Improving the Energy Efficiency of WSN by Using Application-Layer Topologies to Constrain RPL-defined Routing Trees

Bruno F. Marques

Departamento Engenharia Eletrotécnica
Escola Superior de Tecnologia e Gestão
Instituto Superior Politécnico de Viseu, Portugal
bmarq@estgv.ipv.pt

Manuel P. Ricardo

INESC Porto
Faculdade de Engenharia da
Universidade do Porto, Portugal
mricardo@inescporto.pt

***Abstract*—The deployment of thousands of tiny devices inter-networked together and accessible through the Internet is the result of the increasing trend towards enabling the concepts of *Internet-of-Things*. As these devices may be scattered in a unplanned way, a routing protocol is needed. The RPL protocol is the IETF proposed standard protocol for IPv6-based multi-hop *Wireless Sensor Networks* (WSN). *RPL* requires that communication paths go through a central router which may provide suboptimal paths, making no distinction of the applications the nodes run. To address these issues, an *Application-Driven* extension to *RPL* is proposed which enables the increase of the WSN lifetime by limiting the routing and forwarding functions of the network mainly to nodes running the same application. This paper evaluates the proposed solution coded in *ContikiOS* by means of *Cooja* simulations, and compares it against regular *RPL*. Simulation results confirm that the proposed solution provides lower energy consumption, lower end-to-end delays, and lower total number of packets transmitted and received.**

***Index Terms*—Internet-of-Things (IoT); Wireless Sensor Network (WSN); Application-Driven WSN; RPL; ContikiOS.**

I. Introduction

The growth of wireless networks has resulted in part from requirements for connecting people and advances in radio technologies. Recently there has been an increasing trend towards enabling the Internet-of-Things (IoT) [1]. Thousands of tiny devices interacting with their environments are inter-networked together and accessible through the Internet. For that purpose, several communications protocols have been defined making use of the IEEE 802.15.4 Physical and MAC layers [2]. The 6LoWPAN Network Layer adaptation protocol [3] is an example which bridges the gap between low power devices and the IP world. Since its release, the design of routing protocols became increasingly important [4] and RPL [5] emerged as the IETF proposed standard protocol for IPv6-based multi-hop WSN. Our work is focused on the design of an extension of the RPL routing protocol with the purpose of making the network aware of the traffic applications generate. We assume that sensors form a large IPv6 network and that a sensor is enabled to run one or more applications. We also assume that the applications and the nodes to which they are associated, are not always active, alternating between *on* and *off states*. By jointly considering the neighbors of each node, the applications each node runs, and the forwarding capabilities of a node, we developed a communication solution which enables the data of every application and node to be transferred while keeping the overall energy consumed low. In our solution (*RPL-BMARQ*) we assume that every node will primarily select its parent from a set of nodes running the same application to which the data is associated. For that purpose, the application layer of each node shares information with other layers of the communication stack. The main contribution of this paper is a *cross-layer solution*, characterized as an extension to the RPL routing protocol, which uses information shared by the application and routing layers to construct *Directed Acyclic Graphs* (DAGs), allowing the nodes to select parents with respect to the applications they run. The contribution helps reducing the network energy consumption since it restricts radio communication activities while maintaining throughput fairness and packet reception ratio high. The paper is organized in six sections. Sec. II presents related work. Sec. III describes the rationale of our solution. Sec. IV

978-1-4799-5258-8/14/$31.00 ©2014 IEEE

characterizes the applications and scenarios selected for the study. Sec. V describes the methodology adopted for validating the proposed WSN solution and discusses the results obtained from the simulations performed. Finally, Sec. VI draws the conclusions and presents future work.

II. Related work

RPL [5] is a routing protocol that organizes nodes along a *Destination Oriented Directed Acyclic Graph* [6] (DODAG), normally rooted at a border router node or at a sink node. As described in [7], the DODAG root initiates the DODAG formation by periodically sending *DODAG Information Object* (DIO) messages which it advertises by using link-local multicast addresses [8]. DIO messages carry information such as the DODAG root's identity, the routing metrics in use, and the rank of the originating node *rank* in the DODAG. A node joins the DODAG by taking into consideration these factors, and determines its own rank in the DODAG based on the information advertised by its neighbors in their DIOs. The node chooses as parent in the DODAG the neighbor through which its resulting rank is the smallest. Once a node has joined the DODAG, it sets a path to the root through its parent and can then originate its own DIO messages. So, RPL provides paths from nodes to a root while requiring them to store little forwarding and routing information. In order to keep the size of forwarding and routing tables small, by default RPL does not provide paths from the border router back to the nodes. To overcome this issue, a RPL router that requires a path from itself to a node must send a *Destination Advertisement Object* (DAO) message all the way which records and installs those paths. RPL provides paths that are often much longer than the shortest available paths. The constraint to route only along a DODAG may potentially cause traffic congestion near the DODAG root [8]. Also, the constraint for every possible destination in the DODAG to originate a DAO may be a problem because it is a proactive destination-initiated process which involves sending and receiving to many routing signaling and control messages [8]. The nodes must always be awake in order to send, receive and process those messages. To overcome these issues, our solution tries to choose mainly the nodes running the same applications to form the DAGs, and to put the nodes in the *on* and *off* states in a synchronized manner.

III. RPL-BMARQ rationale

We define *Application-Driven WSN* (ADWSN) as a cross-layer solution aimed to help reducing the energy consumed by a network of sensors executing a set of applications. This paradigm assumes that each application defines its own network and set of nodes so that the exchanged information can be confined to the nodes associated with the application. The nodes share information about the applications they run, and also their duty-cycles. Our solution, *RPL-BMARQ*, stands for ***RPL By Multi-Application ReQuest***. It tries to insure that data of an application is relayed mainly by the nodes running that application. When sink nodes query the other nodes, routing paths should involve preferentially nodes running the same application.

```
if (neighbor == is_root()) then
    add_parent(neighbor);
else
    if (neighbor.app_id == node.app_id && neighbor.rank <
    node→neigbors.rank) then
        add_parent(neighbor);
    else
        start(parent_selection_timer);
        while (parent_selection_timer) do
            if neighbor→neighbor.app_id == node.app_id
            && neighbor.rank < node→neigbors.rank) then
                add_parent(neighbor);
                reconfigure(neighbor(app_id,
                app_duty-cycle));
                advertise_changes();
                stop(parent_select_timer);
                has_parent = True;
            end
        end
        if (has_parent == False) then
            if (neighbor.rank < node→neigbors.rank)) then
                add_parent(neighbor);
                reconfigure(neighbor(app_id,
                app_duty-cycle));
                advertise_changes();
            end
        end
    end
end
```

Algorithm 1: *Pseudocode of RPL-BMARQ DAG creation*

The DAG creation scheme will use mainly the nodes running the same application; the nodes not associated to the application will not participate in DAG creation process, in a first attempt. Algorithm 1 shows how to create DAGs in our solution. A root node starts to create the DAG by sending DIO messages. The nodes surrounding the root use the information carried in these DIO messages, compute their rank and join the DAG, in the same way of regular RPL. The computed rank is jointly sent with the identification of running applications and their duty-cycles, in DIO messages. This information is used by other nodes to update their *neighbor tables*, compute their own rank, and advertise their presence by sending new DIO messages. All node's *neighbor tables*

record the neighbors IP address, the applications they run and their duty-cycles. This information is used to help the node to join the DAG, by looking into it's *neighbor table*. If the node runs the same application of its neighbor and if the latter has a lower rank, the former may choose it as parent, joining the DAG through it. If the node has no neighbors running the same application, a neighbor with lower rank can be reconfigured as running the application, and selected to be the parent. If the timer expires and a node could not be chosen as a parent, the scheme reverts to the regular RPL DAG formation.

IV. RPL-BMARQ EVALUATION

A. Applications Characterization

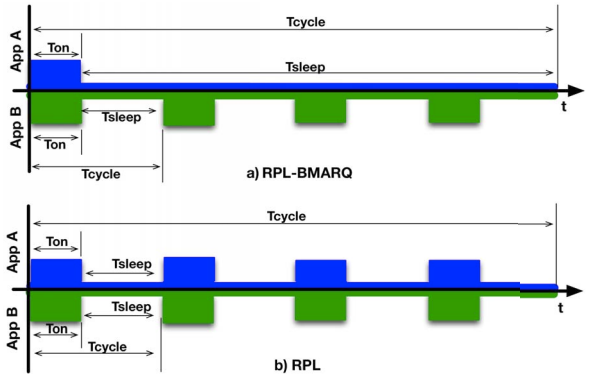

Fig. 1. Applications duty-cycle

Two different applications are used in our experiments. These applications and their topology are characterized by the following aspects: static, organized, pre-planned, no mobility, 16 nodes deployed in a square lattice topology, all sensor nodes battery-powered, except for the sink nodes. We consider multi-hop communication. The traffic pattern is point-to-multipoint when data is queried, and point-to-point when queries are replied. The first application (App. A) has a duty-cycle of one hour. Every node running this application wakes every hour remaining awake during one minute for activity replying to asked data if it is a sensor node. The second application (App. B) has a duty-cycle of 15 minutes. The nodes also wake during one minute to sense data and to communicate. As shown in Fig. 1 a), the period of App. A is 4 times the period of App. B. All sensor nodes are expected to be awake when data is queried and replied, and sleeping when there is no activity.

B. Simulation scenarios

In order to evaluate the *RPL-BMARQ* solution, a square lattice of 4x4 nodes was simulated. The nodes are distributed as shown in Fig. 2, where the four scenarios evaluated are also shown. All the nodes are within a distance of 25 meters for a transmission range of 30 meters, and support one of the two applications. Each application is running in eight nodes, and each node runs a single application. Sink nodes placements where chosen in order to allow long routing paths, since long paths consume more energy. In *Scenario 1*, the nodes running App. A were selected so that a long path could be obtained. In the *Scenario 2*, both applications have the same node distribution; in these scenarios we want to investigate the influence of the application duty-cycle in energy consumption. *Scenarios* 3 and *4* are used to investigate situations where at least one node from other application is required to relay data. In the scenarios evaluated, sink nodes are always awake, and sink node running App. B (node 9) was chosen as DAG root because of it's application duty-cycle. Since our solution constrains the paths to the nodes associated to the application, a node needs to be woken up only when its applications run and not for generic routing and forwarding purposes. In contrast, RPL was used as shown in Fig. 1 b); in this case despite the applications having different periods, all the nodes would have to wake up every 15 minutes in order to process routing messages. In our solution, a query is *"multicasted"* only to the nodes associated to the application and not the entire WSN. When the application nodes reply, only the nodes running the application will send and forward *unicast* packets. So, using the *RPL-BMARQ solution*, the routing paths are chosen not only according to RPL objective functions, but also considering the nodes belonging to the application for which the paths are required, and the total number of neighbors a node has, at least in the first attempt. Fig. 2 c) shows a particular node deployment, where nodes 10, 13, and 16 running application B are unable to receive sink queries because they are isolated. In this case, node 8 would be selected to participate in the routing and forwarding process. Fig. 2 d) shows a second case of node deployment where some of the nodes of App. A (nodes 7 and 8) are also isolated. In order to study the behavior of the RPL and the *RPL-BMARQ* solutions, both have been implemented on *ContikiOS* [9], which is an operating system used for wireless sensor networks. *ContikiOS 2.6* was chosen because it includes an IPv6 stack with *6LoWPAN* support, as well as *ContikiRPL*, which is an basic RPL implementation. The communication between the nodes is achieved using the *CSMA-CA* physical access scheme with *NullRDC*, which means that the *Mac Layer*, by it-self, does not put the nodes in the *on* and *off* states. The simulations were

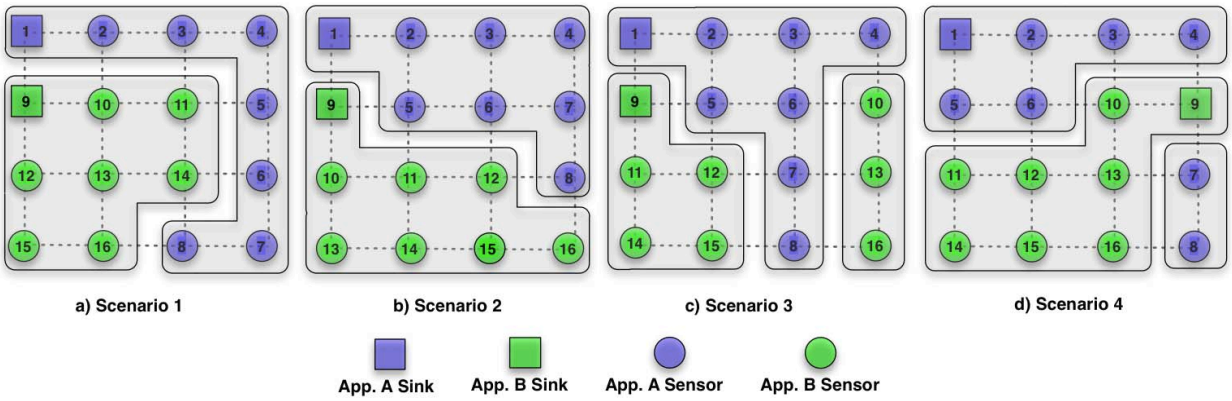

Fig. 2. Nodes deployment in different square lattice mesh topologies

performed using *Cooja* [10], a simulator for *ContikiOS*, which allows developers to test code and systems before running it on a target hardware. The hardware platform selected was *TelosB* [11] which uses IEEE 802.15.4 radios. The 2.4 GHz radio medium model chosen was the *Unit Disk Graph Medium* (UDGM). UDGM models the transmission range between the nodes as an ideal disk; the nodes behind it do not receive packets, while the nodes within the transmission distance receive all the messages. *UDGM* confirms the functionality and behavior of our solution. We assume that all the nodes within the disk have transmission and reception success ratios of 100%, and packets have a IPv6 Payload of 82 bytes, except for signaling and control ones.

V. Results and Discussion

In order to evaluate the *RPL-BMARQ* and *RPL* solutions we simulated each scenario ten times. Each simulation simulates the applications running for 24 hours. From the results we have ignored the initial 10% of total simulated time in order to have guarantees about system stability, e.g. letting DAGs and routing tables to be stable. We used 95% of confidence intervals [12]. We focused our study on *energy consumption*, *query success ratio*, *fairness*, *delay*, and the total number of *Layer 3* transmitted and received packets. These parameters are defined below.

A. Energy

$$E = E_{TX} + E_{RX} + E_{Idle} + E_{Off} \quad (1)$$

We consider energy consumption related only to communication aspects, eg. packet transmission, reception, *radio idle* the state where a node has its *radio on* and is waiting to send or to receive a data packet, and *radio off*. We aim to investigate how much energy is consumed by the nodes in that states, as shown by Eq. 1. E_{TX} is the total energy consumed when sending packets, E_{RX} is the total energy consumed when receiving packets, E_{Idle} is the total energy consumed by a node when it has the radio in the *idle state*, and E_{Off} is the total energy consumed by a node when it has the *radio off*. Generically, we compute energy E_{state} as $I_{state} \times V \times t_{state}$, considering I_{state} the current consumed by the node in the $state$, V the voltage supplied to the node, and t_{state} the total time the node is in that $state$. The I_{state} and V values depend on existing platforms (e.g. TelosB [11]), and a generic energy model defining the total energy consumed by the nodes is detailed in [13].

	Nominal
Current in Transmit (0 dBm) mode (mA)	19.5
Current in Receive mode (mA)	21.8
Current in MCU on, radio on - idle mode (μA)	365
Current in MCU on, radio off (mA)	1.8
Power supply (V)	3.6

TABLE I
Excerpt from TelosB specification

The results obtained from the simulations are shown in Fig. 3 which shows the total time, for 95% confidence intervals, in which nodes radios were in their different considered states. As it can be seen, the permanence time in each state excluding *radio off* is lower for *RPL-BMARQ* than for *RPL*. The *radio off state* time corresponds to the total time in which radios were turned *off*. Using *RPL-BMARQ*, this value is larger then that using *RPL*, which reflects the major contribution of *RPL-BMARQ* design. Applying the information of Table I, which was extracted from [11], to Eq. 1, we can compute the total energy consumed by the nodes (see Fig. 4).

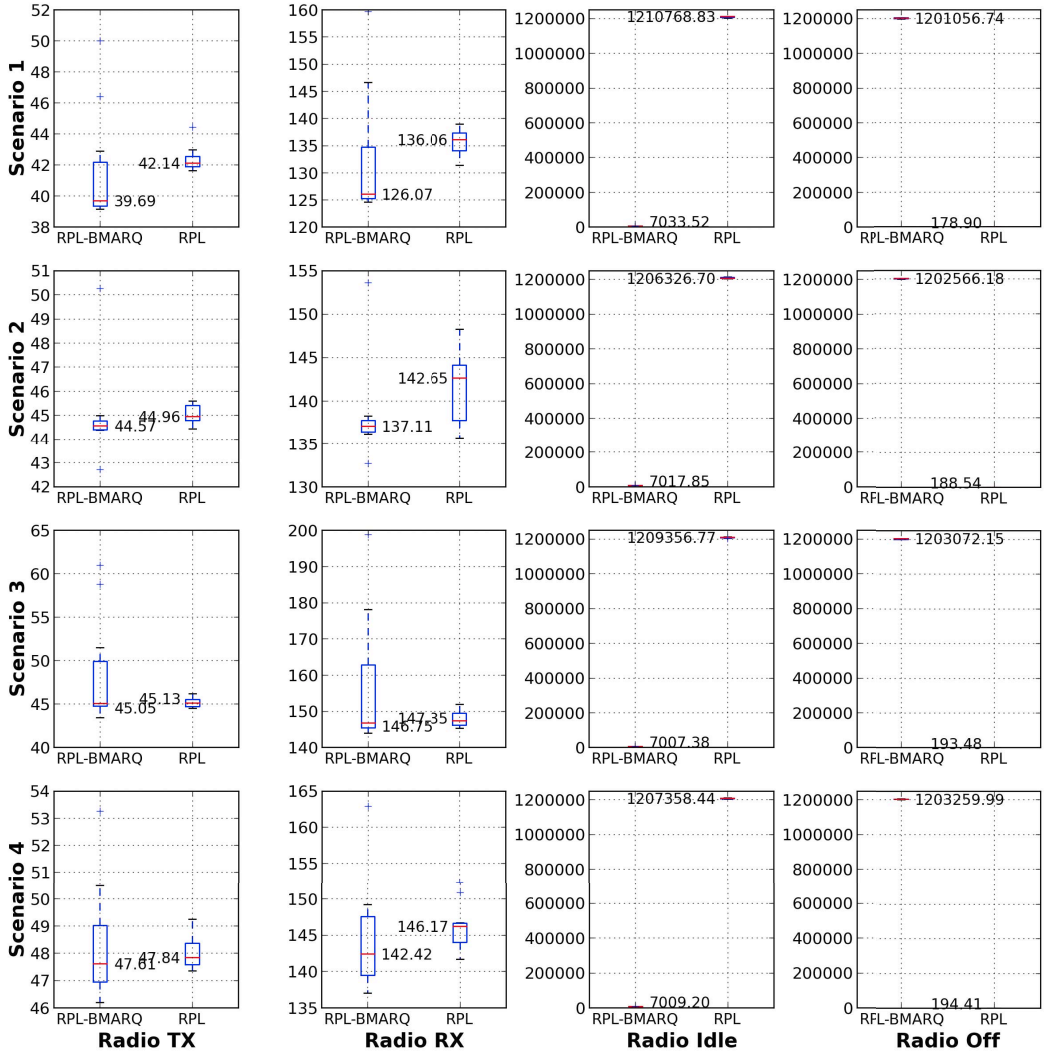

Fig. 3. Mean total radio activity time (in seconds)

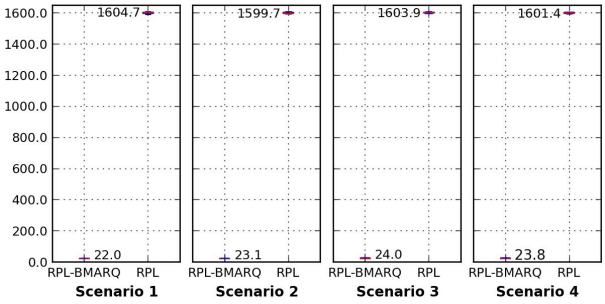

Fig. 4. Energy consumed by each solution in each scenario (in J)

Results show that the total energy consumed by the nodes considering those states, and using the *RPL-BMARQ* solution is very low, as it turns the nodes radio *off* during more time than *RPL*. Using Eq. 2 we compute the energy gain for *RPL-BMARQ*, and present it in Fig. 5.

$$Gain = \frac{E_{RPL} - E_{RPL-BMARQ}}{E_{RPL}} \times 100, \qquad (2)$$

E_{RPL} is the total of energy consumed by the nodes for the *RPL* solution, and $E_{RPL-BMARQ}$ is the total of energy consumed by the nodes for our solution. Results show that, using the *RPL-BMARQ* solution, the nodes spend about 98% less energy than the same nodes running *RPL*, and thus *RPL-BMARQ* extends network lifetime.

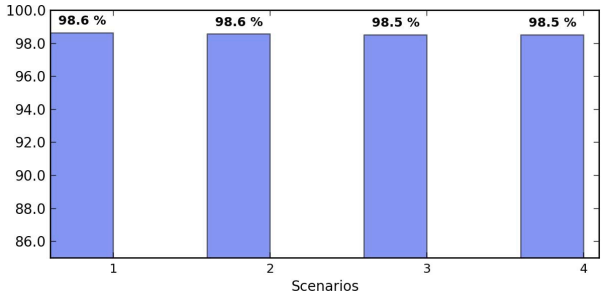

Fig. 5. RPL-BMARQ energy gain in each scenario (in %)

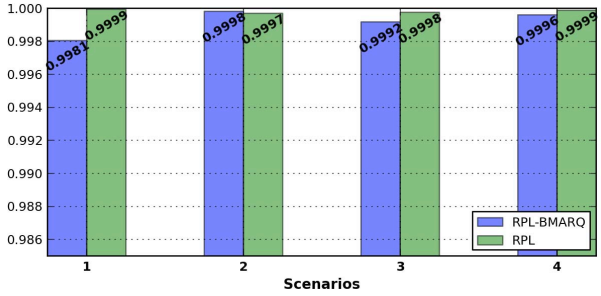

Fig. 7. QSR fairness

B. Query Success Ratio

$$QSR = \frac{number\_of\_received\_replies}{number\_of\_expected\_replies}, 0 \leq QSR \leq 100\% \quad (3)$$

We define *Query Success Ratio* (QSR) as the ratio between the number of reply packets received by a sink node in response to a query packet, and the number of replies the sink expects to receive (see Eq. 3). In our case, $QSR = \frac{number\_of\_received\_replies}{7}$. Fig. 6 shows simulation results with 95% confidence interval. One can conclude that RPL presents better result than *RPL-BMARQ*, except for *Scenario 1*. Please note that in Fig. 6 we are representing *QSR* values between 97 and 100%. We can conclude that with *RPL-BMARQ* QSR does not suffer much.

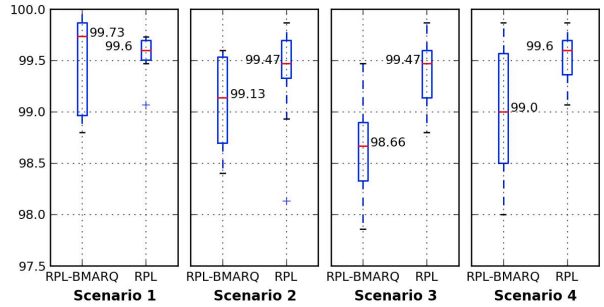

Fig. 6. Mean Query Success Ratio - QSR (in %)

C. QSR fairness

$$\gamma(x_1, x_2, \cdots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2}, 0 \leq \gamma() \leq 1 \quad (4)$$

Jain's fairness index is defined in Eq. 4 [14]. A straightforward computation shows that the fairness measure γ ranges from $\frac{1}{n}$ (maximum unfairness) to 1 (all x_i equal) [12]. In our case, x_i corresponds to the *QSR* per node. Fig. 7 shows *QSR fairness* mean values obtained from simulations for the 4 scenarios. All solutions present fairness indexes above 99%.

D. Delay

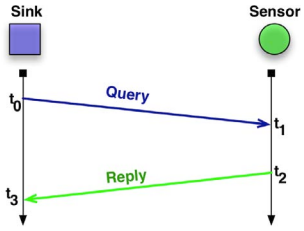

Fig. 8. Mean End-to-End Delay

We define *End-to-End Delay* as the time interval between the time instant a query was sent by a sink node and the time the sink receives the correspondent reply, but excluding the node processing time, as shown in Fig. 8. We compute this delay as $(t_3 - t_0) - (t_2 - t_1)$. Fig. 9 shows 95% confidence intervals delay values for the simulated scenarios using both solutions. We note that with *RPL-BMARQ* we can achieve lower mean delays. For instance in *Scenario 4*, the mean delay value is 994 ms for *RPL-BMARQ* and 999 ms for RPL.

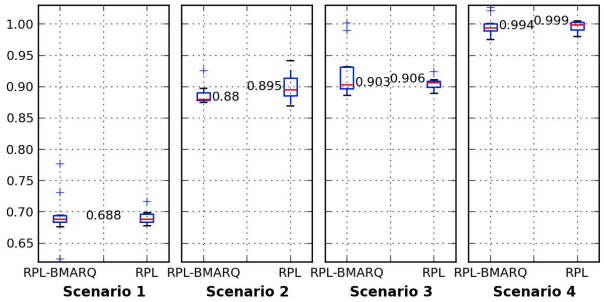

Fig. 9. Mean End-to-End Delay (in s)

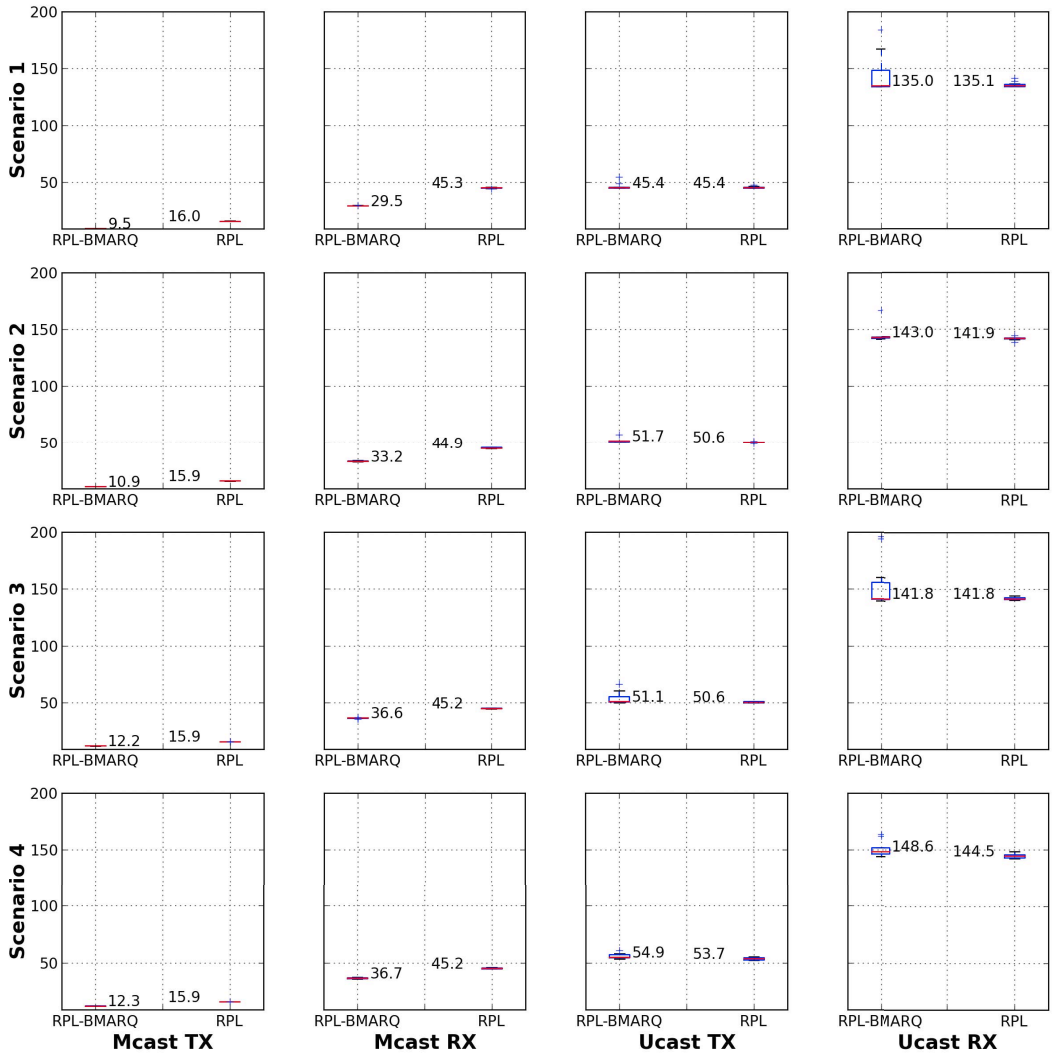

Fig. 10. Mean total number of packets per query

E. Packets per query

In the simulations performed, the sink node running *App. A* generates a total of 21 queries, whilst sink node running *App. B* generates 86 queries. Analyzing the data extracted from simulations we could investigate on a *per-query* basis, how many *Layer 3* multicast and unicast packets were sent and received by all the nodes. We consider also signaling and routing packets. Fig. 10 shows the results for the scenarios simulated using both solutions. For all the scenarios, the *RPL-BMARQ* solution presents lower mean number of total *Layer 3* multicast packets, sent and received. For *Scenario 1*, using *RPL-BMARQ*, and *per-query*, the mean number of total packets sent is 9.5 and the mean number of total packets received is 29.5; using *RPL*, the mean number is higher (16 packets sent and 45.3 packets received). For *Scenario 2*, using *RPL-BMARQ* the mean number of total multicast packets sent is 10.9 and the mean number of total multicast packets received is 33.2; using *RPL*, the mean number of total multicast packets is also higher (15.9 for packets sent and 44.9 for packets received). For *Scenario 3*, using *RPL-BMARQ* the mean number of total multicast packets sent is 12.2 and the mean number of total multicast packets received is 36.6; using *RPL*, the mean number of total multicast packets is also higher (15.9 packets sent and 45.2 packets received). Finally, for *Scenario 4*, using *RPL-BMARQ* the mean number of total multicast packets sent is 12.3 and the mean number of total multicast packets received is 36.7; using *RPL*,

the mean number of total multicast packets also higher (15.9 packets sent and 45.2 packets received). For the mean number of total *Layer 3* unicast packets sent and received, the results show that both solutions present very close numbers. For *Scenario 1* the mean number of total unicast packets sent using both solutions is 45.4; for the same scenario, using *RPL-BMARQ* the mean number of total unicast packets received is 135, whilst using *RPL* the same is 135.1. For *Scenario 2* the mean number of total unicast packets sent is 51.7 for *RPL-BMARQ* and 50.6 for *RPL*, and the mean number of total unicast packets received is 143 for *RPL-BMARQ* and 141.9 for *RPL*. For *Scenario 3* the mean number of total unicast packets sent is 51.1 for *RPL-BMARQ* and 50.6 for *RPL*, and the mean number of total unicast packets received is the same for both solutions (141.8). Finally, for *Scenario 4* the mean number of total unicast packets sent is 54.9 for *RPL-BMARQ* and 53.7 for *RPL*; the mean number of total unicast packets received is 148.6 for *RPL-BMARQ* and 144.5 for *RPL*. From above analysis we conclude that the major gain of the *RPL-BMARQ* solution relies on the total number of multicast packets sent and received which is lower than the equivalent *RPL*. On the other hand, the number of transmission and reception of unicast packets are higher. We suspect of packet collision, which causes more packet retransmissions.

VI. Conclusions and future work

In this paper we propose a solution (*RPL-BMARQ*) to deploy WSN defined by the applications the nodes run. The paper presents and discusses the performance of RPL and *RPL-BMARQ* by means of simulating sensor devices using *IEEE 802.15.4* radios. The experiments were conducted by means of simulations using *ContikiOS* and *Cooja*, and confirmed that the proposed solution reduces overall network energy consumption, thus increasing the network lifetime. *RPL-BMARQ* reduces the total number of transmission and reception of multicast packets, constrains the paths mainly to the nodes running the same application, and decreases delays. Further experiments with *RPL-BMARQ* need to be performed in a real environment to help identify shortcomings of the solution currently designed.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645 – 1660, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[2] IEEE-Computer-Society, "Ieee std 802.15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans)," September 2006, revision of IEEE Std 802.15.4-2003.

[3] G. Montenegro, N. Kushalnagar, and D. Culler, "Transmission of ipv6 packets over ieee 802.15.4 networks," September 2007, iETF.

[4] C.-M. Tang, Y. Zhang, and Y.-P. Wu, "The p2p-rpl routing protocol research and implementation in contiki operating system," in *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, 2012, pp. 1472–1475.

[5] 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 (Proposed Standard), Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[6] J. Martin, "Oper. Res." *Oper. Res.*, vol. 13, pp. 44–66, 1965.

[7] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi, "A Performance Analysis of Point-to-Point Routing along a Directed Acyclic Graph in Low Power and Lossy Networks," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, 2010, pp. 111–116.

[8] E. Baccelli, M. Philipp, and M. Goyal, "The P2P-RPL routing protocol for IPv6 sensor networks: Testbed experiments," in *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, 2011, pp. 1–6.

[9] A. Dunkels, "Contiki OS, open source, highly portable, multitasking operating system for memory-efficient networked embedded systems and wireless sensor networks ," available at http://www.contiki-os.org.

[10] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, 2006, pp. 641–648.

[11] "Wireless Modules, TelosB," available at http://www.xbow.com/Products/productdetails.aspx?sid=252.

[12] J. Boudec, *Performance Evaluation of Computer and Communication Systems*, ser. Computer and communication sciences. EFPL Press, 2010. [Online]. Available: http://books.google.pt/books?id=nibpCdEjUEYC

[13] B. Marques and M. Ricardo, "Application-Driven design to extend WSN lifetime," in *Proc. 1st Portuguese National Conference on Sensor Networks (CNRS2011), Coimbra, Portugal*, March 2011.

[14] R. Jain, D.-M. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," *CoRR*, vol. cs.NI/9809099, 1998. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr9809.html#cs-NI-9809099

Acknowledgment

This work is financed by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «MC-WMNs/PTDC/EEA-TEL/120176/2010», and within the fellowship «SFRH/BD/36221/2007». Authors would like to thank also the support from Faculty of Engineering, University of Porto, to thank the support from the INESC Porto, and to thank the support from the School of Technology and Management of Viseu's Electrical Engineering Department.