RAWMAC: A Routing Aware Wave-based MAC Protocol for WSNs

Pietro Gonizzi*, Paolo Medagliani†, Gianluigi Ferrari*, Jérémie Leguay†
*Department of Information Engineering, University of Parma,
Parco Area delle Scienze 181/a, 43124 Parma, Italy
Email: pietro.gonizzi@studenti.unipr.it, gianluigi.ferrari@unipr.it
†Thales Communications & Security, 4 avenue des Louvresses, 92622 Gennevilliers, France
Email: paolo.medagliani@thalesgroup.com, jeremie.leguay@thalesgroup.com

***Abstract*—In Wireless Sensor Networks (WSNs) for monitoring applications, energy saving and fast data collection are two challenging tasks. Asynchronous radio duty cycling protocols can achieve very low energy consumption in low traffic conditions and they are fault-tolerant to clock drifts. However, they may exhibit a delay degradation due to the decoupled wake-up periods of the nodes. In this paper, we present RAWMAC, a cross-layer approach where RPL, a tree-based routing protocol, orchestrates the asynchronous duty-cycled ContikiMAC MAC layer. The wake-up instants of the nodes are dynamically aligned, with respect to the RPL topology, to minimize the delay for data collection. We implement RAWMAC for the Contiki operating system and we analyze the impact of several key system parameters. Results show that RAWMAC outperforms ContikiMAC in terms of delay for data collection, while keeping the same performance in terms of throughput and energy consumption.**

I. Introduction

The *Internet of Things* (IoT) allows to exploit low power and battery-operated wireless devices for long lasting monitoring systems. These devices can self-organize and form a Wireless Sensor Network (WSN) to monitor a large area of interest. They are typically composed of an embedded microcontroller with some memory, a radio transceiver, physical transducers that sense the environment, and a battery.

The need for low power but interoperable WSN systems has already been addressed in the literature. Radio duty cycling techniques at the MAC layer have been introduced to reduce energy consumption by turning on and off periodically the radio interfaces [1]. Similarly, IPv6 extensions and RPL, the IPv6 Routing Protocol for Low power and Lossy Networks [2], have been proposed to provide IP connectivity on low power radios such as IEEE 802.15.4. This paper focuses on a practical and standard compliant coordination between RPL and the asynchronous duty-cycled ContikiMAC MAC layer [3] to minimize energy consumption while meeting tight delay objectives.

MAC protocols can be schematically divided into synchronous and asynchronous. Synchronous protocols align all nodes on one common schedule. While these protocols are suitable for delay-sensitive applications, they require an underlying synchronization mechanism, which consumes more energy, computation resources, and bandwidth [4]. On the other hand, asynchronous protocols do not require any control overhead for the synchronization; therefore, these protocols improve energy efficiency and are more robust to clock drifts, but they may exhibit a delay degradation due to the decoupled wake-up periods of the nodes.

In this paper, we propose RAWMAC, an adaptation layer which exploits the routing layer, where the RPL protocol is used, into a management layer for asynchronous duty-cycled MAC protocols (such as ContikiMAC). The idea is to exploit the Directed Acyclic Graph (DAG) built by RPL to align each node wake-up phase with that of its preferred parent, creating a data propagation "wave" from the leaves of the DAG to the root. This allows to significantly reduce the latency, as it depends only on small propagation delays and on the internal processing at each device. By properly configuring the phase lock mechanism of ContikiMAC, the transmitting node wakes up only when the receiving node is ready to receive the packet, so that the energy consumption is kept as low as possible. A similar approach to align duty cycles using routing information has been proposed in DMAC [5]. However, DMAC considers a synchronous MAC layer with static routing, while we aggregate an asynchronous duty cycling mechanism with a dynamic routing plane, such as RPL.

The main contributions of our work are: (i) it leverages on existing standards for unslotted constrained WSNs, such as RPL; (ii) it configures the asynchronous MAC layer wake-ups using information from a dynamic IoT-oriented routing plane; and (iii) it provides an implementation for a real deployment, based on Contiki OS.

The obtained results show that RAWMAC outperforms ContikiMAC in terms of delay for data collection, while keeping the same performance in terms of throughput and energy consumption. In addition, the creation of propagation "waves" does not impact the packet delivery ratio of the WSN, since no overhead is introduced by RAWMAC. Conversely, if we fix a delay threshold for data collection, RAWMAC allows to satisfy the requirement with a larger duty cycle, resulting in a consistent energy saving.

The structure of this paper is the following. Section II presents the related work. Section III presents an overview of ContikiMAC and RPL, which are the protocols used to implement our mechanism. Section IV describes our proposed solution in detail; analytical bounds for the delay are also derived. Section V presents the performance evaluation of RAWMAC. Finally, Section VI concludes the paper.

II. Related Work

In order to overcome intrinsic limitations of traditional single MAC layer protocols, a number of cross-layer approaches have been proposed. The cross-layer optimization involving physical, MAC and routing layers has been considered in [6]: by jointly optimizing MAC and routing layers, this work adapts the wake up phases in order to minimize the delays. This approach echoes the one proposed in [7], where a "wave" of propagation is created to collect alarms generated by the nodes deployed in a given monitored area. However, this approach requires an external configuration phase, since the collection path must be a priori determined. In [8], authors present an interesting solution, based on IEEE 802.15.4 and RPL, to reduce the delay for data collection. However, the proposed approach is for slotted cluster-tree networks with very low duty-cycles (several minutes), which does not fit the requirements of a surveillance system, where required delays are in the order of milliseconds. The idea of considering propagation waves has also been presented in [9], where authors focus on alarms collection and redistribution via an energy efficient broadcast. However, network topology is statically defined at network startup and never updated during time. Finally, a similar mechanism is proposed in DMAC [5]. DMAC is based on the staggering of duty cycles using information coming from the routing plane. However, this solution does not use a dynamic routing protocol, such as RPL, and it relies on a synchronous approach rather than ContikiMAC. In addition, to the best of our knowledge, no implementation of DMAC is available in Contiki, therefore no comparison tests are possible.

Current work on 6TiSCH [10] at IETF aims at providing a management layer for the synchronous and multichannel 802.15.e MAC layer. The idea is that an external Path Computation Element (PCE) solves the joint MAC and routing optimization to minimize the collection times while minimizing the global energy consumption. However, in order to provide fault-tolerance to clock drifts, nodes failures and propagation issues, we believe that managed asynchronous MAC protocols are very powerful as they can still operate in asynchronous mode in the case of failure.

III. ContikiMAC and RPL

With respect to related works outlined in Section II, our solution is based on ContikiMAC and RPL, the "de facto" emerging protocols for IoT. Therefore, in the following we present a quick overview of the two protocols.

A. ContikiMAC

ContikiMAC is an asynchronous, sender-initiated radio duty cycling protocol [3]. In order to transmit a packet, a sender repeatedly transmits its packet until it receives a link layer ACKnowledgment (ACK) from a receiver. The packet is repeated, in the worst case, for an entire sleeping interval, to ensure that the receiver awakes at least once during this period. The sleeping interval is denoted as *cycle time* (C_{T}, dimension: [s]). On the other side, the receiver periodically wakes up, with period C_{T}, to check for possible incoming packet transmissions. If a packet transmission is detected during a wake-up, the receiver turns its radio transceiver on to be able to receive the entire packet. After complete reception, the receiver sends the ACK to the sender. Before transmitting, the node must sense if the channel is available. In order to reduce energy consumption, ContikiMAC introduces a phase lock mechanism to learn the wake-up phase of the receiver. By recording the time of the reception of the ACK, and assuming that the receiver will wake-up at the constant interval C_{T}, the phase lock mechanism is able to estimate the wake-up time of the receiver. This allows to decrease the number of probes needed in the following transmissions, thus reducing the energy consumption. A graphical representation of the phase lock mechanism is shown in Fig. 1.

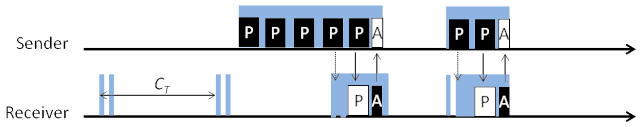


Fig. 1. Phase lock mechanism in ContikiMAC: the sender learns the wake-up phase of the receiver after reception of an ACK. At the second transmission, the sender can decrease the number of probes. The cycle time is C_{T}.

B. RPL

RPL is a distance-vector routing protocol based on the creation of a tree-like structure, referred to as *Destination Oriented Acyclic Directed Graph* (DODAG) [2]. The tree is anchored at one or more nodes, denoted as DAG root(s). The cost of each path in the tree is evaluated according to metrics defined in an objective function. The current RPL implementation for the Contiki operating system adopts, as default, the Expected Transmission Count (ETX) metric, which tries to minimize the total number of packet transmissions required to successfully deliver a packet to the ultimate destination.

In order to build and maintain the topology, RPL uses two types of control messages: *DODAG Information Objects* (DIOs) are broadcast messages used in the down direction to help nodes to join the topology. *Destination Advertisement Objects* (DAOs) are unicast messages used to populate the routing tables of ancestor nodes in the DAG and to create downstream routes. RPL uses a *trickle* mechanism to reduce the transmission of redundant DIO messages when the network is stable. RPL has two modes of operation, namely, *storing* and *non-storing*: in storing mode, downward routes are stored by each node in the tree; in non-storing mode, routes are stored only by the root.

IV. Design Principles of RAWMAC

We assume that a WSN is deployed in a surveillance system with bounded delay requirements. In such a scenario, the traffic transmitted upward, which consists of alerts, is far more critical than the one directed downward, which may include configuration requests or software updates. Nodes are organized in a tree-like structure, e.g., a RPL DAG, rooted at a sink node. In order to save energy, nodes periodically switch their radios on and off.

The goal of RAWMAC is to minimize the data collection process (i.e., the delay of upward traffic), while keeping radio duty-cycling for energy saving purposes. We assume that the

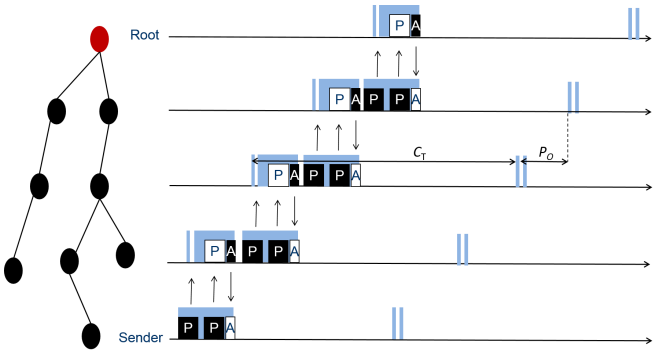


Fig. 2. Design principle of RAWMAC: wake-up phase alignment. According to the network topology built by RPL, each node aligns its wake-up phase to that of its preferred parent in the DAG. The top node in red is the root of the DAG.

sleeping period is the same for all nodes and has been set to meet the requirements of external applications. To achieve the described goal, nodes progressively align their wake-up phases to those of their preferred parents in the RPL DAG, configuring the asynchronous phase lock mechanism of ContikiMAC. This phase shifting mechanism reduces the delay for alert collection, without requiring any overhead for the alignment of the phases.

A. Wake-up Phase Alignment

An illustrative representation of the wake-up phase alignment in RAWMAC is shown in Fig. 2. As long as the routing structure is established, a node shifts its wake-up phase in order to be aligned with that of its parent. More precisely, it sets the wake-up phase to the time at which it saw the last link layer ACK from its RPL preferred parent. Since the preferred parent must have been awake to receive the packet, the reception of the ACK means that it has successfully transmitted a packet within the preferred parent's wake-up window and, thus, that it has found the preferred parent's wake-up phase. We define the phase offset P_{o} (dimension: [s]) as the offset between the node's wake-up phase and the wake-up phase of its parent. Given the node's sleeping interval C_{T}, it holds that $0 \leq P_{\mathrm{o}} \leq C_{\mathrm{T}}$. The parameter P_{o} has indeed to be chosen carefully, since it has an impact on the system delay performance. If P_{o} is too short, a node relaying a packet may not be able to catch its parent's wake-up because the reception of the same packet from its child has not completed yet. If this is the case, then the child should wait the next cycle time C_{T} to be able to forward the packet. If P_{o} is too long, instead, the delay increases significantly, as the sender has to wait for the receiver to wake up to be able to transmit the packet.

RAWMAC adjusts the phase alignment every time the node receives an ACK from its preferred parent: this happens for the transmission of application data, and when RPL DAO messages are sent. In addition, RAWMAC leverages the phase lock mechanism of ContikiMAC which allows the transmitting node to send the packet only when the destination node is ready to receive it, allowing a considerable energy saving since useless packet strobing is suppressed.

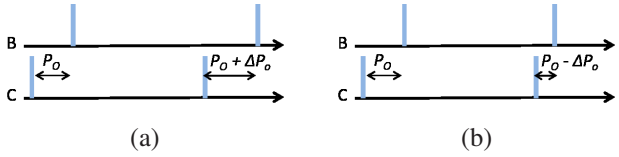


Fig. 3. Examples of clock drift which may occur after the wake-up phase alignment. In (a), the wake-up phases diverge. In (b), the wake-up phases get very close with each other. ΔP_{o} is the clock drift.

With RAWMAC, nodes which are located at the same hop distance from the sink in the routing structure calibrate the same wake-up phase. In case a parent has several children, packet collisions may arise. RAWMAC handles this problem exactly as ContikiMAC, relying on a CSMA strategy. In Section V, we evaluate the impact of traffic load on the delay and energy consumption performance. In particular, we highlight how the collisions affect those two performance indicators, trying to identify the maximum offered traffic that can be supported by the network with negligible performance degradation.

B. Clock Drift

The phase lock mechanism estimates the wake-up phase of the receiver with some error. The reception of the ACK does not correspond to the time at which the receiver has switched the radio on, but it comes from several contributions, such as: (i) the time spent by the receiver for a partial reception of the packet; (ii) the time for a complete reception of a packet; (iii) the time to process the packet and to generate the ACK; and (iv) the time to deliver the ACK back to the sender. In short, the reception of the ACK depends on the length of the transmitted packet. In addition, the internal clocks of the nodes are not synchronized with each other, and different oscillators may cause small clock drifts, which would affect the phase synchronization as well.

RAWMAC introduces the following mechanism to prevent a node from shifting its wake-up phase if the offset with its preferred parent has not changed significantly. We define a phase offset threshold ΔP_{o} (dimension: [s]), with $\Delta P_{\mathrm{o}} \leq P_{\mathrm{o}}$, to check if the phase should be aligned or not. In particular, denoting with ϕ_{C} and $\phi_{\mathrm{B,C}}$ (dimension: [s]) the wake-up phases of node C and of node C's preferred parent B (estimated by C), with $\phi_{\mathrm{C}} \leq \phi_{\mathrm{B,C}}$, respectively, node C aligns its wake-up phase to B's wake-up phase if

$$\phi_{\mathrm{B,C}} - \phi_{\mathrm{C}} \geq P_{\mathrm{o}} + \Delta P_{\mathrm{o}} \vee \phi_{\mathrm{B,C}} - \phi_{\mathrm{C}} \leq P_{\mathrm{o}} - \Delta P_{\mathrm{o}}. \quad (1)$$

Note that Eq. 1 underlies the assumption that the drift may cause a convergence as well as a divergence of the phases with equal probability, as shown in Fig. 3.

Similarly to P_{o}, the parameter ΔP_{o} has to be chosen carefully, since it has an impact on the system delay performance. If ΔP_{o} is too short, a continuous reshift of the phase would occur at any new received ACK, which may cause instabilities downward and may make children lose the synchronization with their preferred parents. If ΔP_{o} is too long, reshift would occur more occasionally, which may cause a node to lose the synchronization with its preferred parent.

C. Topology Reconfiguration

Changes in the network topology may occur. In particular, a node may change preferred parent whenever a neighbor has a lower path cost to the DAG root. If a node changes its preferred parent in the DAG, RAWMAC aligns its wake-up phase to that of the new preferred parent. This may cause children to lose the synchronization, and the effect would propagate to children nodes, since each child should reconfigure its wake-up phase as well.

In order to be more reactive against changes of the routing structure, we modify the phase lock mechanism of Contiki-MAC. If a node does not receive an ACK after 4 consecutive transmissions—16 in ContikiMAC— it starts a new phase discovery process.

D. Delay Evaluation

Under the assumption of low generated traffic and equal duty cycles at the nodes, the transmission delay can be analytically evaluated as follows.

The phase offset used by RAWMAC to align the wake-up phase of a node with that of its parent node is denoted as P_{o}. As demonstrated in [11], the forwarding delay on a single hop d_{sh}, with uncorrelated phases, can be expressed as

$$d_{\mathrm{sh}} = C_{\mathrm{T}}/2 + P_{\mathrm{min}} \tag{2}$$

where P_{min} (dimension: [s]) is the minimum forwarding time needed to forward a packet to a neighbor. In Section V we will derive this parameter by simulations. In the case of RAWMAC, each node shifts the wake-up phase to match that of its preferred parent (which is, indeed, a neighbor). The shift is dictated by the phase-offset P_{o}. In this case, the upward delay $D_{\mathrm{up,R}}$ for a node which is h hops away from the sink can be expressed in two ways, depending on the value of P_{o}:

$$D_{\mathrm{up,R}} = \begin{cases} (h-1)\cdot P_{\mathrm{o}} + d_{\mathrm{sh}} & P_{\mathrm{o}} > P_{\mathrm{min}} \\ (h-1)\cdot (P_{\mathrm{o}} + C_{\mathrm{T}}) + d_{\mathrm{sh}} & 0 \le P_{\mathrm{o}} \le P_{\mathrm{min}} \end{cases} \tag{3}$$

In the second expression in Eq. 3, P_{o} is shorter than the minimum forwarding time P_{min}: therefore, each intermediate node has to wait an additional cycle time C_{T} to forward the packet. For the first hop transmission, the delay is given by d_{sh}, since there is no correlation between the packet generation instant and the status of the duty-cycle of the node. For the remaining $h-1$ hops, the delay is given only by the P_o, since the propagation wave has been created.

Similar considerations hold for the delay in the downward direction, denoted as $D_{\mathrm{down,R}}$. In particular, two cases can be distinguished in this case as well:

$$D_{\mathrm{down,R}} = \begin{cases} (h-1) * (C_{\mathrm{T}} - P_{\mathrm{o}}) + d_{\mathrm{sh}} & C_{\mathrm{T}} - P_{\mathrm{o}} > P_{\mathrm{min}} \\ (h-1) * (C_{\mathrm{T}} - P_{\mathrm{o}} + C_{\mathrm{T}}) + d_{\mathrm{sh}} & 0 \le C_{\mathrm{T}} - P_{\mathrm{o}} \le P_{\mathrm{min}} \end{cases} \tag{4}$$

As before, the delay for the first hop transmission is given by d_{sh}.

In the case of ContikiMAC, the average downward delay $D_{\mathrm{down,C}}$, from the sink to a node placed at h hops, is equivalent to the delay in the up direction $D_{\mathrm{up,C}}$, since the wake-up phases are uncorrelated. Thus, one obtains

$$D_{\mathrm{down,C}} = D_{\mathrm{up,C}} = h \cdot (C_{\mathrm{T}}/2 + P_{\mathrm{min}})\,. \tag{5}$$

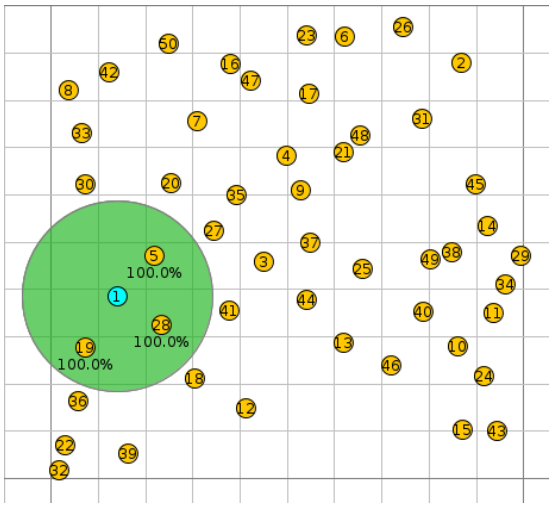


Fig. 4. Evaluation scenario with $N = 50$ nodes. We use the unit disk radio model with distance loss. The transmission range is set to 20 m.

V. Performance Evaluation

We have implemented RAWMAC in Contiki 3.x and we have tested it via Cooja, a Java-based simulator for Contiki-based WSNs [12]. The simulated scenario, shown in Fig. 4, is composed of $N = 50$ randomly-deployed nodes. Two types of node are deployed: (*i*) a sink node, responsible for configuring the RPL DAG and collecting the data, and (*ii*) sending nodes, in charge of transmitting packets periodically to the sink and, eventually, relaying incoming packets. As soon as the network is started, the sink creates an RPL DAG which can be joined by the other nodes. In Fig. 4, the sink node is node 1. Each sending node generates an 8-byte length alert packet at a random instant within consecutive time slots of duration T (dimension: [s]). For instance: one packet at a random instant in $[0, T]$; one packet at a random instant in $[T, 2T]$; etc. Therefore, the average packet generation rate r per node (dimension: [pck/s]) is equal to $1/T$. Packets are sent to the sink using the UDP transport protocol. The duration of the simulations T_{sim} (dimension: [s]) is 5 hours. In order to bound statistical fluctuations, each simulation result is obtained by averaging over three runs (with different random seeds).

Table I summarizes the main parameters of RAWMAC, with the associated values. In the remainder of this section, when not explicitly stated, we will consider $T = 120$ s.

In order to estimate energy consumption, we rely on Powertrace, a Cooja plugin that measures and logs energy consumption for each node in the network. Our simulations have been carried out with Tmote Sky nodes, whose datasheet current consumptions for transmission and reception phases, namely, I_{Tx} and I_{Rx}, respectively, are indicated in Table I. We study how the system performance is affected by (*i*) the phase offset P_{o}, (*ii*) the phase offset threshold ΔP_{o}, and (*iii*) the average packet generation period T. The performance of

Fixed parameters		
Duration of the simulation	T_{sim}	5 hours
Number of nodes	N	50
Radio duty cycle	C_{T}	250 ms
Transmission current consumption	I_{Tx}	20 mA
Reception current consumption	I_{Rx}	20 mA
Voltage tension	V_{DD}	3 V
Packet length	L	8 bit
Variable parameters		
Phase offset	P_{o}	$25 \div 55$ ms
Phase offset threshold	ΔP_{o}	$1 \div 9$ ms
Packet generation period	T	$25 \div 200$ s

TABLE I. FIXED AND VARIABLE PARAMETERS CONSIDERED IN SIMULATIONS.

RAWMAC is directly compared with that of ContikiMAC in terms of delay to and from the sink, energy consumption, packet loss rate, number of parent changes, and number of wake-up phase realignments.

A. RPL Topology

In this subsection, we first evaluate the depth of the RPL DAG created according to the considered topology. Routing information is collected from the nodes periodically, during the simulations, in order to track, over time, changes in the routing structure. This analysis is expedient to understand, in the following subsections, the impact of traffic load and of the DAG depth on RAWMAC. In Fig. 5 (a), the DAG depth is shown for each node involved in the network. Since topology variations are present, we also show the standard deviation associated to the average number of hops at each node. Node 1 is not shown since it is the DAG root. It can be observed that the resulting DAG is quite stable and nodes rarely change their positions. However, nodes at a higher depth in the DAG experience a higher variability than those closer to the root. This result is confirmed in Fig. 5 (b), where the Probability Mass Function (PMF) of the nodes distribution, as a function of the number of hops to the DAG root, is shown. Each mass is associated with the corresponding standard deviation. There are only a few nodes which are 8 or 9-hops away from the sink. It can also be observed that deep nodes (at least 6 hops away) are not very stable. This result is reasonable since the computation of the ETX objective function is more reliable at shorter depths. If there is an error in the measurements of the metric, this error propagates hop-by-hop, causing instability in the DAG at higher depths. For this reason, in the following we consider only a maximum DAG depth of 7 hops.

B. Impact of the Phase Offset P_{o}

As stated in Section IV-A, the phase shift impacts the delay and energy consumption performance. In Fig. 6 (a), we show the impact of the phase offset on the upward delay performance. In order to validate our adaptation layer, we compare RAWMAC with ContikiMAC using the phase lock mechanism. The performance results with RAWMAC have been generated considering $\Delta P_{\text{o}} = 6$ ms. According to Eq. (2), the delay at the first hop (i.e., at depth 1 in the DAG) is the same for RAWMAC and ContikiMAC, since it depends only on the packet generation instant. At higher depths, two different behaviors can be observed. First, as

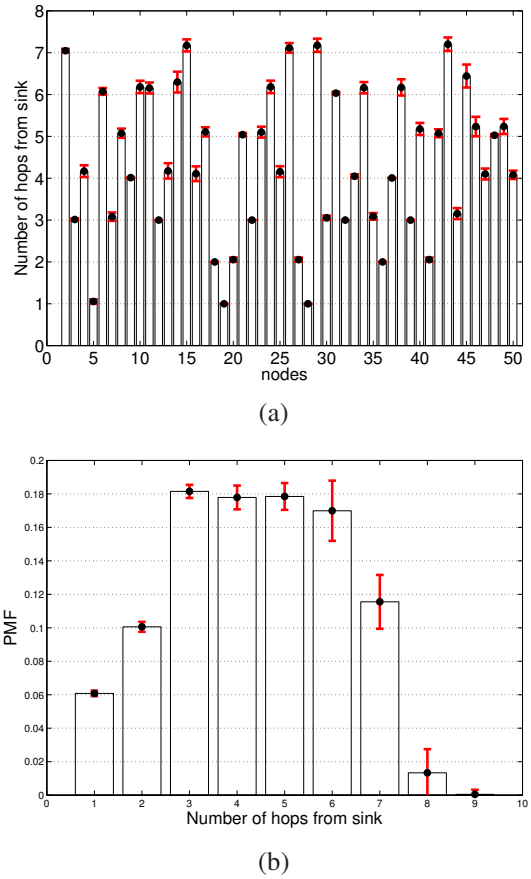


Fig. 5. Configuration of the RPL routing topology. In (a), the average hop distance and standard deviation are shown for each node. In (b), the PMF of the number of hops (to the DAG root) is shown.

expected, for proper values of P_{o} RAWMAC outperforms ContikiMAC, especially for increasing number of hops to the sink. In particular, for packets generated 6- or 7-hops away from the sink, the gain of RAWMAC, with respect to ContikiMAC, is over 30%. Second, we can remark that when P_{o} is too small (i.e., $P_{\text{o}} = 25$ ms), a packet cannot "ride" the same wave till the sink: at some depth, it has to wait for the subsequent wave to reach the next parent. This means that, with reference to Eq. (2), we can approximate $P_{\min} \approx 35$ ms.

RAWMAC has been conceived for fast and delay-efficient data collection. The cost of this adaptation mechanism is a larger delay for downward traffic. In Fig. 6 (b), the downward delay is shown as a function of the DAG depth of the destination node, considering various values of P_{o}. More precisely, the sink generates a packet, on average, every $T = 120$ s and this packet is directed to one randomly-chosen node in the WSN. Besides statistical fluctuations, ContikiMAC performs like RAWMAC for the very first hops, while it outperforms RAWMAC when the destination node is deeper in the DAG, i.e., farther from the sink.

In Fig. 6 (c), the energy consumption of the radio interface in the active mode, denoted with E_{act} (dimension: [mJ]), is shown as a function of the number of hops to the sink, for various values of P_{o}. As expected, there is roughly no difference between RAWMAC and ContikiMAC, since they

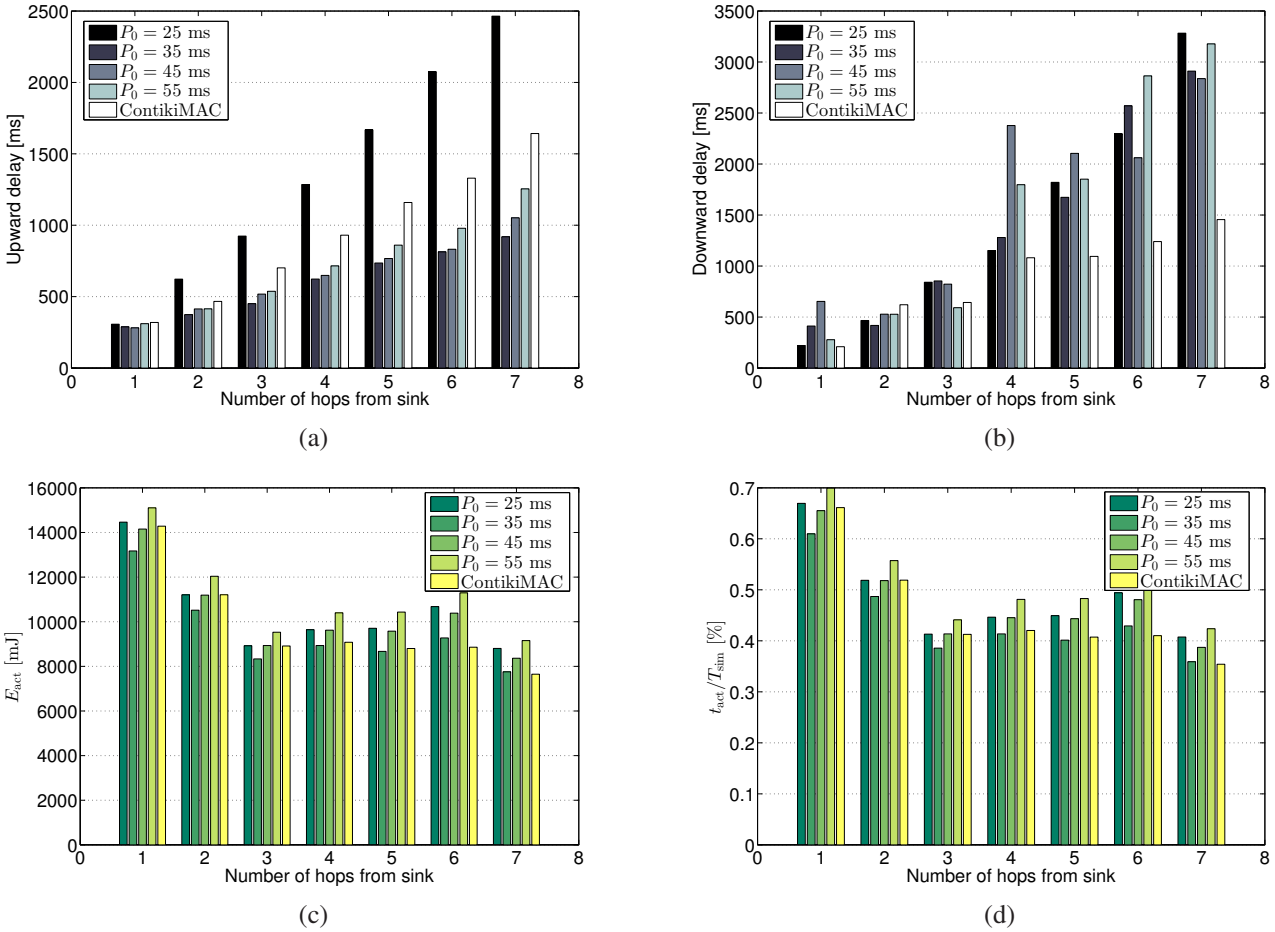


Fig. 6. (a) Average delay upward, (b) average delay downward, (c) energy consumption, and (d) percentage of activity of radio interface as functions of the number of hops to the DAG root, for different values of the phase offset P_{o}. Performance is compared with ContikiMAC.

are both based on the phase lock mechanism that allows to turn radio interfaces on only when the receiver is expected to be available. Even for $P_{\mathrm{o}} = 25$ ms, when RAWMAC is outperformed by ContikiMAC in terms of upward delay (see Fig. 6 (a)), the energy performance is still comparable to that of ContikiMAC, since relaying nodes postpone their transmissions without staying awake till the intended receivers wake up.

From the energy consumption curves it is possible to derive the time each node has spent in transmission and reception during the simulation. In particular, the power consumption of the radio in the active mode P_{act}, that is when the node is either transmitting or receiving, can be expressed as

$$P_{\mathrm{act}} = V_{\mathrm{DD}} I_{\mathrm{act}} = V_{\mathrm{DD}}(I_{\mathrm{Tx}} + I_{\mathrm{Rx}}) = \frac{E_{\mathrm{act}}}{t_{\mathrm{act}}} \qquad (6)$$

where I_{act} is the current consumption due to the active phase, that is for transmitting (I_{Tx}) and receiving (I_{Rx}) a packet, E_{act} is the energy consumption due to the active phase, and t_{act} is the time spent in the active phase, over the duration of the simulation T_{sim}. The values for V_{DD}, I_{Tx}, I_{Rx}, and T_{sim} are shown in Table I. From (6), one can compute t_{act} and, consequently, the percentage of time spent for radio activity by a node over T_{sim}, which is investigated in Fig. 6 (d), for various values of P_{o}. For nodes closer to the sink, the amount of time spent with the radio interface on is approximately 1.4% with both RAWMAC and ContikiMAC. Intuitively, the deeper the position of a node in the DAG, the higher should t_{act} be. This trend is confirmed up to the third hop. However, since the network is rather dense, at 4-hops from the sink there are more contentions to transmit to the parents, resulting in a much higher radio activity. The same trend can be highlighted for ContikiMAC, confirming that the higher radio activity at 4 hops is due to topology configuration.

C. Impact of the Phase Offset Threshold ΔP_{o}

Another parameter that has a relevant impact on RAWMAC is the tolerance to the phase shift offset. Internal clocks imperfections and internal processing delays may make a node unprepared to receive a packet at a scheduled instant. If no tolerance is introduced, the sender would always need to track the changes of its parent's wake up scheduling. Clearly, this would impact on the node's children activity, since they should track as well the changed wake-up instant. As the misalignment between nodes may cause retransmissions, the phase offset threshold ΔP_{o} has an impact on the ETX metric used by RPL and may lead to network instability.

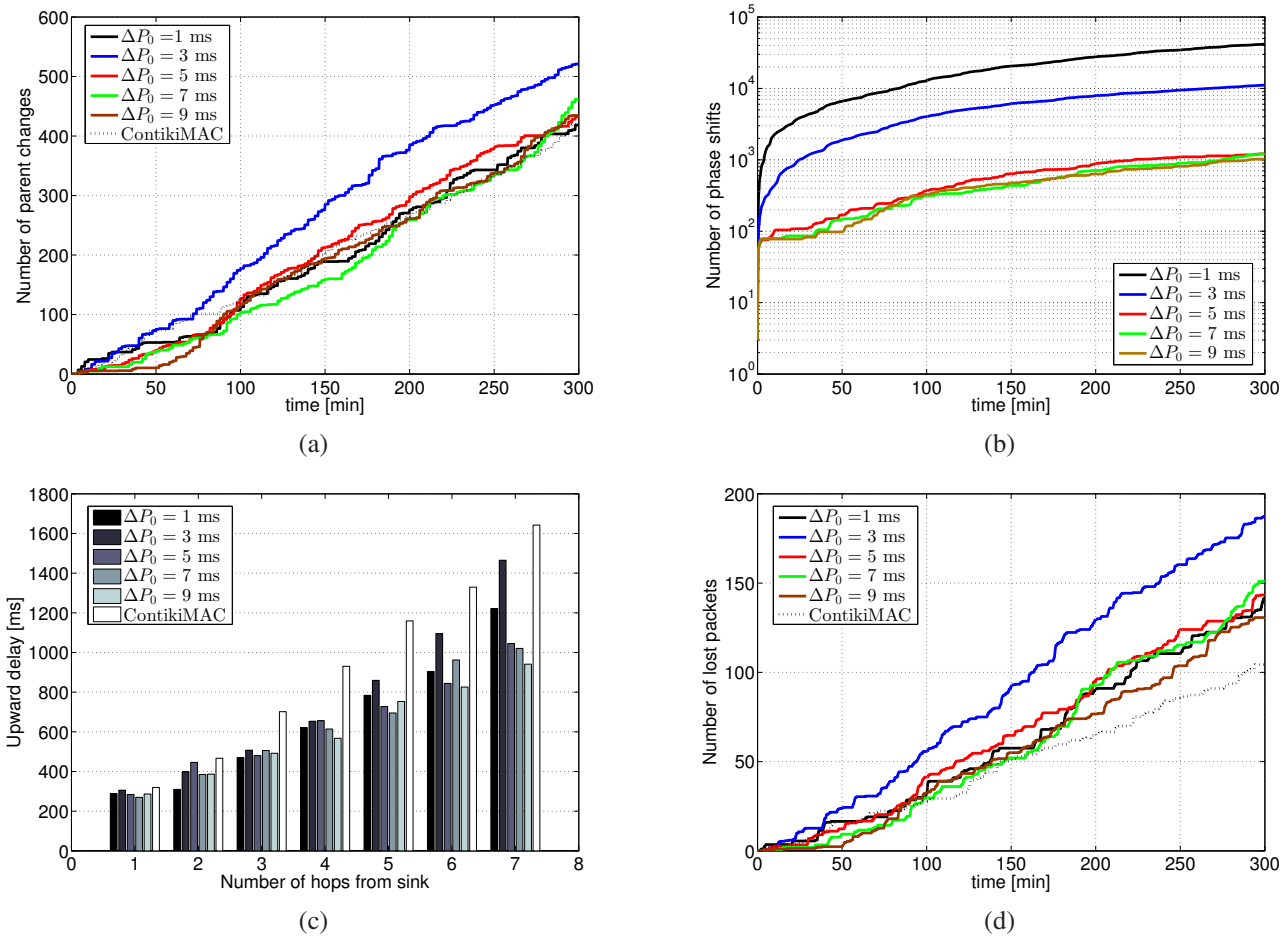


Fig. 7. (a) Parent changes as a function of the simulation time, (b) phase shifts as a function of the simulation time, (c) upward delay as a function of the RPL tree depth, and (d) packet losses as a function of the simulation time, for different values of the delta phase offset ΔP_{o}. Performance is compared with ContikiMAC.

In Fig. 7 (a), the cumulative number of parent changes experienced by the nodes in the network is shown as a function of simulation time, considering several values of ΔP_{o}. Simulations have been carried out with $P_{\mathrm{o}} = 40$ ms. Focusing on the values of ΔP_{o} which guarantee the best performance, namely, $\Delta P_{\mathrm{o}} = 5 \div 9$ ms, it can be observed that the number of RPL parent changes with RAWMAC is the same as with ContikiMAC. Therefore, it can be concluded that these changes are caused by the instability of the DAG rather than by the impact of phase shifts on the RPL metric.

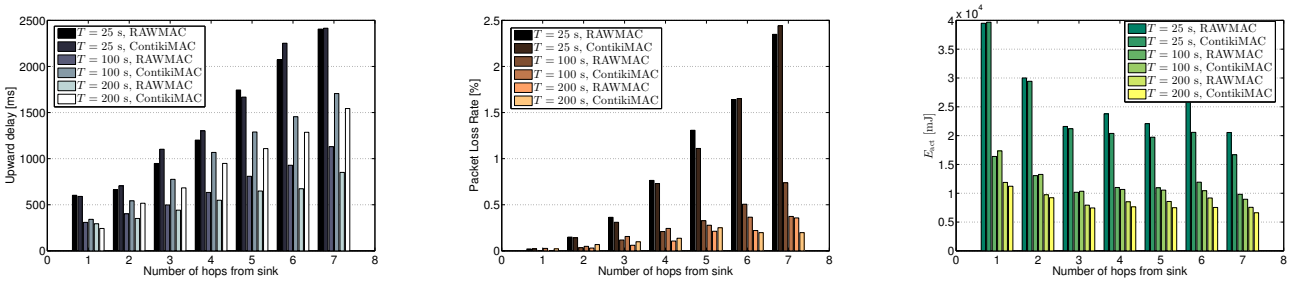
In Fig. 7 (b), we evaluate the number of phase shifts that are experienced by the nodes as a function of the simulation time. The number of shifts for $\Delta P_{\mathrm{o}} = 5 \div 9$ ms is basically the same, whereas for $\Delta P_{\mathrm{o}} = 1 \div 3$ ms the number of phase shifts is one order of magnitude larger than for the other values of ΔP_{o}. This means that the nodes continuously try to track their parents, resulting in worse delay performance, as shown in Fig. 7 (c), where the upward delay is shown as a function of the depth of the RPL DAG. The effect of the phase offset threshold becomes more evident the longer is the number of hops, resulting in a higher delay. In this case, the delay is due to the misalignments introduced by the nodes trying to track the changes of their parents' phase offsets.

In Fig. 7 (d), the packet loss rate is shown as a function of simulation time. The losses introduced by the presence of "waves" are really limited. Even for small values of ΔP_{o} the packets only experience higher delay but no supplementary losses with respect to those experienced with ContikiMAC.

D. Impact of the Packet Generation Period T

In order to measure the impact of traffic load on RAWMAC, we vary the packet generation period T from 25 s up to 200 s. According to previous results, P_{o} and ΔP_{o} are set to 40 ms and 6 ms, respectively. In Fig. 8 (a) the upward delay is shown as a function of the distance of the nodes from the sink. For small values of T, the performance of RAWMAC and ContikiMAC is the same since the delay is mainly due to retransmissions introduced by the access contention mechanism. When T increases, i.e., in low traffic conditions, RAWMAC outperforms ContikiMAC due to the propagation waves that allow to quickly deliver packets to the sink.

This behavior is confirmed by the packet loss rate shown in Fig. 8 (b). For small values of T, the traffic offered to the WSN is too high, so nodes that are far from the sink (i.e., deeper) experience higher losses, and packets are dropped



(a) (b) (c)

Fig. 8. (a) Average delay upward, (b) packet losses, and (c) energy consumption as functions of the hop distance, for various values of the packet generation period T. Performance is compared with ContikiMAC.

since nodes fail to retransmit. On the other hand, when T is large, the packet loss rate for the two protocols is comparable.

From an energy point of view, as shown in Fig. 8 (c), the nodes closer to the sink have a larger energy consumption since they have to relay information coming from their children. When T is small, the high energy consumption is not only due to the large number of packets to be forwarded, but also to the large number of packets' retransmissions which are due to collisions.

VI. Conclusions and Perspectives

In this paper we have presented RAWMAC, an adaptation layer for minimizing the alert collection delay in monitoring systems. RAWMAC uses routing as a management module for asynchronous MAC layers, which are more robust to clock drifts, in order to create a wave of propagation from the leaves to the root of a routing tree. We have exemplified this adaptation layer on RPL and ContikiMAC, as these routing protocols are emerging as standards de facto for the Internet of Things. Performance results have shown that RAWMAC largely outperforms the standalone ContikiMAC in terms of delay for upward traffic. From energy and packet losses points of view, instead, RAWMAC under the same configuration parameters has the same performance of pure ContikiMAC. This means that, given a target maximum delay, RAWMAC allows to largely reduce the duty-cycle of the nodes compared to ContikiMAC, resulting in a remarkable energy saving. In addition, we have studied the impact of the parent changes due to RPL and of the misalignment due to clock drifts. RAWMAC proves to be sufficiently reactive to track all these changes without affecting the packet delivery ratio, that remains equivalent to that of ContikiMAC.

As future work, we plan to extend RAWMAC to take into account efficient broadcasting exploiting the mechanism for the creation of waves introduced by RAWMAC. Another relevant research direction is the design and performance evaluation in the presence of heterogeneous radio duty-cycling across the nodes.

Acknowledgement

This work is partly funded by the European Commission, under Grant no. 288879, CALIPSO project - Connect All IP-based Smart Objects. The work reflects only the authors' views; the European Community is not liable for any use that may be made of the information contained herein. This work is also supported by the French National Research Agency (ANR) under grant reference IRIS ANR-11-INFR-0016.

References

[1] M. Doudou, D. Djenouri, and N. Badache, "Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 528–550, Second Quarter 2013.

[2] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[3] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, Dec. 2011. [Online]. Available: http://www.sics.se/~adam/dunkels11contikimac.pdf

[4] M. Doudou, D. Djenouri, N. Badache, and A. Bouabdallah, "Synchronous contention-based {MAC} protocols for delay-sensitive wireless sensor networks: A review and taxonomy," *Journal of Network and Computer Applications*, vol. 38, no. 0, pp. 172 – 184, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804513000994

[5] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency mac for tree-based data gathering in sensor networks: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 7, no. 7, pp. 863–875, Sep. 2007.

[6] A. Cabezas, R. Medina, N. Pea T, and M. Labrador, "Low energy and low latency in wireless sensor networks," in *Proc. of ICC*, 2009.

[7] P. Medagliani, G. Ferrari, V. Gay, and J. Leguay, "Cross-layer design and analysis of wsn-based mobile target detection systems," *Ad Hoc Networks*, no. 2, Mar. 2013.

[8] B. Pavkovic, W.-J. Hwang, and F. Theoleyre, "Cluster-directed acyclic graph formation for ieee 802.15.4 in multihop topologies," in *Proc. of NTMS*, 2012.

[9] P. Guo, T. Jiang, Q. Zhang, and K. Zhang, "Sleep scheduling for critical event monitoring in wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 2, pp. 345–352, Feb 2012.

[10] X. Vilajosana, P. Tuset-Peiro, F. Vazquez-Gallego, J. Alonso-Zarate, and L. Alonso, "Standardized low-power wireless communication technologies for distributed sensing applications," *Sensors*, vol. 14, no. 2, pp. 2663–2682, 2014. [Online]. Available: http://www.mdpi.com/1424-8220/14/2/2663

[11] P. Gonizzi, R. Monica, and G. Ferrari, "Design and Evaluation of a Delay-Efficient RPL Routing Metric," in *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference* (IWCMC 2013), Cagliari, Italy, July 2013.

[12] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proc. of IEEE LCN*, 2006.