Energy Consumption and Memory Footprint Evaluation of RPL and CTP in TinyOS

Henrique S. Ogawa, Bruno T. de Oliveira, Tiago de J. Rodrigues, Bruno Albertini and Cíntia B. Margi

***Abstract*— One important issue addressed in Wireless Sensor Networks research and standardization is routing among nodes. While RPL, an IETF RFC, and CTP are two well-known examples, they have distinct behavior. In order to select a routing protocol, one need to understand the overhead introduced by each routing protocol. We present an use case considering energy consumption measurements of RPL and CTP, implemented on a testbed based on TinyOS. We also compare energy consumption estimation obtained from simulations running on COOJA with real measurements from our testbed. Lastly, we present metrics for several scenarios running both RPL and CTP.**

***Keywords*— Wireless sensor networks, routing protocol, simulations, overhead.**

I. Introduction

Wireless Sensor Networks (WSN) have been used to support several different applications, mainly related to monitoring, detection and tracking. Nodes in a WSN are typically battery-powered and resource constrained (i.e. limited amount of memory, processing and communication), and communicate through a multihop ad hoc network [2]. There is not a unique definition for the Internet of Things (IoT), but most of them agree that it is composed of devices capable of sensing/actuation, communication and processing [11]. Thus, WSN becomes a key technology for IoT.

WSN communication pattern was initially depicted considering sensors replying to sink queries, and thus Direct-Diffusion and SPIN fit properly. Other communication patterns would consider node to sink, and node to node communication. Given the node to sink communication pattern, the following trend in WSN deployments was tree-based routing, where the sink becomes the tree root and receives all sensed data from the nodes. Thus CTP (Collection Tree Protocol) [3] became a *de facto* standard for TinyOS [5] applications.

Following the effort on standardizing and deploying IP version 6 and the IoT discussion, IETF (The Internet Engineering Task Force) created working groups to address the use of IPv6 over IEEE 802.15.4 networks and related routing issues. RFC 6550 - *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks* describes a routing protocol that supports traffic flows including point-to-point, point-to-multipoint (i.e. from a sink to nodes), and multipoint-to-point (from nodes towards a sink). Therefore, RPL could support different types of applications with several communication patterns running in a given WSN (or low-power lossy network - LLN).

Henrique Seiti Ogawa, Bruno Trevizan de Oliveira, Tiago de Jesus Rodrigues, Bruno de Carvalho Albertini and Cíntia Borges Margi. Universidade de São Paulo, São Paulo-SP, Brazil, E-mails: {henrique.ogawa, brunotrevizan, tiago.jesus.rodrigues, balbertini, cintia}@usp.br.

According to Ko et al. [6], RPL implementation for TinyOS provides an efficient routing performance for the multipoint-to-point traffic pattern that is comparable with CTP in terms of protocol overhead and packet reception ratio. RPL implementation for TinyOS includes a 15-byte header while CTP includes an 8-byte header [6].

Herberg and Clausen [4] present a comparative study of LOAD and RPL with bidirectional traffic using ns2. Authors claim that RPL and LOAD provide similar data delivery ratios, but RPL incurs in more overhead while being more constrained in the types of topologies supported. The main drawback in this study is the use of IEEE 802.11 link layer, which provides higher bandwidth than LLNs.

The main question that arises is how much overhead a protocol that supports different communication patterns will introduce. Another way to look at the problem is to answer if following the IETF standard will be expensive in terms of network and node resources.

This work presents a comparison between two main routing protocols used on WSN. First we present energy consumption measurements of RPL and CTP implemented on a testbed based on TelosB and TinyOS, which are the first results of this kind to the best of our knowledge. Second we compare energy consumption estimates obtained from time spent in each radio state through simulations running on COOJA [8] with real measurements from our testbed. Third, we scaled our testbed size using COOJA in order to estimate CTP's and RPL's efficiency over large and dense networks, using as parameters the data packets' latency times and loss rates, the rate between control packets' number and the data packets' number over the time, and energy consumption estimates. Furthermore, we provide memory footprint of CTP and RPL over our application.

The IEEE 802.15.4 standard includes provision for radio duty cycling in its specification. TinyOS has a Low Power Listening (LPL) mechanism implemented, which uses a wake up routine according to a static and predefined duty cycle period, keeping the radio off as much as possible. In order to compare the energy efficiency in the motes, we estimated it with and without LPL enabled for large networks.

This paper is organized as follows. Section II presents an overview of CTP and RPL, while Section III presents our experimental methodology. Results are presented and discussed in Section IV and we conclude the paper in Section V.

II. CTP and RPL overview

Collection Tree Protocol (CTP) [3] determines a tree from leaf nodes to a (or multiple) root nodes. It starts with sink nodes (i.e. the ones configured to receive data) advertising themselves as logical tree roots. Then nearby nodes replies, assembling a set of routing trees to the sink. Each sensor node replies only to the nearest root. Route constitution to root nodes uses Expected Transmissions (ETX) [3] as routing gradient, prioritizing routes with the lowest ETX values. Link estimation in CTP design is used to evaluate the communication link between the neighbors [3]. CTP has the ability to resolve routing inconsistencies by broadcasting beacon frames periodically [3].

IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [12] is the IETF standard protocol for IPv6 routing over multihop wireless sensor networks. It is a protocol based on distance vector, using routing metrics to assemble a Destination-Oriented Acyclic Graph (DODAG) rooted at the border router. RPL is tree-oriented in the sense that one or more root nodes in a network may constitute a topology that trickles downward to the sensor nodes.

III. Experimental Methodology

The first experimental procedure involved the development and implementation of an application using the available CTP and RPL libraries for TinyOS [5] version 2.1.2. These libraries were used to implement two versions of the same application, differing only by the underlying routing protocol (CTP and RPL). The purpose of the application is to send two predefined bytes periodically (2s) from a source sensor node to the sink node (root) over a multihop network. All nodes operate in TinyOS Low Power Listening (LPL) mode.

The experimental multihop network (i.e. our testbed) consists of a three-node topology in which there is a source, a router, and a sink node; where source should not communicate directly with the sink node. Given the lack of accuracy to determine the radio range of the TelosB [7] motes and its susceptibility to environmental noise, the three-node multihop line-up was chosen to simplify the problem of network topology definition with a real testbed. In order to control the topology, the output power of CC2420 radio chip [1] available on the TelosB was reduced to minimal (-25 dBm).

To perform the simulations mimicking our testbed with TelosB devices running CTP and RPL on TinyOS, we adopted COOJA [8]. COOJA is a cross-level sensor network simulator that adopt a hybrid approach, being able to simulate the network level through a Java environment implementation and the operating system level based on native sensor node program, integrating them by using Java Native Interface. It allows the use of the platform compiled code by emulating TI MSP430 processor present on TelosB mote.

Roussel et al. [9] argue that studies that rely on COOJA emulation to evaluate time-related performance on WSN have a potential risk of suffering from inexact results due to the timing inaccuracy related to delays during packet loading into RF TX buffer in comparison to their experiments made on physical motes. For the TelosB platform they present results to RIOT and Contiki OSes, in which the most significant impacts of the packet loading in the transmission timing are 57% and 13% respectively, and observed delays differences are 15% and 11% respectively. Performing a simple multiplication we can conclude that the impact would be at most 8.6% for RIOT and 1.4% for Contiki. However, even if the presented results are correct, we consider that COOJA remains a relevant tool to conduct this type of study, especially when the objective is to compare protocols, since the delay would affect both protocols in the same way. In addition, numerous hardware and environmental variables (e.g. interference from different sources and device temperature) may also affect studies that adopt physical platforms, providing inaccurate results much more difficult to reproduce and verify in comparison with experiments using a simulator.

In order to select the most appropriate duty cycle we tested four arbitrary TinyOS LPL sleep intervals in COOJA, aiming to determine the less costly from the power consumption perspective (i.e. smallest radio activity). Table I presents the average time fraction of radio activity of the TelosB motes running the TinyOS CTP and RPL application (*CtpTestApp* and *RplTestApp*, respectively) in a 10-minute COOJA simulation. Given these data we selected 128 ms sleep interval.

TABLE I
Fraction of radio activity for different duty cycles

Sleep Interval (ms)	*CtpTestApp*	*RplTestApp*
64	11,05%	13,55%
128	**8,26%**	**12,33%**
256	9,01%	13,79%
512	12,06%	18,19%

The memory footprint metrics considered for applications and protocols are occupation of code memory and utilization of volatile memory (RAM), obtained through the tools MSP430-size and MSP430-ram usage, respectively. These tools are part of compiler toolchain used to generate programs for the TelosB sensor, the MSP430GCC [10].

The experimental procedure for the large scale networks tests was also based on simulations performed on COOJA. For these tests, we used the same CTP's and RPL's applications mentioned before, changing the data packets' contents from the two predefined bytes to a four-byte message, containing the sender device's node ID and a counter. The data packets' frequency were also changed from one at each 2s to one at each 5s. All nodes, but the sink, were configured to send data packets. The network parameters are shown in Table II.

TABLE II
Network Parameters

Parameter	**Value**
Network topology	Square grid
Grid distance parameter	20m
Grid size	25 / 49 / 64 / 81 motes
Number of sinks	1
Sink position	upper-left corner
Radio communication radius	30m
LPL sleep interval	128 ms and none
CTP trickle timer	128 ms to 512000 ms
RPL DIO trickle timer	256 ms to 262144 ms
network simulation duration	5 minutes

A. Testbed Energy Consumption Measurements

In order to obtain an exact measurement of energy consumption and execution time, we used a measurement setup in which direct measures were carried out on the TelosB mote, executing the TinyOS [5] *CtpTestApp* and *RplTestApp* applications. The digital multimeter Agilent 34401A was used to monitor the current flow at a sampling rate of 500 Hz, while a precision power supply, Agilent E3631A, was configured to supply 3V to the TelosB sensor. A General Purpose Interface Bus (IEEE 488 standard) cable was used to connect the multimeter to a computer running the software LabView, which collects and records the measurement samples. Instantaneous current gathered through the measurement setup was integrated by a Matlab script using Simpsons method, and multiplied by the constant voltage to obtain the energy consumption.

Our measurement setup has a limit of 50K current samples, providing a 100 seconds measurement window, starting simultaneously with node's boot process. We repeated the measurement cycle 10 times for each node for each application, summing up 60 data samples. Applications ran over TinyOS with a 128 ms duty cycle, the less costly as in Table I.

B. Simulation-based Energy Consumption Calculation

Although COOJA cannot provide energy consumption values such as the multimeter, it supplies the total time the radio is turned on ($D_{radioOn}$), as well as the fraction of time the radio is transmitting (D_{TX}) and receiving (D_{RX}). The estimated energy consumption can be obtained from Equation 1. Idle listening is given by $D_{idle} = D_{radioOn} - D_{TX} - D_{RX}$. We used 21.8 mA as constant current draw when receiving (I_{RX}) and 10.3 mA when transmitting (I_{TX}), according to TelosB and CC2420 radio datasheets [7], [1].

Then we executed the *CtpTestApp* and *RplTestApp* in the COOJA simulator during 100 seconds. We obtained the estimated energy consumption given the time fractions provided by COOJA and using Equation 1, where: E = energy consumption (J), V = system voltage (V), T = time of simulation (s), D_{idle} = time percentage of idle listening, D_{RX} = time percentage of RX, D_{TX} = time percentage of TX, I_{RX} = current drained in RX (A), I_{TX} = current drained in TX (A).

$$E = V \times T \times ((D_{idle} + D_{RX}) \times I_{RX} + D_{TX} \times I_{TX}), \quad (1)$$

C. Packets Related Rates Measurements

To obtain the data and control packets information, we used Mote Output and the Radio Messages tools from COOJA. The first logs all the motes' outputs, time-stamped and with source identified. Second logs all radio messages time-stamped, message size (bytes), source and destination IDs. Application sends the mote's ID and a counter on the payload (including the sink), so we could match both logs and calculate message latency and loss rate. Data packet payload size was selected to be easy recognizable, so we could differentiate data and control packets.

IV. Results and Analysis

Following the methodology described previously in Section III-A, we obtained the energy consumption for 100 seconds of operation of RPL and CTP applications. Figure 1 depicts the energy consumption, showing the average results for each node with its respective error bars and a confidence interval of 95%. Sink and source nodes consumes more energy when using RPL in comparison to CTP, but difference is not significant given the confidence interval. Median difference for sink nodes are around 3.8%, while 7.8% for source nodes. However, routing nodes spent about 46% more energy when using RPL to perform the same task as CTP. This variation is related to DODAG maintenance advertisement overhead (DAO and DIO), plus the amount of protocols required to support RPL (6LoWPAN, ICMPv6, and UDP), as noticed by Ko et al [6].

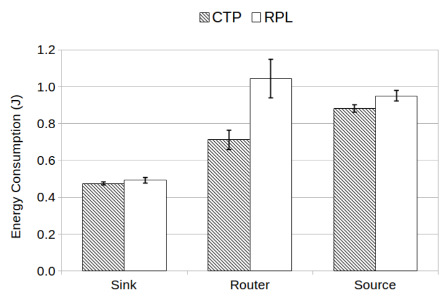

Fig. 1. Comparison between CTP and RPL energy consumption for 100 seconds of operation.

Using the methodology described in Section III-B, we estimated energy consumption. The obtained results are shown in Figure 2, in which we compare the COOJA-simulation obtained results with the experimental ones.

Comparing energy consumption between COOJA simulation and testbed, we observed that for CTP results differ by 4% for the sink node, 5% for source and 19% for router. Similarly, RPL presented a 12% difference for sink, 9% for source, and 3.4% from router. Overall, the average difference is 8.7%. Despite the difference, there is a clear trend when comparing the results. We think that the difference between simulation and experimental results is caused by electromagnetic interference over the non-shielded measurement setup, as well as the deterministic behavior from COOJA.

Table III shows the RAM and ROM memory usage of each TinyOS compiled application, including an Active Message and a BLIP [6] compiled application for comparison purposes. We compiled both BLIP and RPL applications with and without header compression to estimate the memory footprint fraction of BLIP, RPL, and header compression itself in the memory usage of *RplTestApp*.

Considering TelosB (10KB of RAM and 48KB of ROM), an application using only Active Message (without routing protocol), allocates 3.6% of available RAM and 23% of ROM. Adding CTP routing to a similar application needs 12% more RAM and 13.8% of ROM, while using RPL requires 46% more RAM and 26.3% more ROM when compared to CTP,

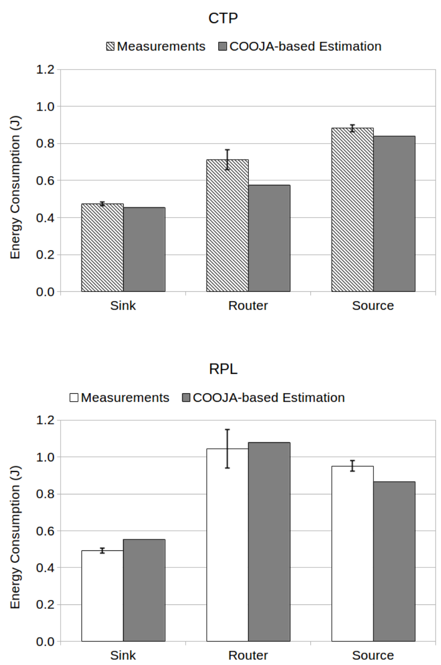

Fig. 2. Comparison between COOJA-based estimation calculus and energy consumption measurements for CTP and RPL in 100 seconds of operation.

TABLE III
MEMORY USAGE

	RAM (bytes)	ROM (bytes)
Active Message	366	11330
CtpTestApp	1602	18116
BLIP (no header compression)	4794	23096
BLIP (header compression)	4794	25392
RplTestApp (no header compression)	6322	31028
RplTestApp (header compression)	6322	33324

given that RPL keeps 20 downwards routes on the fly and 20 parent candidates, and CTP stores only 10 parent candidates. The most substantial portion of *RplTestApp* memory footprint comes from BLIP, since RPL requires a 6LoWPAN implementation. Header compression adds 4.7% of ROM without affecting RAM usage, no matter if using BLIP or RPL.

Following the methodology described in Section III-C, we obtained results concerning the latency average times and the data packets loss rates for both protocols with and without the TinyOS LPL enabled. Results are shown in Figures 3 and 4.

From Figure 3 the minimum latency is approximately constant for both protocols, in both modes of operations and for all networks size, since it is related to the time necessary for one of the sink's neighbor to send it a packet. Notice that the average latency and loss rates (Figure 4) are larger when LPL is enabled, since packets wait longer in the queue while the radio is turned off and some are discarded once the queue is full. For both modes of operation, loss rates is two orders of magnitude greater for RPL than CTP, since RPL has a larger number of control packets (around one order of magnitude) as depicted in Figure 5, filling the limited queues. For the same reason, CTP is more efficient in the average latency with the LPL disabled. But when it is enabled, RPL becomes more

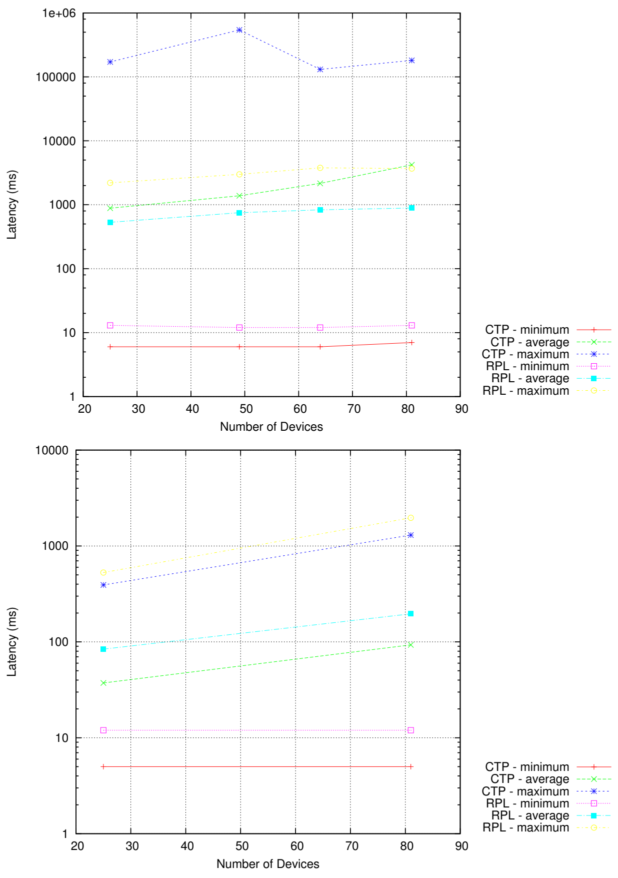

Fig. 3. Data packet latency for CTP and RPL with (above) and without (below) enabled Low Power Listening for different network sizes.

efficient due to its larger loss rates, and the packets that arrive to the sink are majority of the sink's closest neighbors.

Figure 5 shows that the rate of data packets by control packets number increases over the time, because the first one happen with a fixed period of 5 seconds for all nodes, and the second decreases over the time, as the network stabilizes. It is also remarkable that RPL has much more control packets than CTP, in order to maintain the DODAG and the protocols necessary to support RPL. Disabling LPL also reduces the amount of control packets for both protocols, since less packets are lost and the network convergence is faster.

The motes energy consumption increase when LPL is enabled as depicted in Figure 6, due to radio duty-cycle which needs more time turned on in order to flush the packet queue. Small networks benefit from CTP given its lower energy consumption in comparison to RPL, while the opposite occurs for larger networks. Thus, CTP nodes consume less energy than RPL up to the point in which the Cumulative Distribution Function (cdf) is approximately 0.5, and becomes more costly from this point on. While using LPL decreases the overall energy consumption over time, it degrades the network metrics for both protocols.

V. Final Considerations

The main question addressed was how much overhead (in terms of energy consumption and memory) RPL, the IETF

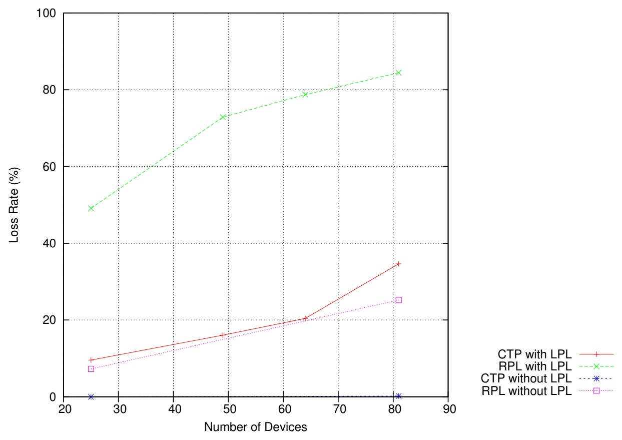

Fig. 4. Data packet loss rates for CTP and RPL with and without enabled Low Power Listening for different network sizes.

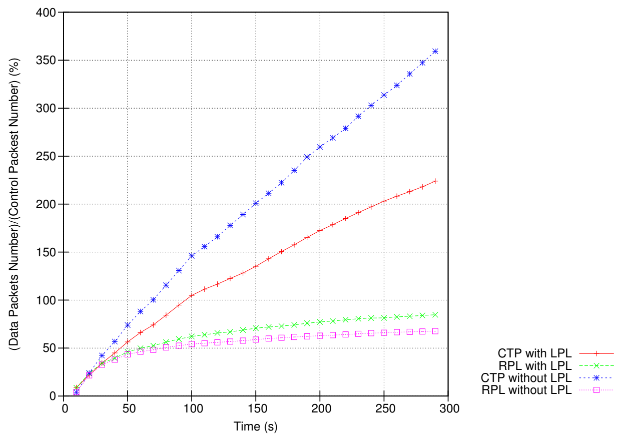

Fig. 5. Relation of data and control packets over simulation time for CTP and RPL with and without enabled LPL enable for 25-node grid network.

standard, introduces compared to CTP. We presented energy consumption measurements of RPL and CTP implemented on a testbed based on TelosB and TinyOS, which are the first results of this kind to the best of our knowledge. For our setup, RPL requires a large amount of memory on TelosB and consumes about 20% more energy than CTP, while it provides more flexibility in terms of communication patterns.

We also estimated the energy consumption using the time spent in each radio state obtained from simulations running on COOJA [8], and compared the results with real measurements from our testbed. Results indicate that the use of COOJA provide a reasonable accurate estimate for energy consumption, what enables researchers to scale the testbed sizes.

Next we evaluated both protocols through simulations for larger networks. We noticed that CTP is more reliable than RPL, since its loss rate was smaller. While TinyOS Low Power Listening (LPL) mode is a good option for energy-constrained networks, it increases latency times and packets loss rates for both RPL and CTP.

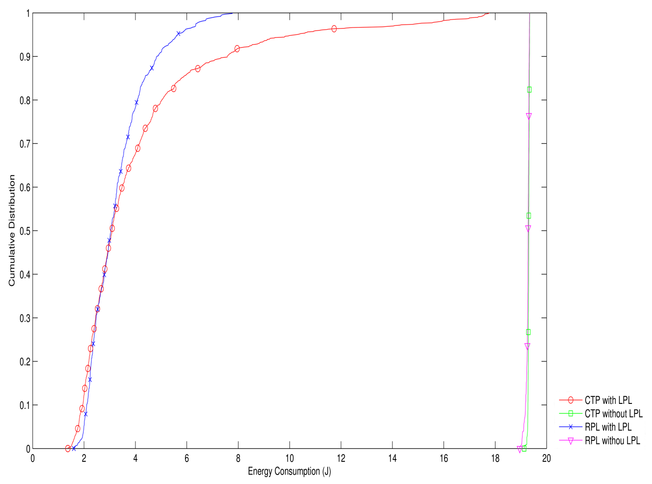

Fig. 6. Energy consumption cumulative distribution function estimates for CTP and RPL with and without enabled LPL, for 25, 49, 64 and 81 nodes networks.

Acknowledgments

This work is funded by FAPESP grants #2013/15417-4 and #2014/06479-9 and by FDTE grant #1450. C. B. Margi is supported by CNPq research fellowship #307304/2015-9.

References

[1] Chipcon. CC2420 Datasheet. `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`, 2007.

[2] D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. *Computer Magazine*, 37(8):41–49, 2004.

[3] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM.

[4] U. Herberg and T. Clausen. A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-directional Traffic in Low-power and Lossy Networks (LLN). In *Proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *SIGPLAN Notices*, 35(11):93–104, 2000.

[6] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis. In *Proceedings of the Workshop on Extending the Internet to Low power and Lossy Networks - IPSN'11*, Apr.

[7] MEMSIC. Telosb datasheet. `http://www.memsic.com/userfiles/files/DataSheets/WSN/telosb_datasheet.pdf`, 2011.

[8] 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*, pages 641–648, Nov 2006.

[9] K. Roussel, Y.-Q. Song, and O. Zendra. Using Cooja for WSN Simulations: Some New Uses and Limits. In K. Roemer, editor, *EWSN 2016 - NextMote workshop*, EWSN 2016 - NextMote workshop, pages 319–324, Graz, Austria, Feb. 2016. ACM, Junction Publishing.

[10] S. Underwood. MSPGCC. `http://mspgcc.sourceforge.net/`, 2009.

[11] R. Want, B. N. Schilit, and S. Jenson. Enabling the internet of things. *Computer*, 48(1):28–35, Jan 2015.

[12] 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.