introduces ContikiMAC [4], RPL, and RAWMAC protocols. The proposed approach is described in Section IV, while experimental results are shown in Section V. Finally, in Section VI we draw our conclusions.

II. Related Work

In the literature, it is possible to identify the following two main categories in radio duty cycling mechanisms: 1) *synchronous* mechanisms, requiring a complete synchronization between neighboring nodes (e.g., T-MAC [5]) and 2) *asynchronous* mechanisms, not depending on any *a-priori* synchronization and to be further subdivided into sender-initiated (e.g., B-MAC [6]) and receiver-initiated (e.g., BladeMAC [7]). As illustrative examples, with the WiseMAC protocol [8] nodes learn the wake-up phase of each other through a phase-lock optimization; in the X-MAC protocol [9], the sender wakes up the receiver using short preambles. The CXMAC protocol is a simplified implementation of X-MAC, where a mote periodically sends a short probe; when a potential receiver wakes up and gets such a probe with its own address, it replies

Manuscript received October 15, 2018; revised February 20, 2019 and April 8, 2019; accepted April 25, 2019. Date of publication May 2, 2019; date of current version October 8, 2019. The work of L. Davoli and G. Ferrari was supported in part by the European Commission H2020 Framework Program through AFarCloud Project "Aggregate Farming in the Cloud" under Grant 783221 and in part by the "Iniziative di Sostegno alla Ricerca di Ateneo" Program of the University of Parma through "Multi-Interface IoT Systems for Multi-Layer Information Processing" Project. *(Corresponding author: Luca Davoli.)*

The authors are with the Department of Engineering and Architecture, University of Parma, 43124 Parma, Italy, (e-mail: riccardo.monica@unipr.it; luca.davoli@unipr.it; gianluigi.ferrari@unipr.it).

Digital Object Identifier 10.1109/JIOT.2019.2914578

with an ACK [10], [11]. In [12], a data aggregation approach for WSNs, based on the joint generation of a conflict-free schedule and an aggregation tree, is proposed. In [13], the latency minimization problem is supported in a cloud-oriented way with an aggregation scheme centered at the cluster head. Finally, a joint duty-cycle optimization control is proposed in [14], trying to define a global power management strategy suitable for WSNs, especially in industrial scenarios. The derivation of traffic latency minimization approaches in heterogeneous environments represents an interesting research activity which goes beyond the scope of this paper and will be the subject of future works.

The approach proposed in this paper relies on the RPL protocol, which is recalled in Section III-B. The opportunistic routing in wireless sensor networks (ORWs) [15] is a routing mechanism that forwards packets to the first awaken neighbor offering routing progress toward the destination node. In [16], an RPL routing metric for delay minimization is presented. The routing over low power and lossy networks (ROLL) Working Group [17] has specified OF0 [18], in which the only routing metric adopted refers to the hop count. The minimum rank with hysteresis objective function (MRHOF) [19] minimizes metrics that are additive along a route, and uses hysteresis to reduce churn in response to small metric changes. Other existing approaches rely on: predetermined scheduling mechanisms, such as WirelessHART [20] or time-slotted channel hopping (TSCH) [21]; periodic and asynchronous wake-up mechanisms, such as low power listening (LPL) [22] or low power probing (LPP) [23]; dynamic slot allocation schemes, such as GoMacH [24], ZMAC [25], and iQueue-MAC [26].

III. ContikiMAC, RPL, AND RAWMAC

The proposed approach is based on ContikiMAC, RPL, and RAWMAC [3]. A short overview of these protocols is presented in the following.

A. ContikiMAC

ContikiMAC is an asynchronous and sender-initiated radio duty cycling protocol. The nodes periodically wake up to check for possible incoming packet transmissions. The period between wake-ups, defined as cycle time, is denoted as C_T. All nodes in a network have the same cycle time, but relative wake-up phases between pairs of nodes are essentially random, as they depend on the power-on instants of the nodes. A sender repeatedly transmits its packet until it receives a link-layer acknowledgment (ACK) from a receiver. This implies that the packet is repeated, in the worst case, for an entire cycle time, to ensure that the receiver awakes at least once. When the receiver detects a packet transmission during a wake-up, it keeps its radio transceiver on to receive the entire packet. Then, if the receiver is the recipient of the packet, an ACK packet is sent back to the sender. In order to reduce energy consumption and radio channel occupancy, ContikiMAC introduces a phase-lock mechanism. By recording, for each neighboring node, the last instant an ACK was received, a node can estimate the wake-up time of each neighbor, assuming a constant wake-up period C_T. Then, a transmission is started a small guard time P_g (dimension: ms) before the estimated receiver's wake-up.

B. RPL Routing Protocol

RPL is a distance-vector routing protocol based on the organization of the nodes in a tree-like network topology, referred to as destination-oriented directed acyclic graph (DODAG). The tree is anchored at a node, denoted as DAG root, and the cost of each path is evaluated according to metrics defined in an objective function (OF). Each node selects a parent node, which is the neighboring node with the shortest path to the root node. Thus, unicast communications between any node and the DAG root are optimized according to the metric. The current RPL implementation for the Contiki OS adopts, as default, the expected transmission count (ETX) metric [27], which tries to minimize the average number of packet transmissions required in order to deliver a packet to the ultimate destination. Nevertheless, other solutions can be adopted, leading, for example, to paths with the smallest number of poor quality links or of intermediate hops. RPL uses two types of control messages to maintain the topology: 1) a node broadcasts DODAG information objects (DIOs) to inform nearby nodes about its distance to the DAG root and 2) destination advertisement objects (DAOs) are unicast messages sent to the selected parent, used to populate the routing tables of ancestor nodes in the DODAG. Moreover, RPL uses a trickle mechanism to reduce the transmission of redundant DIO messages when the network is stable.

C. RAWMAC

RAWMAC is an adaptation layer in which RPL configures the ContikiMAC wake-up phase. In detail, in a WSN where nodes send data to the root node, RAWMAC reduces the delay between the instants of packet creation at the sensor node and reception at the root node. A node with RAWMAC exploits ContikiMAC phase discovery mechanism to align its phase so that it wakes up right before its preferred parent. When a node receives, at its own wake-up time, a packet to be routed upward, the parent wake up is about to occur. Therefore, a data propagation wave is created, from the leaves of the DODAG to the root. Packets traveling along this wave can reach the root node with minimal latency.

IV. PROPOSED APPROACH

We consider a generic RR protocol in a network organized as an RPL DODAG, in which a *request* IP(v6) packet is sent by the DAG root node downward, toward a target node in the network. After a known processing time, the target node sends back a *response* upward to the DAG root. Our main goal is to minimize the delay between the generation of the request and the reception of the response at the DAG root.

To minimize the *downward delay* from the root node to the target, the nodes align their radio wake-up phases so that they wake up in downward sequence, i.e., each node wakes up right after its parent in the RPL DODAG. Therefore, nodes which relay packets from parent to child have to wait only a short time for the wake-up of the next-hop node. The downward PA approach is detailed in Section IV-A.

To minimize the *upward delay* of the response from the target node to the DAG root, intermediate nodes traversed by the request schedule an extra wake-up when the response is going to traverse them toward the DAG root. Therefore, nodes

which relay the response have to wait only a short time for the wake-up of the parent node. The upward response transmission optimization is described in Section IV-B.

In Section IV-C, the proposed protocol is integrated with the RAWMAC low-latency harvesting protocol [3]. Both approaches in Sections IV-A and IV-B introduce new types of wake-ups, which may interfere with the ContikiMAC phase-lock mechanism: this issue is addressed in Section IV-D.

A. Wake-Up Phase Alignment

Nodes can, in principle, generate packets at any time, but a node can receive packets to be routed only at a wake-up. To reduce the downward delay for packet relay, each node shifts its wake-up phase so that it is aligned with that of its parent. A positive time offset P_o (dimension: ms) is added to the phase, in order to account for the time necessary to receive the packet and send it to the next node. The offset P_o should be chosen carefully: if it is too short, the packet may "miss" the wake-up of the next node and, then, the packet needs to wait a whole cycle time for the next wake-up; if it is too long, the packet waits uselessly and the transmission delay increases.

When a node C receives an ACK from its parent B at time $t_{C,B}$, it changes its own wake-up phase ϕ_C to the following ϕ'_C:

$$\phi'_C = \left(t_{C,B} + P_o\right) \bmod C_T \tag{1}$$

where the modulus operator $(a \bmod b) \doteq a - \lfloor a/b \rfloor b$. To prevent frequent updates due to small inaccuracies in the measurement of $t_{C,B}$, the phase is updated only if changed by more than a (properly chosen, as will be shown in Section V-A) threshold ΔP_o, i.e., if $|\phi'_C - \phi_C| > \Delta P_o$.

After the PA has completed, the network behavior for the RR protocol is shown in Fig. 1, in which, on the left, the request packet transmission from the root node to a target node is carried out. Nodes seem to wake up in succession when hit by waves, which carry packets which quickly reach the destination. The packet may be created at any time with a uniform distribution, so it waits at the root (on average) a time interval equal to $C_T/2$ for the next wave. The root starts transmitting a little earlier (by the guard time P_g) than the predicted wake-up time. Then, every intermediate hop adds P_o to the delay. Finally, the reception time at the last node is denoted as P_ℓ. The total downward average delay can thus be (theoretically) written as

$$D_d(h) = C_T/2 + P_g + (h-1)\,P_o + P_\ell \tag{2}$$

where h is the depth of the target node, i.e., the number of hops from the DAG root.

The response packet is transmitted back to the root node as illustrated in the right part of Fig. 1. A request is indeed able to quickly reach (propagating downward) the target node. However, in the upward direction, the response packet has to wait $C_T - P_o$, at each intermediate node, for the next wake-up of the parent node. Therefore, the theoretical average RR delay at the DAG route with downward PA is

$$\begin{aligned} D_r(h) &= D_d + h(C_T - P_o) \\ &= C_T/2 + (h-1)\,C_T + (C_T - P_o) + P_g + P_\ell. \end{aligned} \tag{3}$$

For sufficiently large values of h, D_r is almost independent of P_o. In particular, a similar result would be obtained in

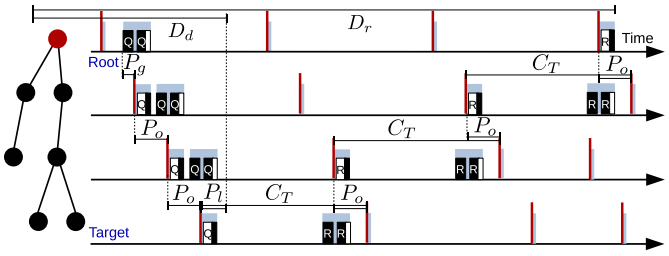

Fig. 1. Timeline of a request packet Q (left) and the corresponding response R (right) after PA. Red vertical lines represent wake-ups, while light blue background rectangles highlight where the node's radio is on. Packet transmission and reception are displayed in black and white, respectively.

RAWMAC, where P_o is chosen differently to optimize the upward wave instead of the downward wave.

A correction term $D_{\text{coll}}(h)$ may be added to the delay expression in (3) to account for collisions. In the CSMA MAC layer of Contiki OS: after the first collision, the first retransmission attempt is performed after a back-off time uniformly distributed within $[C_T, 5C_T]$ (mean: $3C_T$); after a second collision, the second retransmission is attempted after a back-off time uniformly distributed within $[C_T, 9C_T]$ (mean: $5C_T$); and after the third collision, a final retransmission is attempted after waiting a back-off time uniformly distributed within $[C_T, 13C_T]$ (mean: $7C_T$). After a fourth collision, the packet is dropped [28]. Therefore, the average collision delay $D_{c,\text{hop}}$ over a single hop is

$$D_{c,\text{hop}} = \begin{cases} 0, & \text{if 0 collisions} \\ 3C_T, & \text{if 1 collision} \\ (3+5)C_T, & \text{if 2 collisions} \\ (3+5+7)C_T, & \text{if 3 collisions.} \end{cases} \tag{4}$$

Hence, indicating the packet collision probability as p_c, the probabilities of exactly zero, one, two, and three collisions can be evaluated as follows:

$$\begin{aligned} p_{c,0} &= (1-p_c)/p_{c,\text{tot}} \\ p_{c,1} &= p_c(1-p_c)/p_{c,\text{tot}} \\ p_{c,2} &= p_c^2(1-p_c)/p_{c,\text{tot}} \\ p_{c,3} &= p_c^3(1-p_c)/p_{c,\text{tot}} \end{aligned} \tag{5}$$

where $p_{c,\text{tot}} = (1-p_c^4)$ is a normalization coefficient.[1] Hence, from (4) and (5), the average supplementary collision delay for a single hop can be computed as follows:

$$D'_{c,\text{hop}} = 0\,p_{c,0} + 3C_T\,p_{c,1} + 8C_T\,p_{c,2} + 15C_T\,p_{c,3}. \tag{6}$$

Therefore, since the delay $D'_{c,\text{hop}}$ affects each hop upward and downward, the total $D_{\text{coll}}(h)$ for a target node at depth h can be expressed as

$$\begin{aligned} D_{\text{coll}}(h) &= 2h\,D'_{c,\text{hop}} \\ &= 2h\,(1-p_c)\,\frac{3C_T\,p_c + 8C_T\,p_c^2 + 15C_T\,p_c^3}{\left(1-p_c^4\right)}. \end{aligned} \tag{7}$$

Finally, since typically the packet collision probability $p_c \ll 1$, it can be concluded that $(1-p_c^4) \simeq 1$, $p_{c,2} \ll p_{c,1}$,

[1] Note that since a packet is dropped at the fourth collision, the supplementary collision delay in (5) is computed considering only the cases in which eventually a packet goes through.

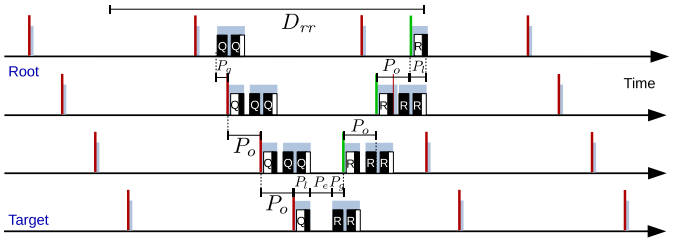

Fig. 2. Timeline of a network with PA and response optimization. A request Q is sent from the root node and a response R is received. Standard wake-ups are represented by red vertical lines. Green lines represent extra wake-ups.

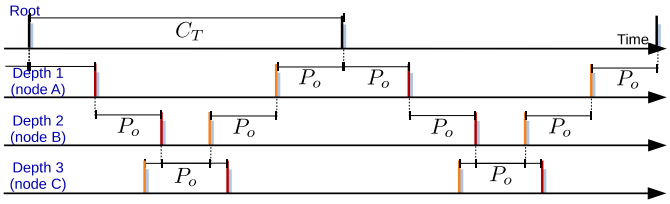

Fig. 3. Wake-ups of a network where both PA and RAWMAC are in effect. Root node wake-ups are represented by black vertical lines. PA and RAWMAC wake-ups are represented by red and orange lines, respectively.

and $p_{c,3} \ll p_{c,1}$. This allows to approximate $D_{\text{coll}}(h)$ as follows:

$$D_{\text{coll}}(h) \simeq 2h\, 3C_T\, p_c. \tag{8}$$

B. Response Delay Optimization

The upward delay experienced by the response packet is minimized by creating a second wave of extra wake-ups, denoted as response wave (RW), from target node to root node. This second wave is created on-demand, upon detection of a request packet. Packets which belong to an RR protocol are marked in the differentiated services (DSs) field (6 bits) of the IPv6 header. We define the "Request" DS code point as 000001 (request packets) and the "Response" DS code point as 000011 (response packets). These two code points should not interfere with any other protocol, because DS values with the least significant bit set to 1 are unassigned (i.e., free to use) according to RFC 2474 [29].

Whenever a node routes a request message, it schedules an RW wake-up at the predicted response arrival time. The desired behavior is shown in Fig. 2. Each hop adds a delay time P_o for the request and a further delay P_o for the response. An additional short fixed time P_e is due to response processing at the target node. Therefore, given the forwarding of a request packet in correspondence to a wake-up at time t_Q, the response packet is predicted to arrive at the following instant t_R:

$$t_R(r) = t_Q + 2\,P_o(r-1) + P_o + P_g + P_e + P_\ell \tag{9}$$

where r is the number of hops (in the DODAG) between the current node and the target node. The RPL implementation was modified so that nodes include the hop count r for each descendant node in the DAO messages. A special case occurs for the DAG root: if it generates a packet too close to the next wake-up, it must wait a whole cycle time C_T to send it. A response packet at the DAG root node relative to a request generated at time t_G is predicted to arrive at the following instant:

$$t'_R(r) = \begin{cases} t_W + t_R(r) - P_o, & \text{if } t_W - t_G > P_g \\ t_W + t_R(r) + C_T - P_o, & \text{otherwise} \end{cases} \tag{10}$$

where t_W is the time of the next wake-up of the first-level child node and r corresponds, in this case, to the depth of the target node.

If a node does not receive a response packet at the scheduled RW wake-up, it assumes that packet transmission failed and that the packet is going to be retransmitted later. The RW wake-up is repeated W_R times, every C_T, attempting to create another wave for the retransmission; after W_R unsuccessful attempts, the response packet is declared lost.

We remark that when a request packet is routed downward: 1) the next hop and the hop count r to the target node are found in the RPL database and 2) the radio duty cycle (RDC) layer predicts the response time using (9) and schedules an RW wake-up. Whenever a response packet is routed upward: 1) the RDC layer is informed so that no more RW wake-ups are generated and 2) phase-lock is ignored, so that the packet is sent immediately by the MAC layer to take advantage of the RW wake-up.

The total upward average delay is thus (theoretically) equal to

$$D_u(h) = P_g + (h-1)\,P_o + P_\ell. \tag{11}$$

The average RR delay D_{rr} is computed by adding the downward delay D_d (2), the upward delay D_u (11), and the processing time P_e, obtaining

$$\begin{aligned} D_{rr}(h) &= D_d(h) + P_e + D_u(h) \\ &= C_T/2 + 2\,(h-1)\,P_o + 2P_g + 2P_\ell + P_e. \end{aligned} \tag{12}$$

The delay D_{rr} is expected to be lower than D_r (3), because, in general, $2P_o < C_T$.

C. Integration With RAWMAC

Unlike the PA procedure proposed in Section IV-A, the RAWMAC protocol aligns the node wake-ups to minimize the upward delay to the root node [3]. In the following, we integrate RAWMAC and PA by performing two wake-ups at each duty cycle. The PA wake-up is scheduled P_o *after* the parent PA wake-up, while the RAWMAC wake-up is scheduled P_o *before* the parent RAWMAC wake-up. The root node wakes up only once for each duty cycle, covering both RAWMAC and PA wake-ups (as shown in Fig. 3).

Given the PA wake-up phase ϕ_B^{PA} of a node B, the RAWMAC wake-up phase ϕ_B^{RAW} is

$$\phi_B^{\text{RAW}} = \left(\phi_B^{\text{PA}} - 2\,P_o\,h_B\right) \bmod C_T \tag{13}$$

where h_B is the depth of node B in the RPL tree. When a node B receives from parent A an ACK due to the PA wake-up at time $t_{B,A}^{\text{PA}}$, it sets its own wake-up phases as follows:

$$\begin{aligned} \phi_B^{\text{PA}} &= \left(t_{B,A}^{\text{PA}} + P_o\right) \bmod C_T \\ \phi_B^{\text{RAW}} &= \left(t_{B,A}^{\text{PA}} + P_o - 2\,P_o\,h_B\right) \bmod C_T. \end{aligned} \tag{14}$$

Conversely, when node B receives from parent A an ACK due to the RAWMAC wake-up at time $t_{B,A}^{\text{RAW}}$, it sets its own

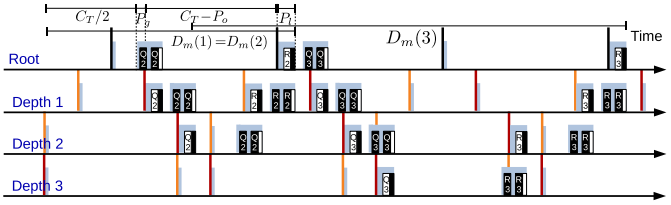

Fig. 4. Two requests are sent by the DAG root to nodes at depths 2 (left) and 3 (right), respectively. The first response follows the next upward wave and reaches the root at its next wake-up. The second response misses it and follows the subsequent one.

wake-up phases as follows:

$$\phi_B^{\mathrm{RAW}} = \left(t_{B,A}^{\mathrm{RAW}} - P_o\right) \bmod C_T$$
$$\phi_B^{\mathrm{PA}} = \left(t_{B,A}^{\mathrm{RAW}} - P_o + 2\,P_o\,h_B\right) \bmod C_T. \quad (15)$$

The method to differentiate between RAWMAC ACKs and PA ACKs is presented in Section IV-D.

According to the phase-lock mechanism of ContikiMAC (Section III-A), a node B sends a packet at the next wake-up of a neighbor node C. When two wake-ups are performed at each duty cycle, two phases are estimated by node B: ϕ_C^{RAW} and ϕ_C^{PA}. Upon receiving a PA ACK from node C, node B computes ϕ_C^{RAW} using (13). The specular equation computes ϕ_C^{PA} given ϕ_C^{RAW}. We remark that (13) depends on h_C: since in an RPL DODAG a node communicates only to its parent and children, h_C can be inferred as

$$h_C = \begin{cases} h_B - 1, & \text{if } C \text{ is } B\text{'s preferred parent} \\ h_B + 1, & \text{otherwise.} \end{cases} \quad (16)$$

Fig. 4 illustrates the behavior of an RR protocol when both RAWMAC and PA are used. Requests follow a downward wave generated by PA and responses follow an upward wave generated by RAWMAC. For small values of h, the average delay is constant and equals to $D_m(1) = C_T/2 + P_g + P_\ell + C_T - P_o$. The delay does not depend on the hop count as long as the response can catch the next upward wave. Each hop requires a time offset P_o for the request and a time offset P_o for the response, so that the accumulated multihop delay is

$$D_a(h) = 2\,P_o(h-1) + P_o + P_g + P_\ell + P_e. \quad (17)$$

A packet misses an upward wave whenever the accumulated hop delay $D_a(h)$ exceeds a multiple of a cycle time C_T. Taking into account (17), whenever the depth h is such that

$$h > 1 + \left(nC_T - P_o - P_g - P_\ell - P_e\right)/2P_o \quad (18)$$

with $n \in \{1, 2, 3, \ldots\}$, then the delay D_m increases by nC_T. Hence, the theoretical RR delay D_m increases in discrete steps as follows:

$$D_m(h) = \frac{C_T}{2} + P_g + P_\ell + C_T - P_o + C_T \left\lfloor \frac{D_a(h)}{C_T} \right\rfloor. \quad (19)$$

In general, the delay $D_m(h)$ is not shorter than the RW delay $D_{rr}(h)$ (12). However, the RW approach is based on the assumption that requests are sent by the DAG root and responses are sent by other nodes in the network, as the upward wave is created only on-demand. Conversely, as both upward and downward waves are always active in RAWMAC, requests from nodes to the DAG root and requests from DAG root to the nodes are optimized simultaneously.

D. Link-Layer Acknowledgment Modifications

The approaches proposed in Sections IV-B and IV-C add extra wake-ups to the standard duty cycle: this interferes with the ContikiMAC phase-lock mechanism. In addition to wasted energy and delayed/failed transmissions, incorrect parent phase estimation may cause a phase change in the whole RPL DODAG subtree due to wake-up PA (Section IV-A). Therefore, we modify the link-layer ACK packet so that the type of generating wake-up can be inferred from it. Three kinds of ACKs are possible: 1) a standard RDC ACK, due to a PA wake-up; 2) an RW ACK, caused by the RR extra wake-ups; and 3) an RAWMAC ACK, caused by RAWMAC wake-ups. Phase update should not be performed in 2), because RW wake-ups do not depend on the node phase. In cases 1) and 3), the node receiving the ACK estimates the sender phase as described in Section IV-C.

According to the IEEE 802.15.4 standard [30, Sec. 7.3.3], the ACK packet (Imm-Ack) is composed of 5 bytes at the MAC layer: the first 2 bytes hold a bitmask (Frame Control Flags), where bit 7 is unused and bit 5 (Ack Request) is meaningless for Imm-Ack packets. We exploit these bits to differentiate between standard PA, RW, and RAWMAC ACKs, setting these two bits as follows: $\{b_5, b_7\} = \{0, 0\}$ for PA; $\{0, 1\}$ for RAWMAC; and $\{1, 1\}$ for RW.

V. Experimental Evaluation

A. Experimental Setup

The proposed approach is implemented in Contiki OS, v. 2.6, carrying out the tests via Cooja, a Java-based simulator for Contiki-based wireless sensor nodes [2]. The radio transmission range is set to 50 m, with 100% transmission success rate. Nodes at distances between 50 and 100 m from the transmission source are set in the "interference" range, meaning that the received data cannot be decoded.

The test network, shown in Fig. 5, comprises 11 nodes and self-organizes into an RPL DODAG using the standard ETX RPL metric. Node 11 has been configured as the DAG root. The remaining ten nodes act as servers and are queried by the root. A UDP "echo" RR protocol has been implemented at the server nodes, with UDP packets sent by the root to each server node. The UDP payload consists of 15 bytes and contains a unique request identifier. Upon reception of a UDP packet destined to itself, a server node sends a response packet containing the same payload to the root node. If the request packet contains the Request code point in the IPv6 DS field, the Response code point is set in the response packet. The delay is computed as the time difference between the instant of request generation and the reception of the corresponding response at the transport layer of the root node.

For a more accurate timing, most nodes generate ACK packets in hardware, before handling the received packet to the operating system (Contiki OS). Hence, by default, the node firmware cannot change flags in the ACK packet, which is required by our approach (as highlighted in Section IV-D). Therefore, we modified the Cooja hardware simulator, so that Frame Control Flags could be set or unset by manipulating an appropriate control register.

The management of extra wake-ups does not require a significant algorithmic complexity. Each node maintains an

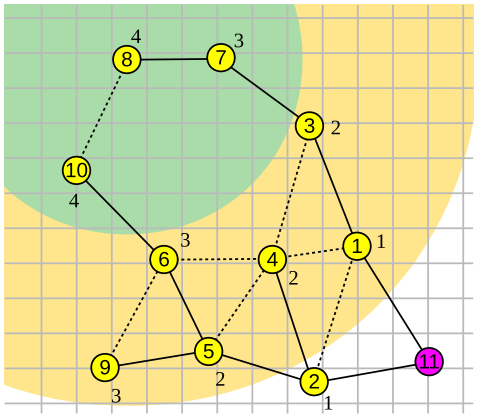

Fig. 5. RPL-based network used in the experimental evaluation. Nodes are represented by small circles, with node ID displayed inside each circle. Node 11 is the DAG root. Grid spacing is 10 m. Solid or dashed segments connect nodes within radio transmission range of each other. Solid segments represent the links chosen by RPL to form the DODAG. Numbers near the nodes indicate the depth h in the RPL DODAG. For illustration purposes, transmission and interference range of node 8 are displayed with two concentric circles, green and yellow, respectively.

TABLE I
FIXED PARAMETERS

Parameter	Value	Description
P_g	16.2 ms	ContikiMAC guard time
P_ℓ	7.0 ms	Packet reception time
P_e	10.0 ms	Response computation time
P_o	35.7 ms	Phase offset
ΔP_o	8.0 ms	Phase update threshold
W_R	10	RW wake-up attempts

TABLE II
EXPERIMENTAL CONFIGURATIONS

Experiment	PA	C_T [s]	RAWMAC	RW
A	Yes	0.125	No	Yes
B	Yes	0.125	No	No
C	Yes	0.250	Yes	Yes
D	Yes	0.250	Yes	No
E	Yes	0.250	No	Yes
F	Yes	0.250	No	No

ordered list of the extra wake-ups which are currently active. In our implementation, an entry in the list contains information on the phase offset of the wake-up (2 bytes), the number of cycle times before the first wake-up is performed (1 byte), the number of wake-ups yet to be performed down from W_R (1 byte), and the IPv6 address of the target node (16 bytes). Hence, for each RR simultaneously active in the network, 20 bytes are needed in the memory of each traversed node.

Six experiments have been carried out to evaluate the behavior of the proposed protocols. The fixed parameters are shown in Table I. In detail, the value chosen for P_g corresponds to the default guard time of ContikiMAC for Sky nodes [31], computed as $10 \cdot$ `CHECK_TIME` + `CHECK_TIME_TX`, i.e., $10 \cdot 2(t_c + t_r) + 6(t_c + t_r)$, where t_c is the interval between consecutive clear channel assessments (CCAs), and t_r is the time for each CCA, as defined in [4], with $t_c = 1/2000$ s and $t_r = 1/8192$ s. Packet reception time P_l corresponds to the estimated time required at the physical layer to receive and acknowledge a packet, which is limited to a 127-byte maximum payload [30]. The response computation

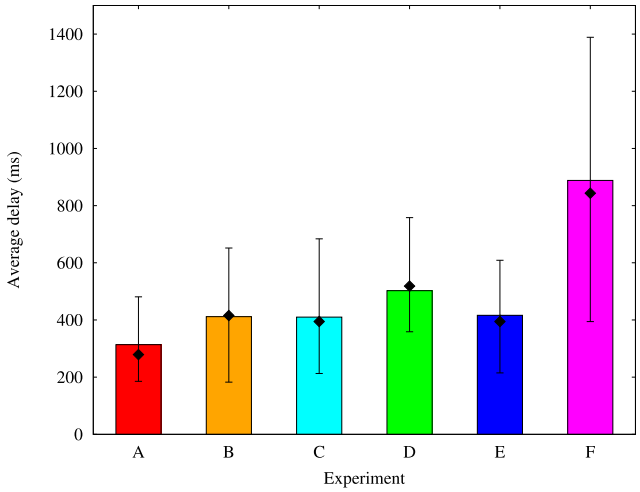

Fig. 6. Histogram bars show the average RR delay in the whole network in each experiment. Error bars high and low limits are set to the average delay of the two target nodes with highest and lowest average delay, respectively. The diamonds mark the theoretical average delay, with reference to parameters in Tables I and II.

time P_e depends on the application layer and has been measured empirically for our echo protocol. The phase offset P_o and the phase update threshold ΔP_o have been set according to [3], where the best P_o has been estimated as around 35 ms ([3, Fig. 6(a)]) and the best ΔP_o in between $7 \div 9$ ms ([3, Fig. 7]).

In each experiment, the root first waits 1 min for the formation of the RPL DODAG. Then, the root node sends 250 requests, subsequently, to each of the ten server nodes, for a total of 2500 requests. A request is sent every interval $T_{\text{RR}} = 4$ s plus an additional random time uniformly distributed between 0 s and 1 s. If the response does not reach the root node within 5 s after the generation of the request, the response is considered lost and is not included in the results. Overall, about 1% of the responses were lost. The nodes' configurations, for each experiment, are summarized in Table II. As can be seen from the results, PA is always effective in minimizing downward delay. The cycle time was set to: 0.125 ms (8 Hz) in experiments A and B; and 0.250 ms (4 Hz) in experiments C, D, E, and F. Cycle times within this range have been used in the original design of ContikiMAC [4] and in previous works [3], [10], [16], [28], [31]. In a real application, the cycle time C_T depends on several factors (e.g., on the available energy to be consumed) and should be chosen depending on the specific application requirements: a long cycle time implies less wake-ups, hence it usually reduces resources usage but increases delay. In experiments C and D, PA for downward delay minimization has been integrated with RAWMAC wake-ups for upward delay minimization (Section IV-C). RW (Section IV-B) is used in experiments A and E. In experiment C, both RAWMAC and RW have been used simultaneously: an extra upward wave is created by RW, when the response arrival is predicted, in addition to the upward waves already generated by RAWMAC at fixed intervals.

B. Delay Evaluation

In Fig. 6, the average RR delay for each experiment is shown, together with the associated range between

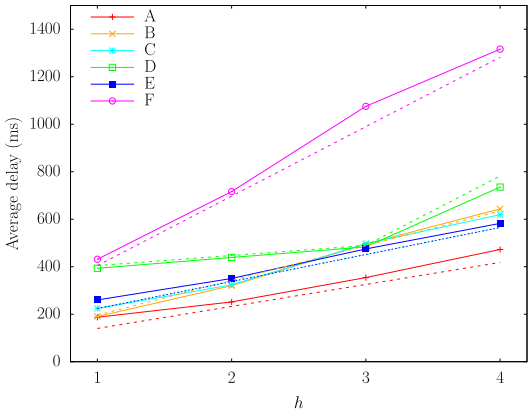

Fig. 7. Average delay by node depth in the DODAG, for each experiment (continuous lines) and predicted by the theoretical models including the supplementary collision delay given by (7) (dashed lines).

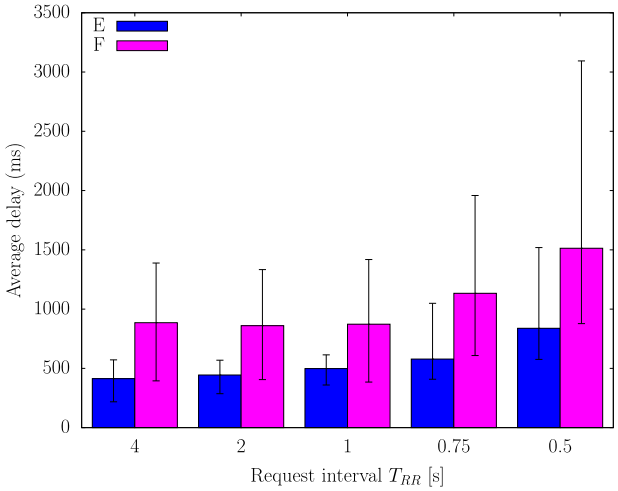

Fig. 8. Average RR delay in the network, for various values of T_{RR}. Error bars' high and low limits are set to the average delays of the two target nodes with highest and lowest average delays, respectively.

minimum and maximum delays. As expected, the RW upward optimization protocol lowers the average delay, with respect to the same configuration without the use of RW. The delay reduction is more evident for experiment E (about 53% lower with respect to experiment F) than for A (about 24% lower with respect to B). The delay also decreases by about 43% when introducing RAWMAC wake-ups (D) in a network where PA is used (F), but the delay is higher than the one obtained with the RW protocol (E). In fact, with both PA and RAWMAC, packets travel upward and downward as quickly as with RW, but they wait for the next upward wave at the server node. For requests from the DAG root to target nodes, the use of RAWMAC and RW together (C) provides little improvement with respect to the use of RW alone (E). As expected, a shorter cycle time C_T decreases the overall delay both with RW (A with respect to E) and without RW (B with respect to F). More precisely, a shorter cycle time reduces the term $C_T/2$ in (12) (Section IV-B) and provides faster packet collision recovery. In Fig. 6, we also show, for comparison purposes, the theoretically produced delays as diamonds. The delays are generally consistent with the theoretical performance predicted by (3) (for experiments B and F), (12) (for experiments A, C, and E), and (19) (for experiment D) in Section IV. The theoretical delay expressions are corrected to take into account collisions, adding the supplementary delay in (7). The single collision probability p_c is estimated, based on our simulation results, as 0.027—this corresponds to the fraction of collided packet transmissions during the experiments. Nevertheless, theoretical predictions do not fit perfectly (although very accurate), as they correspond to an approximation, despite supplementary delays introduced in (7). Moreover, a few effects were not considered in the analysis, such as temporary network congestion and different collision probabilities in different regions of the network, depending on the local node spatial density.

Fig. 7 illustrates the average RR delays, computed separately for each group of nodes at the same depth in the RPL DODAG, and predicted by the theoretical equations recalled in the previous paragraph (i.e., (3) for experiments B and F; (12) for experiments A, C, and E; and (19) for experiment D) and corrected by the additive collision delay given by (7). In general, the delay is approximately a linearly increasing function of the node depth h. It is important to note that theoretical predictions have the same trends of the experimental results, thus approximating them with a relative error between 1% and 14%. In experiment D, as predicted, the delay increases slowly, mostly because of the higher collision probability, until depth 4, where the packet misses the next upward wave and the delay increases by C_T. At depth 3, the delay of experiment D is comparable with that in experiment C, where RW was also active. At that depth, the time to travel from the DAG root to that node and back (11) is almost a cycle time C_T. The response packet waits a very short time at the node, because an upward wave generated by RAWMAC occurs right after the request reception, and the on-demand wave by RW does not provide any additional improvement.

In order to evaluate the behavior of the proposed RW protocol under heavier traffic loads, experiments E and F were repeated considering shorter values of the request interval T_{RR}. As shown in Fig. 8, the RR delay is essentially unchanged for T_{RR} equal to 4 s and 2 s, as most responses can reach the root before the next request is sent. For lower T_{RR}, however, requests start interfering with each other and the average RR delay rises up to about twice as much for $T_{\mathrm{RR}} = 0.5$ s. Nonetheless, the proposed RW protocol (experiment E) is effective at reducing RR delay with respect to standard PA (experiment F) even in these conditions.

C. Delay and Power Consumption

We approximate the power consumption of a simulated node as proportional to the percentage of time which the radio is active for either listening, receiving or transmitting, as recorded by the Cooja PowerTracker. Indeed, in most real nodes, the radio transceiver is the most power-consuming component. A significant part of the radio activity in an experiment occurs during the formation of the RPL DODAG, when the nodes exchange a large amount of DIO and DAO messages. However, after the first 3 min, the DODAG is stable and the RPL trickle mechanism significantly reduces the number of DIO messages. To prevent bias, we computed the radio activity time separately for the first 3 min and for the remaining approximate 3.5 h of simulation.

TABLE III
RADIO ACTIVITY TIME PERCENTAGE

Experiment	First 3 min.	Remaining
A	1.48%	1.03%
B	1.46%	0.97%
C	2.00%	1.07%
D	1.94%	0.98%
E	1.89%	0.73%
F	1.88%	0.70%

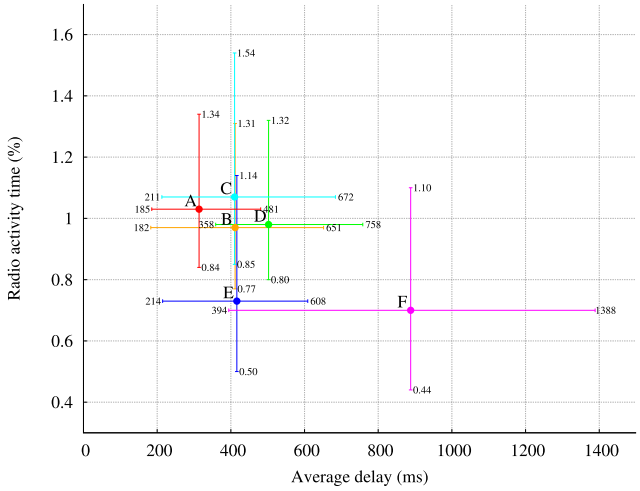

Fig. 9. Average RR delay for the network (x-axis) and radio activity time (y-axis). Horizontal error bars show minimum and maximum average delay (see Fig. 6). Vertical error bars show minimum and maximum radio activity percentage over the nodes in the network.

The observed results are shown in Table III. During the first 3 min, the radio activity is around 2% for experiments C, D, E, and F, while it is about 25% lower for experiments A and B. The results can be attributed to the large amount of broadcast DIO messages during network initialization, each of which is repeated for an entire cycle time. Cycle time is indeed shorter in experiments A and B than in C, D, E, and F. For the remaining part of the simulation, however, lower radio activity occurs for longer cycle time, i.e., in experiments E and F, because a smaller number of wake-ups occur. The same cycle time has been used in experiments C and D, but in these cases two wake-ups occur for each cycle time. Therefore, radio activity is comparable to that of a network with half cycle time, i.e., to configurations A and B. Finally, we observed a slight increase in radio activity due to the RR delay optimization RW, up to 9% in C compared to D.

The tradeoff between delay and power consumption is investigated in Fig. 9. Given a network with large cycle time (experiment F), the RR delay may be greatly reduced (experiment E) by using the proposed RW protocol, with a small relative increase in power consumption. A similar but slightly worse delay performance is instead obtained by integrating the RAWMAC protocol (experiment D). Integration with RAWMAC provides fast data harvesting from the nodes, as per RAWMAC properties. However, radio activity in experiment D increases by 40% and is similar to B, which has half the cycle time. A RR delay similar to E can be obtained by adding RW to *D*, resulting in a small increase in radio activity (experiment C).

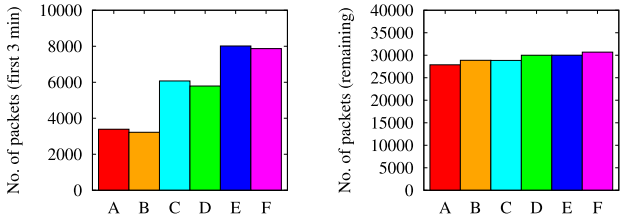

Fig. 10. Total number of transmitted packets at the physical layer, during the first 3 min (left) and the remaining part of the experiments (right).

In Fig. 10, we show the number of packets transmitted at the physical layer during the experiments. The small number of packets for experiments A and B during the first 3 min confirms that, if the cycle time is shorter, broadcast packets are repeated for a shorter time. Moreover, a smaller number of packets are sent in experiments C and D than in E and F during the first 3 min. As two wake-ups are performed for each cycle time in C and D, the ContikiMAC phase-lock mechanism can estimate the phase of the neighboring nodes with less probes. Even so, the numbers of transmitted packets beyond 3 min are comparable.

VI. CONCLUSION

In this paper, we have presented a cross-layer approach for the delay optimization of an RR protocol in RPL-based WSNs. The protocol generates upward and downward wake-up waves to minimize packet propagation delay. The performance of the proposed protocol has been evaluated using the Cooja simulator on the Contiki OS. The proposed approach has been compared with a PA approach and has been found to significantly reduce the RR delay, by about 53% with a 250-ms cycle time and by about 24% with a 125-ms cycle time. Moreover, it has been shown that the protocol can co-exist with the RAWMAC low-latency harvesting protocol. During the experiments, a tradeoff between network latency and power consumption has been highlighted. In particular, our results show that the proposed approach reduces the RR delay at the only cost of a slight increase in power consumption (about 9%). However, the use of RAWMAC alongside the proposed protocol almost doubles power consumption. As a future work, we plan to integrate the proposed approach with a multicast protocol, which allows to simultaneously query multiple nodes.

ACKNOWLEDGMENT

The work reflects only the authors' views; the European Commission is not liable for any use that may be made of the information contained herein.

REFERENCES

[1] T. Winter *et al.*, "RPL: IPv6 routing protocol for low-power and lossy networks," IETF, Fremont, CA, USA, RFC 6550, Mar. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6550

[2] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Tampa, FL, USA, Nov. 2006, pp. 641–648. doi: 10.1109/LCN.2006.322172.

[3] P. Gonizzi, P. Medagliani, G. Ferrari, and J. Leguay, "RAWMAC: A routing aware wave-based MAC protocol for WSNs," in *Proc. 10th IEEE Int. Conf. Wireless Mobile Comput. Netw. Commun.*, Larnaca, Cyprus, Oct. 2014, pp. 205–212. doi: 10.1109/WiMOB.2014.6962172.

[4] A. Dunkels, "The ContikiMAC radio duty cycling protocol," Swedish Inst. Comput. Sci., Rep. T2011:13, Dec. 2011. [Online]. Available: http://dunkels.com/adam/dunkels11contikimac.pdf

[5] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. ACM 1st Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2003, pp. 171–180. doi: 10.1145/958491.958512.

[6] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. ACM 2nd Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2004, pp. 95–107. doi: 10.1145/1031495.1031508.

[7] M. L. Wymore and D. Qiao, "BladeMAC: Radio duty-cycling in a dynamic, cyclical channel," in *Proc. IEEE Int. Conf. Commun.*, Paris, France, May 2017, pp. 1–7. doi: 10.1109/ICC.2017.7996981.

[8] A. El-Hoiydi and J. Decotignie, "WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks," in *Proc. 9th Int. Symp. Comput. Commun.*, vol. 1. Alexandria, Egypt, Jul. 2004, pp. 244–251. doi: 10.1109/ISCC.2004.1358412.

[9] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. ACM 4th Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, Boulder, CO, USA, 2006, pp. 307–320. doi: 10.1145/1182807.1182838.

[10] M.-P. Uwase, M. Bezunartea, J. Tiberghien, J.-M. Dricot, and K. Steenhaut, "Experimental comparison of radio duty cycling protocols for wireless sensor networks," *IEEE Sensors J.*, vol. 17, no. 19, pp. 6474–6482, Oct. 2017. doi: 10.1109/JSEN.2017.2738700.

[11] C. G. Peces, J. Eriksson, and N. Tsiftes, "Sleepy devices vs. Radio duty cycling: The case of lightweight M2M," *IEEE Internet Things J.*, to be published. doi: 10.1109/JIOT.2018.2871721.

[12] Q. Chen, H. Gao, Z. Cai, L. Cheng, and J. Li, "Distributed low-latency data aggregation for duty-cycle wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2347–2360, Oct. 2018. doi: 10.1109/TNET.2018.2868943.

[13] S. Bhandari, S. K. Sharma, and X. Wang, "Latency minimization in wireless IoT using prioritized channel access and data aggregation," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6. doi: 10.1109/GLOCOM.2017.8255038.

[14] A. Castagnetti, A. Pegatoquet, T. N. Le, and M. Auguin, "A joint duty-cycle and transmission power management for energy harvesting WSN," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 928–936, May 2014. doi: 10.1109/TII.2014.2306327.

[15] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *Proc. ACM/IEEE 11th Int. Conf. Inf. Process. Sensor Netw.*, Beijing, China, Apr. 2012, pp. 185–196. doi: 10.1109/IPSN.2012.6920956.

[16] P. Gonizzi, R. Monica, and G. Ferrari, "Design and evaluation of a delay-efficient RPL routing metric," in *Proc. 9th Int. Wireless Commun. Mobile Comput. Conf.*, Sardinia, Italy, Jul. 2013, pp. 1573–1577. doi: 10.1109/IWCMC.2013.6583790.

[17] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing metrics used for path calculation in low-power and lossy networks," IETF, Fremont, CA, USA, RFC 6551, Mar. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6551

[18] P. Thubert, "Objective function zero for the routing protocol for low-power and lossy networks (RPL)," IETF, Fremont, CA, USA, RFC 6552, Mar. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6552

[19] O. Gnawali and P. Levis, "The minimum rank with hysteresis objective function," IETF, Fremont, CA, USA, RFC 6719, Sep. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6719

[20] J. Song *et al.*, "WirelessHART: Applying wireless technology in real-time industrial process control," in *Proc. IEEE Real Time Embedded Technol. Appl. Symp.*, St. Louis, MO, USA, Apr. 2008, pp. 377–386. doi: 10.1109/RTAS.2008.15.

[21] T. Watteyne, M. Palattella, and L. Grieco, "Using IEEE 802.15.4e time-slotted channel hopping (TSCH) in the Internet of Things (IoT): Problem statement," IETF, Fremont, CA, USA, RFC 7554, May 2015. [Online]. Available: https://tools.ietf.org/html/rfc7554

[22] M. Sha, G. Hackmann, and C. Lu, "Energy-efficient low power listening for wireless sensor networks in noisy environments," in *Proc. ACM 12th Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, 2013, pp. 277–288. [Online]. Available: http://doi.acm.org/10.1145/2461381.2461415

[23] J. Park and J. Ko, "Dynamic low-power listening with data-rate proportional wakeup period management," in *Proc. IEEE 22nd Int. Conf. Embedded Real Time Comput. Syst. Appl.*, Daegu, South Korea, Aug. 2016, p. 97. doi: 10.1109/RTCSA.2016.35.

[24] S. Zhuo and Y.-Q. Song, "GoMacH: A traffic adaptive multi-channel MAC protocol for IoT," in *Proc. IEEE 42nd Conf. Local Comput. Netw.*, Singapore, Oct. 2017, pp. 489–497. doi: 10.1109/LCN.2017.72.

[25] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-MAC: A hybrid MAC for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, Jun. 2008. doi: 10.1109/TNET.2007.900704.

[26] S. Zhuo, Z. Wang, Y.-Q. Song, Z. Wang, and L. Almeida, "iQueue-MAC: A traffic adaptive duty-cycled MAC protocol with dynamic slot allocation," in *Proc. IEEE Int. Conf. Sens. Commun. Netw.*, New Orleans, LA, USA, Jun. 2013, pp. 95–103. doi: 10.1109/SAHCN.2013.6644967.

[27] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Netw.*, vol. 11, no. 4, pp. 419–434, Jul. 2005. doi: 10.1007/s11276-005-1766-z.

[28] M. Michel and B. Quoitin, "ContikiMAC vs X-MAC performance analysis," *CoRR*, vol. abs/1404.3589, pp. 1–29, Apr. 2014. [Online]. Available: http://arxiv.org/abs/1404.3589

[29] K. Nichols, S. Blake, F. Baker, and D. L. Black, "Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers," IETF, Fremont, CA, USA, RFC 2474, Dec. 1998. [Online]. Available: https://tools.ietf.org/html/rfc2474

[30] *IEEE Standard for Low-Rate Wireless Networks*, IEEE Standard 802.15.4-2015, pp. 1–709, Apr. 2016. doi: 10.1109/IEEESTD.2016.7460875.

[31] *RDC Phase Optimization*. Accessed: Mar. 29, 2019. [Online]. Available: https://github.com/contiki-os/contiki/wiki/RDC-Phase-optimization

Riccardo Monica (S'15–GS'15–M'17) received the master's degree in computer engineering and the Ph.D. degree in information technologies from the University of Parma, Parma, Italy, in 2014 and 2018, respectively.

He is currently a Post-Doctoral Researcher with the Robotics and Intelligent Machines Laboratory, Department of Engineering and Architecture, University of Parma. His current research interests include robot perception and 3-D reconstruction.

Luca Davoli (GS'15–M'17) received the master's degree in computer engineering and the Ph.D. degree in information technologies from the Department of Information Engineering, University of Parma, Parma, Italy, in 2013 and 2017, respectively.

He is currently a Post-Doctoral Researcher with the Internet of Things (IoT) Laboratory, Department of Engineering and Architecture, University of Parma. His current research interests include IoT, pervasive computing, big stream, mobile computing, and software-defined networking.

Gianluigi Ferrari (S'96–M'98–SM'12) received the Laurea (*summa cum laude*) and Ph.D. degrees in electrical engineering from the University of Parma, Parma, Italy, in 1998 and 2002, respectively.

Since 2002, he has been with the University of Parma, where he is currently an Associate Professor of telecommunications (with National Scientific Qualification for Full Professorship since 2013) and also the Coordinator of the Internet of Things (IoT) Laboratory, Department of Engineering and Architecture. He is the co-founder and the President of things2i s.r.l., a spin-off company of the University of Parma dedicated to IoT and smart systems. He has authored or coauthored extensively. His current research interests include signal processing, advanced communication and networking, and IoT and smart systems.