RPL Modifications to Improve the End-to-End Delay Estimation in WSN

Pedro Pinto
ESTG, Instituto Politécnico de
Viana do Castelo and INESC TEC
Viana do Castelo and Porto, Portugal
Email: pedropinto@estg.ipvc.pt

António Pinto
CIICESI, ESTGF, Politécnico do
Porto and INESC TEC
Porto, Portugal
Email: apinto@inescporto.pt

Manuel Ricardo
INESC TEC, Faculdade de Engenharia,
Universidade do Porto
Porto, Portugal
Email: mricardo@inescporto.pt

Abstract—**Real-time monitoring applications deployed in Low-power and Lossy Networks may generate flows sensitive to delay, where the information is useful for the destination only if it is received within a strict delay boundary. Data packets that will likely miss the application deadline could be discarded during their routing through the network or even be not transmitted at all, thus contributing for a better usage of the network resources.**

This paper presents RA-EEDEM, a set of modifications made to RPL that improve the End-to-End Delay (EED) estimation accuracy. The RA-EEDEM modifications include changes to the RPL metrics and to its Objective Function (OF). The results show that RA-EEDEM improves the accuracy of EED estimation while minimizing its impact on the average EED and Packet Reception Ratio (PRR).

Keywords*—*End-to-End Delay; Delay Estimation; LLN; RPL;

I. Introduction

Real-time monitoring applications can be deployed in Low-power and Lossy Networks (LLNs). These applications are expected to generate real-time data flows that may require different levels of service from the network. In the case of delay sensitive flows the transported data is assumed to be useful for the destination only if it is received within a strict delay boundary, and useless otherwise. Let us assume a LLN where application A running on node 1 sends a data packet to its peer in node 2, and that this data packet must reach the application in node 2 within a maximum delay of 150 ms. If node 1 estimates a delay of 300 ms on the path to node 2 it may avoid the transmission of this packet. Since the nodes used in a LLN have low processing resources and energy constraints, the discard of this useless packet contributes in fact for the enhancement of the overall network performance and for the efficient use of the deployed resources. Thus, it is important to obtain the most accurate application-to-application delay estimation in order to correctly identify the data packets that will likely miss their applications deadlines.

In a previous work [1] we proposed an End-to-End Delay Estimation Mechanism (EEDEM) used for delay sensitive monitoring applications in a Wireless Sensor Network (WSN). EEDEM estimates the End-to-End Delay (EED) based on the internal delay experienced by previously sent packets and delay information from other nodes through the use of Routing Protocol for Low-power and Lossy Networks (RPL). EEDEM depends on the RPL operation; a high refresh rate of routing messages increases the estimation accuracy but it also increases the RPL overhead causing EED to become less predictable.

This paper presents RPL Adaptation for EEDEM (RA-EEDEM) aimed to improve the EED estimation accuracy and minimize the impact on the average EED and PRR. RA-EEDEM includes changes to the RPL metrics and its OF procedures. This work was carried out in the scope of the SELF-PVP project [2] that aims to increase the efficiency of a photo voltaic power plant where solar panels communicate with each other using a WSN in a grid topology.

The structure of this paper is as follows. Section II identifies the related work. Section III describes the basic operation of RPL. Section IV describes RA-EEDEM proposal. Section V presents the simulation environment used to validate the current proposal. Section VI discusses the obtained results. Section VII concludes paper.

II. Related Work

The real-time estimation of EED can be performed using active probing (using specific messages and protocols) or using feedback information delay obtained from normal data traffic or routing protocol messages [3]. Using probe packets to perform EED estimation in LLNs introduces additional traffic, contributing to higher energy consumption and higher loads offered to the network.

An overview of EEDEM, proposed in [1], is presented in Fig. 1. EEDEM estimates per-packet EED based on EED experienced by previous data packets sent along the path from the senders application layer to the destination's application layer. Nodes account internal delays and each node uses RPL [4] to transmit the delay values of the other nodes on the path from the sink to the generator node. When a generator node i has to send a packet to a sink s, it performs an EED estimation based on the collected information. The node internal delays include the Internal Processing Delay (IntProcD) which is the time elapsed while the packet is processed within the stack of the generator node, and the Internal Link Delay (IntLinkD) which is the time elapsed in MAC layer queuing and packet transmission to the next intermediate node. The RPL feedback consists of two different metrics: the Processing Delay Metric (ProcDelayM), which represents the cumulative processing delays up to the sink, and the Path Delay Metric (PathDelayM) which represents the cumulative link delays up to the sink.

978-1-4799-5863-4/14/$31.00 © 2014 IEEE

Thus, the generator node i estimates the EED towards the sink s for the next packet n, with information given by a preferred parent p using:

$$EstimatedEED_{is}^{n} = EEPathD_{ips}^{n-1} + EEProcD_{ips}^{n-1} \quad (1)$$

where:

$$EEPathD_{ips}^{n-1} = IntLinkD_{ip}^{n-1} + PathDelayM_{ps} \quad (2)$$

$$EEProcD_{ips}^{n-1} = IntProcD_{ip}^{n-1} + ProcDelayM_{ps} \quad (3)$$

The RPL metrics above are additive and set to enable routing decisions related to the lowest delay. These metrics are also used to transport the delay information in reverse direction of the data flow in order to enable real-time EED estimation. Such mode of operation makes this EED estimation mechanism highly dependent on the routing protocol operation. The combination of these metrics with the standard RPL operation introduces undesired overhead, which was not studied in [1].

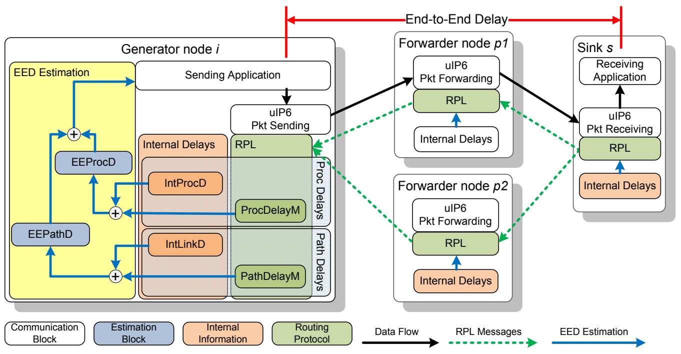

Fig. 1. EED estimation mechanism overview

Other research efforts evaluate RPL performance and propose modifications to improve its operation under specific scenarios. In [5] authors present a performance study focused on RPL improvements and its use in real-life scenarios. In [6] is presented an experimental evaluation of RPL in terms of delivery ratio, control packet overhead, dynamics and robustness, using large scale testbeds. In [7] are proposed two adaptive algorithms to the control RPL Destination Advertisement Option (DAO) messages in order to reduce congestion and packet drops in Urban LLNs.

III. RPL Operation

RPL [4] is a routing protocol defined by IETF Routing Over Low-power and Lossy networks (ROLL) working group. RPL is defined for LLNs where devices have processing power, memory and energy constraints. Multiple instances of RPL may run in a single network; in each instance, the nodes are organized into a tree named Destination-Oriented Directed Acyclic Graph (DODAG), or simply DAG. Each DAG has a root where all paths terminate. The DAG formation is done according to an OF that defines how the used routing metrics are translated into a rank. The values in rank represent the nodes position up to the DAG root and in relation to the remaining nodes.

A. RPL Control Messages

The DAG Information Solicitation (DIS), DAG Information Object (DIO), and Destination Advertisement Object (DAO) messages are used to create and maintain the routing information in each node. Other messages are also defined but these are considered out of the scope of the current proposal. Fig. 2 shows the RPL control messages dynamics.

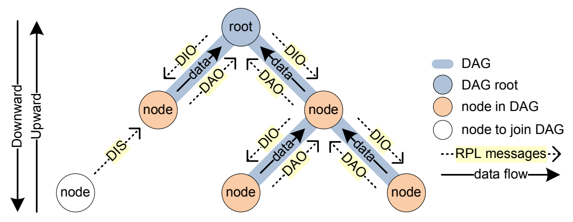

Fig. 2. RPL control messages and data flow dynamics

In order to form a DAG, the DAG root multicasts DIO messages, carrying information of the RPL instance and DAG configuration parameters. This enables the joining of nodes to the DAG, the selection of a parent and participation in the DAG. Instead of waiting to receive the next DIO message, nodes wanting to join an already formed DAG can multicast a DIS message, requesting information (DIOs) from the other RPL nodes. After receiving a DIO from a candidate parent, the node calculates the cost of the path to reach this parent and to the DAG root, taking into account the path cost information received in the DIO. When multiple candidate parents exist, a preferred parent is elected based on the lowest cost to the DAG root. After joining a DAG, the node can start sending or forwarding data, in upwards direction, towards the DAG root.

Although RPL was designed to support multipoint-to-point (MP2P) upwards communications between multiple devices and the DAG root, point-to-multipoint (P2MP) and point-to-point (P2P) communications are also supported in reverse direction. In order to support downward routes, unicast DAO messages can be sent from a child to a preferred parent in order to propagate destination information (addresses and prefixes) in upwards direction.

B. Routing Metrics and OF

All the nodes in a given DAG are configured to support a set of metrics. The values of these metrics are transported through the DAG Metric Container option within the DIO or DAO messages. In [8], the ROLL working group proposes a set of routing metrics, managed by IANA. These metrics can be additive, maximum, minimum or multiplicative.

Within a given DAG, the OF defines how the metrics are converted into a rank value, i.e. a value representing the distance/cost to the DAG root. The OF also defines how a node selects its parents. In this context, up to now, the ROLL working group has defined two OF: OF Zero (OF0) [9] and Minimum Rank with Hysteresis OF (MRHOF) [10]. In OF0 a node will always choose the parent with the lowest rank. In MRHOF the decision to change parents depends not only on the lowest rank but also on a hysteresis value. Nodes will switch to a new parent if its rank is lower than the current one by at least a given hysteresis value.

IV. RA-EEDEM Proposal

RA-EEDEM is based on a set of RPL modifications that were applied and tested within ContikiRPL [11], an open-source RPL implementation integrated in the ContikiOS [12]. ContikiRPL was used with the uIPv6 stack [13] and, at layer 2, with CSMA and ContikiMAC [14].

Since routing protocol overhead has impact in EEDEM results, RA-EEDEM aims to balance the rate of RPL control messages. High rates, namely those providing feedback of the delays in downwards direction, will allow for higher estimation accuracy. Since these control messages compete with data messages for the available network resources, high rates of control messages cause the undesirable effect of increasing the average EED of data packets and degradation of their PRR. The envisioned real-time application supporting RA-EEDEM proposal is assumed to only generate data packets in upwards direction towards the DAG root. Thus, the RPL support for downward routes was disabled and DAO messages were suppressed. Also, node mobility was not considered thus DIS messages were neglected since they are only sent during an initial phase, before nodes join the DAG. Our work focused on balancing the rate of the DIO messages, taking into account the following conditions presented by order of importance:

- Maintain the regular routing process for the data packets - The RPL function and stability should not be compromised.
- Assume no specific application data rate - Prior to the deployment, the application data rate is taken as unknown.
- Maximize the accuracy of EED estimation - Improve EED estimation reducing the overhead of the routing protocol.
- Minimize the impact on performance - Minimize the impact on average EED and on PRR performance.

Since application data rate is taken as unknown, the default configuration regarding DIO messages was not changed and the values defined in ContikiRPL were used. DIO messages are sent downwards and are mainly used to transport routing metrics. Such metrics will then be used by OF in order to select the preferred parent, from a set of candidate parents. The metrics already defined in EEDEM (namely PathDelayM and ProcDelayM) are dynamic metrics and assume values with a millisecond precision which may cause fast oscillations on parent selection. Since parent selection instability imposes higher generation rate of DIO messages, the selection of best parent procedure was changed. This procedure is recursive and tests all the candidate parents within sets of two candidate parents (*p1* and *p2*), returning the best one in each round. After testing multiple pairs of candidate parents this procedure outputs a Preferred Parent (PP). The preferred parent is recurrently compared with new pairs of candidate parents.

An overview of the adopted procedure to select best parent is shown in Fig. 3. The first condition imposes this procedure to select parents with existing ProcDelayM values in favor of parents without ProcDelayM. This allows all nodes to quickly obtain processing delays, and thus improve estimation and reduce convergence time. After that, a Total Delay Metric

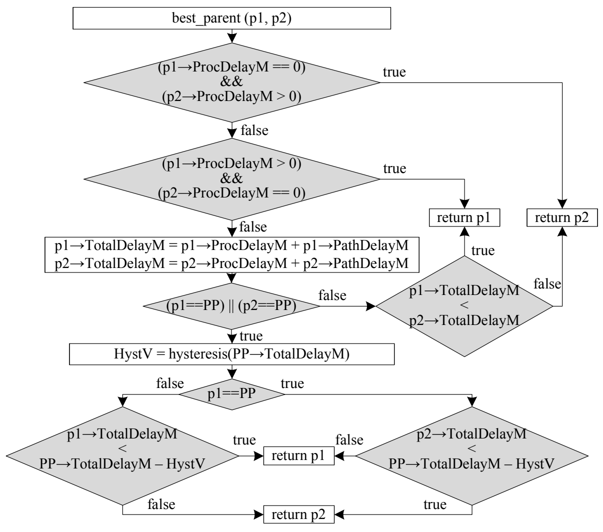
best_parent (p1, p2)
(p1→ProcDelayM == 0) && (p2→ProcDelayM > 0)
true
false
(p1→ProcDelayM > 0) && (p2→ProcDelayM == 0)
true
false
return p1
return p2
p1→TotalDelayM = p1→ProcDelayM + p1→PathDelayM
p2→TotalDelayM = p2→ProcDelayM + p2→PathDelayM
true
(p1==PP) || (p2==PP)
false
p1→TotalDelayM < p2→TotalDelayM
false
true
HystV = hysteresis(PP→TotalDelayM)
false
p1==PP
true
p1→TotalDelayM < PP→TotalDelayM – HystV
true
return p1
false
p2→TotalDelayM < PP→TotalDelayM – HystV
false
return p2
true

Fig. 3. Selection of best parent procedure

(TotalDelayM) for each tested parent is calculated. If neither of parents is the preferred parent, this procedure returns the parent with lowest TotalDelayM. If one of the parents is the preferred parent, a latter comparison is performed using a hysteresis value (HystV) returned by the hysteresis function presented in Algorithm 1. The HystV depends on the nodes EED Rough Estimation (EED_RE), towards the DAG root. In order to obtain this rough estimation a new metric, the Hop Count Metric (HopM) was added to those already defined. Thus, the EED_RE is obtained using HopM multiplied by a constant value K that is assumed to be the worst transmission delay value per hop experienced by a sender node. The K value is obtained using a constant value of 125 ms, which is the default receiver wake-up interval defined in ContikiMAC [14] and multiplied by 2 to include MAC queue delay. The graph of the hysteresis function is presented in Fig. 4. If the PP$\rightarrow$TotalDelayM is less than EED_RE, a negative slope line is used. Otherwise, the minimum hysteresis (MH) is assumed to be 50 ms. The lower the PP$\rightarrow$TotalDelayM value is, the higher the HystV will be, making the parent change less probable. Since a parent change will reset the DIO message timer, this algorithm controls the rate of DIOs in the network and avoids parent selection instability. In order to provide

Algorithm 1: Hysteresis function

```
hysteresis(TotalDelayM){
  K = 250;
  EED_RE = K$\times$ HopM;
  if(TotalDelayM $<$ EED_RE){
    HystV= $\frac{(\frac{EED\_RE}{2})-MH}{0-EED\_RE} \times TotalDelayM + \frac{EED\_RE}{2}$;
  }else{HystV= 50;}
return HystV;}
```

more accurate metrics to the remaining nodes, with delay information that starts from the DAG root, minor changes were also applied to the update metric procedure, which updates the metrics that are used in the DIOs. As a result, each node will only advertise its metrics (ProcDelayM, PathDelayM and

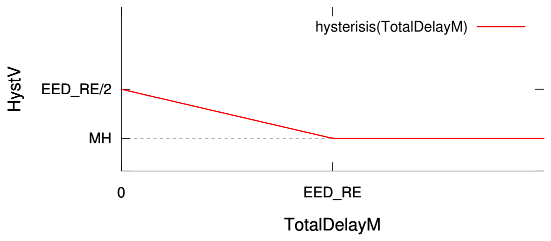

Fig. 4. Hysteresis function graph

HopM) if these are already available from its preferred parent. Whenever a node receives the metrics from a parent in a metric container, it should assume that all metric values account the entire path to the DAG root.

V. Simulation Environment and Setup

RA-EEDEM was evaluated in a grid topology, shown in Fig. 5 which is a simplification of the scenario used by the SELF-PVP project, within an area of 100 m^2. Two types of nodes were considered: the generator/forwarder node (a node that generates and forwards packets) and the sink node (DAG root).

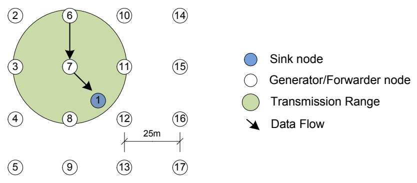

Fig. 5. Grid Topology

The Cooja simulator [15] was used. Each node was simulated as a Tmote Sky [16]. IEEE 802.15.4 MAC and PHY layer specifications were applied using a transmission range of 30 m and the Unit Disk Graph Medium as physical channel model. The nodes ran the Contiki OS 2.5 and were programmed to enable both the debug of application and RPL messages. The application layer used UDP and it generates packets of 100 Bytes in a constant bit rate implemented with a constant Inter-packet Generation Interval (IGI). Simulations were configured to stop whenever the sink received 200 packets from each node. Simulations were repeated 10 times using random seeds. The EED estimations obtained with RA-EEDEM were tested against EEDEM solution [1] and an ETT-based solution where RPL was configured to use the ETX metric. The latter assumes that the EED is estimated using Eq. 4, where ETX is the expected number of transmission attempts required for successfully transmitting a packet, S is the packet size, and D is the data rate of the link.

$$ETT = ETX \times \frac{S}{D} \tag{4}$$

The simulator was configured to output the instant of time when a packet is generated and when the packet reaches the destination application. In order to characterize the accuracy of the EED estimation, when a packet is generated an EED estimation is performed, saved, and later compared with the real EED. The absolute value of the difference between the estimated EED (estEED) and the real EED (realEED) is normalized to realEED, as presented in Eq. 5. The result is given by EED Relative Estimation Error (RE_Error).

$$RE\_Error(\%) = \frac{|estEED - realEED|}{realEED} \tag{5}$$

The RE_Error of all data packets was accounted during each simulation. When the simulation ends, the average and the standard deviation of those errors were obtained.

VI. Results and Analysis

Fig. 6 shows the average and the standard deviation of the RE_Error for the three solutions (see previous Section) using different IGIs. The results show that the Average RE_Error tends to be higher for shorter IGIs. For IGIs larger than 2 seconds, both RA-EEDEM and EEDEM solutions present an Average RE_Error lower than that obtained with the ETT-based solution. For an IGI equal or larger than 3 seconds, RA-EDEM and EEDEM present estimation errors (values ranging from 50% to 60%) lower than the estimation error from the ETT-based solution (values ranging from 85% to 90%). RA-EEDEM presents errors which are 35 percentage points (pp) below the estimations obtained by the ETT-based solution, and 5pp below the values obtained by the EEDEM.

Fig. 7 shows the average number of RPL packets generated per node and per simulation. The results show that for IGIs of 1 second all solutions generate almost the same number of RPL packets. For IGIs larger than 1 second, the ETT-based solution uses a lower number of RPL packets, when compared with the other two solutions. When comparing RA-EEDEM against EEDEM, RA-EEDEM presents a lower number of RPL messages in all circumstances. Combining the results from Fig. 6 and Fig. 7, we can conclude that RA-EEDEM provides a more accurate EED estimation while reducing the overhead of the routing protocol. Fig. 8 shows the average EED for

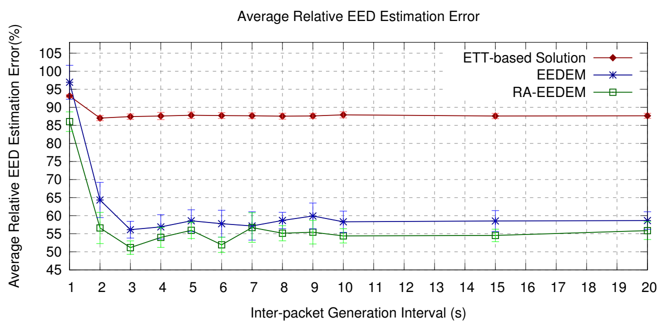

Fig. 6. Average EED Relative Estimation Error

all solutions. The results show that RA-EEDEM and EEDEM present a higher EED, on average, when compared with the ETT-based solution. This is due to the variation of metrics used in RA-EEDEM and EEDEM which impose higher rate of DIO messages. Considering IGI values between 1 and 3 seconds, the RA-EEDEM solution presents a lower average EED, when compared to EEDEM. For IGIs larger than 3 seconds, both RA-EEDEM and EEDEM present approximately the same results, differing from the ETT-based solution by roughly 200 ms. With these results we can conclude that the RA-EEDEM estimation present better results in terms of average EED in high network loads than those obtained using EEDEM. Fig. 9

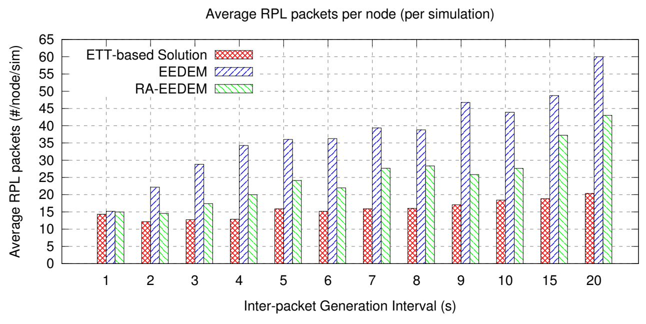

Fig. 7. Average RPL packets per node (per simulation)

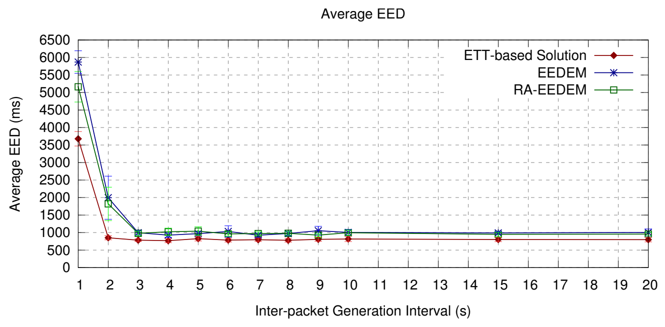

Fig. 8. Average EED

shows the PRR. For IGIs shorter than 7 seconds, the PRR obtained by RA-EEDEM is higher than the PRR obtained using EEDEM, and closer to the results obtained using ETT-based solution. For IGIs higher than 9 seconds, PRR of RA-EEDEM and EEDEM is higher than that obtained using the ETT-based solution. Combining the results shown in Fig. 7, Fig. 8 and Fig. 9, we can conclude that the average EED and PRR will benefit from the reduction of the RPL overhead in high network loads.

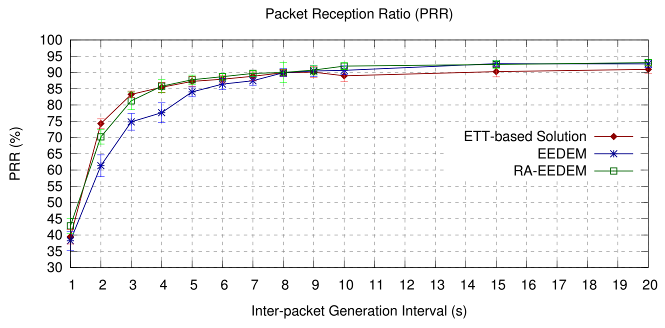

Fig. 9. Packet Reception Ratio

VII. Conclusions

In this paper we propose RA-EEDEM which consists of a set of modifications to RPL aimed to improve the EED estimation accuracy. RA-EEDEM comprises a new metric and modifications to the OF procedures.

RA-EEDEM estimation results were compared against our previous estimation solution EEDEM and a ETT-based solution. The results show that RA-EEDEM improves the accuracy of the EED estimation when compared to the other solutions. When compared to previous solution, RA-EEDEM improves EED estimation by about 5pp (8%) and, for high network loads, presents better PRR by about 10pp (14%), and less Average EED by about 200 ms (12%).

RA-EEDEM can provide a node with a more accurate estimation of delay before it transmits each packet. Further, dropping useless packets will reduce network resource usage and improve the overall performance of the network, while saving energy.

Acknowledgment

This work is partially financed by the ERDF European Regional Development Fund through the COMPETE Programme and by National Funds through the FCT Fundação para a Ciência e a Tecnologia within project SELF-PVP (FCOMP-01-0124-FEDER-013070)

References

[1] P. Pinto, A. Pinto, and M. Ricardo, "End-to-end delay estimation using RPL metrics in WSN," in *Wireless Days (WD), 2013 IFIP*, Nov. 2013, pp. 1–6.

[2] "SELF-PVP: self-organizing power management for photo-voltaic power plants." [Online]. Available: http://www.cmuportugal.org/tiercontent.aspx?id=3374

[3] M. S. R. Baumann, S. Heimlicher, A. Weibel, and S. H. R. Baumann, "A survey on routing metrics," *TIK Report 262, ETH-Zentrum*, 2007.

[4] E. T. Winter, E. P.Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, R. Struik, E. JP Vasseur, and R. Alexander, "RPL: IPv6 routing protocol for low-power and lossy networks," RFC6550, Mar. 2012.

[5] J. Tripathi, J. De Oliveira, and J. P. Vasseur, "A performance evaluation study of RPL: routing protocol for low power and lossy networks," in *2010 44th Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2010, pp. 1–6.

[6] K. Heurtefeux, H. Menouar, and N. AbuAli, "Experimental evaluation of a routing protocol for WSNs: RPL robustness under study," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2013, pp. 491–498.

[7] J. Tripathi and J. De Oliveira, "On adaptive timers for improved RPL operation in low-power and lossy sensor networks," in *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, Jan. 2013, pp. 1–10.

[8] E. JP Vasseur, E. M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing metrics used for path calculation in low-power and lossy networks," RFC6551, Mar. 2012.

[9] P. Thubert, "Objective function zero for the routing protocol for low-power and lossy networks (RPL)," RFC6552, Mar. 2012.

[10] O. Gnawali and P. Levis, "The minimum rank with hysteresis objective function," RFC6719, Sep. 2012.

[11] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-power wireless IPv6 routing with ContikiRPL," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN '10. New York, NY, USA: ACM, 2010, p. 406407.

[12] "Contiki OS." [Online]. Available: http://www.contiki-os.org

[13] M. Durvy, J. Abeill, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels, "Making sensor networks IPv6 ready," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, p. 421422.

[14] A. Dunkels, "The ContikiMAC radio duty cycling protocol," Dec. 2011.

[15] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proceedings 2006 31st IEEE Conference on Local Computer Networks*, 2006, pp. 641–648.

[16] "Tmote sky project." [Online]. Available: http://www.snm.ethz.ch/Projects/TmoteSky