REACTIVE: a Peaceful Coexistence between Deluge and Low Power Listening

Andrea di Cagno, Mario Paoli, Ugo Maria Colesanti, Andrea Vitaletti
Department of Computer, Control, and Management Engineering
University of Rome "La Sapienza"
Email: {dicagno,paoli,colesanti,vitaletti}@dis.uniroma1.it

***Abstract*—In this paper we analyze the interaction between two communication protocols implemented in TinyOS 2.x: Deluge T2, an over-the-air programming protocol, and the Low Power Listening (LPL) implementation provided with the standard Medium Access Control layer, BoX-MAC. We show how the characteristics of the two layers deeply diverge, leading to a sensible performance degradation. We present REACTIVE, a simple algorithm that, integrated in Deluge, is able to dynamically disable and re-enable LPL in order to boost Deluge performance, leveraging on the critical aspects of the layers integration. REACTIVE is able to increase the performance of Deluge, compared to its standard implementation, by a factor of 2.6 in terms of energy efficiency and 7 in terms of dissemination time.**

I. Introduction

Over the Air Programming (OAP) for Wireless Sensor Networks (WSNs) is an essential feature for long-term applications, where physical access to deployed wireless motes cannot be guaranteed or when time and costs of manual reprogramming exceed the benefits. Typically, OAP protocols for WSNs are based on broadcast dissemination of a new image or part of it [1], [2], [3], [4]. Because of its inherent broadcast nature and of the large amount of data disseminated over the network, OAP is an energy consuming task. Commonly used metrics for evaluating the energy efficiency of an OAP protocol are the disseminated image size and the dissemination time: the smaller will the image be, the less network traffic it will require, moreover, the less time it takes for an image to disseminate, the earlier the nodes can activate energy saving features.

Another important aspect that rarely is taken into account is the efficiency of the integration between the OAP protocol and the underlying low-power Medium Access Control (MAC) layer, that is mandatory in long-running network applications. Anyway, it is straightforward to notice the contrasts there exist between requirements of a low-power MAC protocol and an OAP protocol. Indeed, to save power, a low-power MAC protocol increases the overall network latency which degrades the dissemination time of an OAP protocol. On the contrary, the high number of broadcast packets of an OAP protocol sensibly increases the duty-cycle of the MAC layer. In this paper we deeply investigate this relevant aspect and show how, through the knowledge of the underlying MAC protocol, the efficiency of the OAP protocol can be significantly increased.

We used the official 2.1.2 release of the TinyOS operating system [5] that has been ported on our custom hardware platform dedicated to WSNs, the MagoNode [6]. TinyOS comes with an official OAP protocol named Deluge T2 and a low-power MAC protocol featuring Low Power Listening named BoX-MAC [7]. We made a preliminary experiment comparing Deluge with and without LPL showing the parameters we could leverage on to increase its efficiency. We then proposed a simple reactive algorithm integrated in Deluge able to dynamically disable part of the LPL features to boost Deluge over-the-air reprogramming. With our simple approach, we are able to increase the performance of Deluge, compared to its standard implementation, by a factor of 2.6 in terms of energy efficiency and 7 in terms of dissemination time.

In the remainder of this paper we will present the state of the art of OAP procols (section II) and we will introduce BoX-MAC and Deluge (section III). Furthermore, in section IV and V we will present the experimental data and analysis of the reactive algorithm we implemented compared to other solutions. Finally, in section VI we will give some conclusions and intentions for the future.

II. State Of Art

TinyOS does not support loadable modules, which means that the whole ROM image needs to be disseminated before reprogramming can occur. On the contrary, other operating systems like Contiki [8] and SOS [9] have the ability to load dynamic modules, which allow those systems to limit propagation to updated modules only, thus, limiting the overall disseminated code size. In the last years similar techniques acting on the code structure in such a way that only the code differences are propagated and then reassembled locally on each mote before reprogramming, have been presented for TinyOS.

A. Zephyr

Zephyr [1] uses an optimized byte-level algorithm based on rsync to compute delta difference between ROM images. The computed delta is then disseminated and the ROM image reconstructed on each mote. The authors observe that computing byte-level delta is not enough, since small changes in the code might shift entire code blocks, thus, generating big sized deltas. To face this issue, the authors substitute function calls with jumps to fixed code locations acting as indirection table. So all the calls to a particular function in a user program will refer to the same entry in the indirection table. In this way if

978-3-901882-63-0/2014 - Copyright is with IFIP

these calls are placed in different locations (i.e., shifted) only the value on the indirection table will be affected, thus keeping the delta between the two versions small.

B. Dynamic TinyOS

Dynamic TinyOS [2] has the ability to compile TinyOS components as separated ELF files that are linked together at run-time by a *TinyManager*. Separating the components in multiple files allows Dynamic TinyOS to disseminate only updated components while keeping the rest of the code unchanged. With Dynamic TinyOS it is also possible to group components whenever no updates are expected on those parts, in order to improve code optimization of the compiler. The result of this operation is a reduction in size of the ROM image.

C. Elon

Elon [3] introduces the concept of *replaceable* components. The principle relies on the fact that in TinyOS only few application-level components are expected to be updated while core modules are typically left unchanged. Hence, labeling few components as replaceable let the user to disseminate updates for those components only, keeping the update small. In addition, while the base version of replaceable components are stored in ROM, the updates are stored in RAM, without the need to use external flash.

D. Stream

Stream [4] stores in the external flash memory an image that implements the OAP protocol. When a network reprogramming is needed, Stream switches to the OAP image in order to disseminate the new application code. This technique allows Stream to keep a small overhead on the application code size, since it implements a very lightweight module whose job is to disseminate the command to switch to the OAP image. This small overhead greatly reduces the overall application code size, thus the dissemination time and network traffic. However, as pointed out in [3], in real-world complex applications, most of the core modules of TinyOS (i.e., radio driver, routing protocols, etc...) that are used by the OAP protocol, are already imported in the application itself, thus limiting the benefits of this approach.

We want to point out that most of the above solutions try to reduce the power consumption required by an OAP protocol by shrinking the size of the disseminated code. Those solutions are complementary to the one presented in this paper. Indeed, in our solution the power consumption is cut by boosting the Deluge dissemination time. Thus, reducing the code size, whenever is possible, further improves the performance of our approach.

III. BoX-MAC and Deluge

BoX-MAC [7] is the default low-power mac protocol of the TinyOS operating system. It is based on the Low Power Listening technique where each node periodically wakes up and checks for incoming packets. Despite the period is fixed for all the nodes, the wake-up schedule is not shared. Hence, the transmitter has to repeatedly send the packet for the whole period length (figure 1). On unicast packets, the receiver notifies the transmitter with an ack message as soon as the packet is received, letting the transmitter to stop sending.

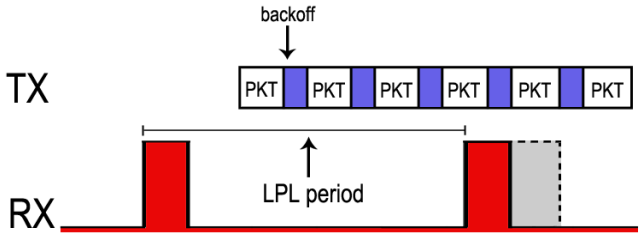

Fig. 1. BoX-MAC transmission of a broadcast packet.

Despite Low Power Listening greatly reduces the radio power consumption, in [10] Langendoen et al. show how under high traffic loads, overhearing and collisions, especially in presence of broadcast packets, significantly degrade the energy efficiency of such approach. This is exactly what we expect during network reprogramming: hundreds of packets, bearing the ROM image, are disseminated in broadcast over the network. Thus, we have to keep into account the above cited issue when using OAP protocols on top of BoX-MAC.

TinyOS provides Deluge T2 [11], a reliable data dissemination protocol for propagating large data objects from one or more sources to all the other nodes of the wireless sensor network (WSN). Deluge can be exploited to support the over-the-air network reprogramming of a WSN, disseminating in a reliable way ROM images over the network. The start and stop of the network reprogramming is managed by the Drip [12] dissemination protocol, which disseminates broadcast messages carrying program version and commands. Drip is based on Trickle [13], a density-aware algorithm that organizes data dissemination in rounds. The round frequency decays exponentially to limit the number of packets retransmissions. However, when a message handling an updated command is detected, the frequency is restored to its initial value. Upon the reception of a Drip message (DRIP) carrying a *start command*, the mote starts transmitting broadcast advertisement messages (ADV) to notify the neighbors of its current version number and the portion of updated ROM it already has. The transmission of ADV messages is also based on the Trickle algorithm, thus, it is organized in rounds. At each round, all the motes check whenever there is a node with an updated version number and with updated portions of ROM. In such case, they perform an unicast requests (REQ) to their neighbor to start downloading data packets (DATA) containing the updated code. DATA messages are transmitted as broadcast packets to allow neighbors overhearing, thus, greatly reducing REQ and DATA messages.

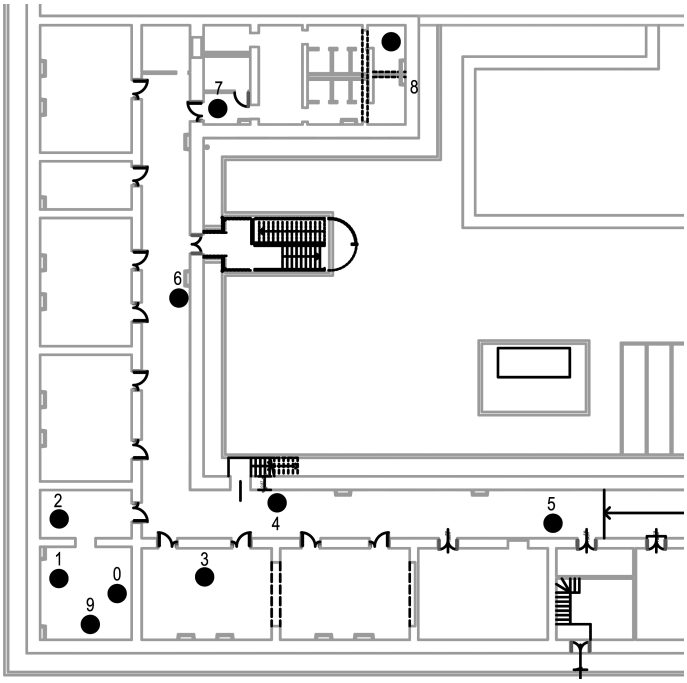

Fig. 2. Testbed map at Department of Computer, Control and Management Engineering.

IV. Preliminary Experiments and Analysis

A. Setup

We analyzed the effects and interactions between the Deluge and BoX-MAC protocols, deploying a wireless sensor network in the basement of the the Department of Computer, Control and Management Engineering of our university (figure 2). The deployed sensor network is made of 10 MagoNodes [6] running TinyOS 2.1.2 and using the RfxLink library that implements BoX-MAC's Low Power Listening. To emulate long-lasting network deployments, we set the duty cycle to 1%, which leads to a LPL period of 500ms against a listen period of 5ms. In addition, the application running on wireless motes is made of the Collection Tree Protocol (CTP) [14] which was programmed to periodically forward 1 dummy packet per minute toward a mote acting as sink. We enabled the Traffic Monitor provided by the RfxLink library to keep trace of the radio power consumption, the number of packets transmitted and received. In addition, we used a modified version of Deluge that notifies the application layer the start/stop events related to the over-the-air reprogramming. When on of those events is triggered, a snapshot with the statistics is taken and transmitted over the network exploiting the CTP protocol. In this way, we are able to take accurate measurements on ROM image dissemination time, power consumption and other statistics about motes.

B. Preliminary Experiments

The preliminary experiments consist in comparing the differences between wireless reprogramming of a fixed-size ROM image on a network running the Deluge protocol with LPL disabled, hence with 100% duty cycle, against the same job performed with LPL with duty cycle set at 1%. We will refer to these two modes as *NO-LPL* the former and *W-LPL* the latter. We use the following metrics to perform the comparison:

- *Dissemination time:* the average time elapsed from the start command, received by means of a drip message, to the completion of the image download
- *Radio duty cycle:* the average percentage of radio activity over the dissemination time
- *AVG TX packets:* the average number of packets transmitted by each mote
- *Packet types weight:* which represents the percentage composition of AVG TX packets in terms of DRIP, ADV, REQ and DATA packets.

We performed the preliminary experiment running 10 times the over-the-air reprogramming of a 28KB ROM, which corresponds to the Blink application available in TinyOS 2.1.2 with Deluge support.

C. Preliminary Results

The results are summarized in table I, figure 3a and figure 3b. The most relevant result is the sharp performance degradation in terms of dissemination time, which increases by a factor of 8 switching from NO-LPL to W-LPL mode, and the co-related increase of the AVG TX Packets by a factor of 100. The enormous increase in network traffic in the W-LPL mode increases the duty cycle up to 36%. This, combined with the longer dissemination time, makes the energy consumption to increase up to 3 times the NO-LPL one. Looking at AVG TX packets composition in figure 3a, we observe how in NO-LPL the main network traffic is made of DATA messages (81.2%) and, in smaller quantity, on ADV messages (13.6%). The Deluge description of section III easily explains these values: DATA packets represent the predominant part of the network traffic due to the multi-hop dissemination of the 28KB ROM image. Furthermore, we recall that, when a new ROM block has been downloaded, the mote resets the Trickle timer to increase the ADV frequency with respect to its neighbors. This procedure, repeated several times on several nodes, makes the ADV packets to represent a significant portion of the overall traffic.

TABLE I
Preliminaty results

	NO-LPL		W-LPL	
	Value	Std-dev	Value	Std-dev
Time (s)	65	12.7	524	27.6
Duty Cycle (%)	100	0	36	10
Energy (J)	3.29	0.3	9.6	3

Figure 3b shows the sharp increase of the transmission network traffic when enabling W-LPL. The increase is explained by the fact that each packet transmission in W-LPL is repeated hundreds of time, as long as the 500ms LPL period. Despite an increase was expected, what is noticeable is the difference of the packet types weight: while with NO-LPL the DATA packets were representing more than 80% of the overall traffic,

in W-LPL it reduces to 42.2% only, while ADV messages become the predominant part of the network traffic, growing from 13.6% with NO-LPL to 52.9% with W-LPL. To find the reason for this unexpected behavior we have to look at the events which trigger a reset of the Trickle timer for ADV dissemination, which are:

- a new block has been stored
- the whole image has been downloaded
- reception of the Drip start command
- reception of an ADV from a neighbor node that has different number of blocks.

While the behaviors caused by the first three events are almost the same in both W-LPL and NO-LPL mode, the response at the occurrence of the latter is different. Indeed, using LPL increases the probability that the nodes have different block numbers during the dissemination phase. This implies that ADV received from a neighbor will trigger a reset on the Trickle time, increasing the transmission rate of ADV messages.

From this preliminary experiment we can conclude that introducing LPL, drastically degrades the overall OAP performance in terms of dissemination and power efficiency, as expected. On the other side LPL must be enabled when running long-lasting sensor networks.

V. Proposed Algorithms

A. DATA-NO-LPL

An easy and intuitive method, inspired by Stream (section II), to mitigate the effect of LPL on the Deluge performance can be summarized as follows: upon the reception of the start command, switch from the current running application to a dedicated *golden image* already stored in ROM which does not use LPL. The only goal of the golden image is to let a new image version get disseminated over the wireless network. At the end of the dissemination, a node switch to the new image version which uses LPL. The drawback of this approach is that every node in the network must receive the dissemination command almost at the same time. In this way all the nodes are able to participate to the dissemination phase, switching to the golden image and so disabling LPL. Otherwise, nodes are unable to receive packets of the new image version since they are still using an image featuring LPL, instead of the golden image.

Recalling the outcomes of section IV-C, we conceived a new approach, named DATA-NO-LPL, that drastically reduces the network traffic by dynamically disabling LPL from DATA packets. The proposed solution is able to guarantee state correctness, i.e., the motes consistently switch between LPL and non-LPL modes, and, thanks to the sensibly lower network traffic, greatly reduces the image dissemination time.

To describe in details our approach, we first need to introduce the following definitions: we refer to *full-LPL* as a mote running LPL both, in transmission and reception, while we will refer to *tx-LPL* as a mote running LPL in transmission while keeping a fully-on radio in reception. First of all, we observe that a mote running tx-LPL does not compromise the overall mote-to-mote communication since an always-on receiver is still able to receive packets from an LPL transmitter. Symmetrically, a tx-LPL mote would continue to transmit using LPL, which would let the mote to communicate with another mote running full-LPL. As a matter of fact, a hybrid network of motes running full-LPL and tx-LPL would not compromise the overall network functionality.

DATA-NO-LPL takes advantage of this observation and uses the Drip messages, that carry the start/stop commands of Deluge, to trigger a switch between full-LPL and tx-LPL on all the motes upon the reception of a start command. A mote running our algorithm would switch back to full-LPL at the completion of the new image download. This simple algorithm allows the mote to react dynamically to the start of an over-the-air reprogramming without the need to add any kind of control overhead to the protocol. Of course this is not enough to solve the performance degradation observed in the previous experiments, since it would not disable the LPL in transmission, which is the main responsible of the performance degradation. Moreover, keeping the radio always on increases the energy consumption. However, this simple algorithm allows us to perform some optimization.

In particular, DATA-NO-LPL is based on the observation that DATA messages in Deluge are transmitted upon the reception of a REQ message. This means that the mote sending the REQ message has already received the start command from Drip, thus it is running tx-LPL. This implies that the mote is able to receive packets without LPL, so we can safely disable LPL transmission on the DATA packets. Since we have seen that DATA packets represent half of the overall traffic in an OAP protocol running LPL, without LPL we should notice a great reductions in dissemination time. This lead us to repeat the same test performed in the preliminary experiments running Deluge with DATA-NO-LPL. Results are shown in table II and figure 3c. We figure out that, as expected, the dissemination time drops up to 50% when compared with the W-LPL mode and that the DATA messages represent 42% of the overall traffic when transmitted in W-LPL mode, while in NO-LPL mode they represent only the 2%.

Despite the great reduction in both, dissemination time and transmitted DATA packets, the overall energy consumption increases from 9.6 Joules of the W-LPL mode to 14.9 Joules of the DATA-NO-LPL one. This sensible increase is related to the fact that the 50% reduction in dissemination time is compensated by fact that the radio duty cycle is 100% in DATA-NO-LPL, while in W-LPL it is 36%. In order to outperform the energy consumption of the W-LPL mode, our solution should reduce the dissemination time up to 70%.

B. REACTIVE

Having a look back to figure 3c, we notice that most of the remaining network traffic is now related to ADV messages (93.5%) that are still transmitted in LPL. This is the reason why REACTIVE, which extends DATA-NO-LPL, leverage on the ability to dynamically disable LPL on ADV messages.

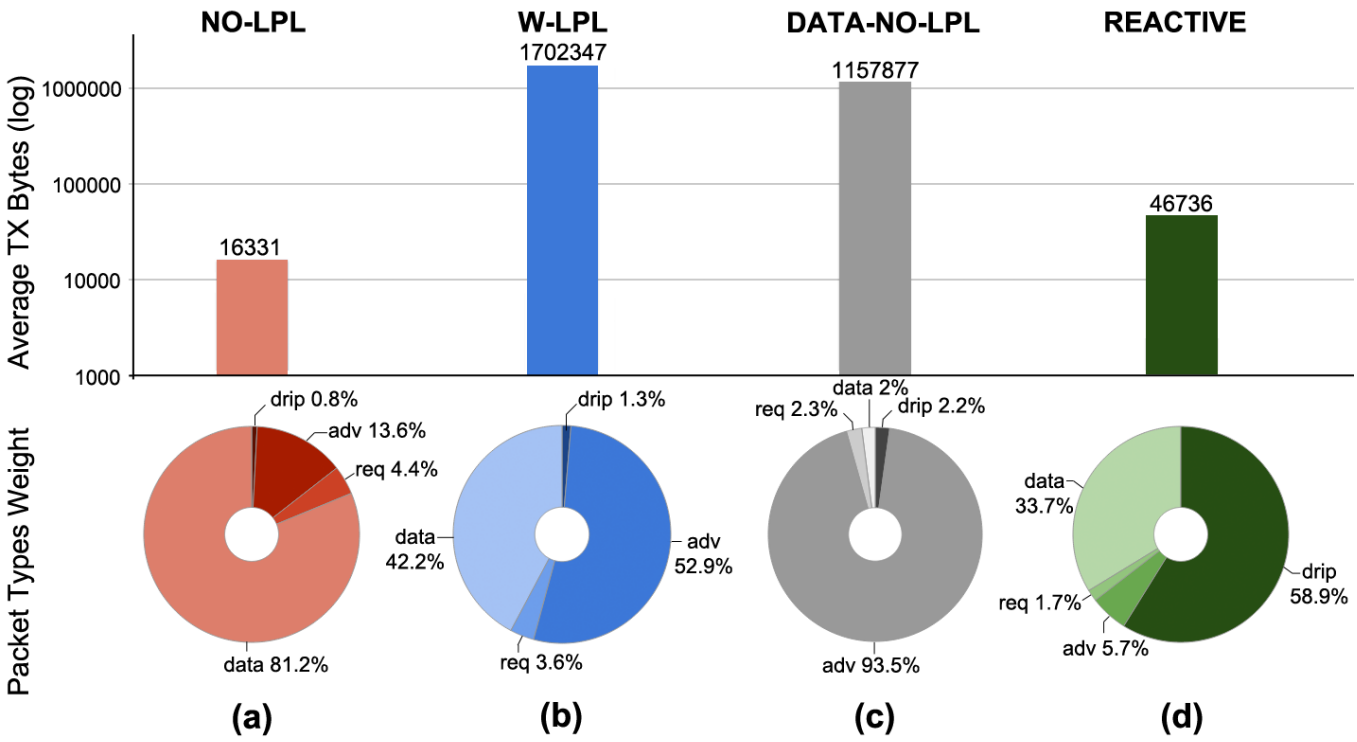

Fig. 3. Network traffic during over-the-air reprogramming

As opposed to DATA-NO-LPL, removing LPL from ADV messages is not straightforward. Indeed, issues arise during start and stop transition phases, where not all the nodes share the same LPL state, in particular:

- when starting over-the-air programming, nodes can be in tx-LPL or full-LPL based on the fact they already received the Drip start message or not;
- when finishing the download of the image, the node switch back to full-LPL but other motes might still be downloading the image, hence, they might still be in tx-LPL

The problems during transition phases is that an ADV message transmitted without LPL could not be overheard by its neighbor. Actually, the start phase transition does not represent an obstacle keeping state correctness in the network since a node that has not yet received the Drip start message, would anyway ignore incoming ADV messages. On the other side, as soon as it receives the start message, it would switch to tx-LPL, thus receiving ADV messages sent without LPL. The real obstacle is represented by the early switch during the stop transition. In fact, a node switching back to full-LPL will no longer provide DATA packets to a node in tx-LPL. This will prevent the him to finish the image download, keeping it in tx-LPL almost forever.

To prevent this issue, REACTIVE uses a timeout timer initialized to a value τ that determines whether the ADV messages are sent in LPL or not. The principle is that if a mote finishes the image download, it is prevented from switching back to full-LPL until the timer fires. However, the timer is reset each time the mote detects an ongoing reprogramming activity. This means that the timer is reset when:

- the mote sent a REQ message,
- the mote sent a DATA message,
- the mote receives a REQ message,
- the mote receives an ADV message from a mote that requires data.

This ensures that the mote does not switch back to full-LPL until reprogramming activity is detected in the surrounding neighborhood. It is important to point out that this simple timeout timer added to our algorithm also guarantees state correctness even in special cases where the timeout timer erroneously elapses. In fact, let assume two nodes that are performing over-the-air reprogramming. The one with the less up-to-date ROM image is requesting data to its neighbor. But if the neighbor has already finished to update the ROM image and if the τ value of the timeout timer is too short, it will switch back to full-LPL. This will leave the former node in tx-LPL without the possibility to communicate with its neighbor. However, the absence of communication will prevent the timeout timer to reset, thus, at some point it will fire on this node too. As a result, the node will start sending ADV messages with LPL enabled that will notify its neighbor and resume the communication.

We tested REACTIVE following the same experimental setup described in section IV-A, setting the τ value equal to 4 seconds. The results are summarized in table II and figure 3d. Table II shows how REACTIVE shrinks the dissemination time up to 7 times when compared to W-LPL and up to 4 times

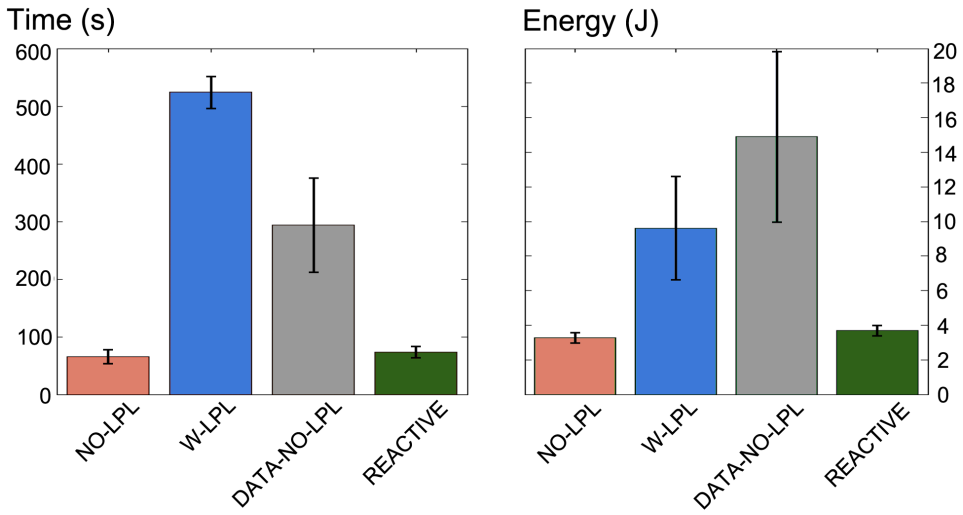

Fig. 4. Performance summary in terms of dissemination time and energy consumption

when compared to DATA-NO-LPL. Motes in REACTIVE have an average energy consumption of 3.7 Joules which is 61.5% less than W-LPL. Indeed REACTIVE dynamically disables most of LPL communication during an over-the-air programming phase. This brings REACTIVE performance close to NO-LPL, but with the advantage of running on a long lasting wireless network based on low power MAC protocols such as BoX-MAC. Figure 4 shows a summary of the results observed in this paper.

TABLE II
ALGORITHMS RESULTS

	DATA-NO-LPL		REACTIVE	
	Value	Std-dev	Value	Std-dev
Time (s)	294	82	73	10
Duty Cycle (%)	100	0	100	0
Energy (J)	14.9	4.9	3.7	0.3

VI. CONCLUSION

This paper presents a reactive approach to the Over the Air Programming (OAP) for Wireless Sensor Network in a LPL scenario. Our modified protocol, named REACTIVE, minimizes the reprogramming overhead of LPL, by dynamically shrinking LPL transmissions of Deluge T2 protocol. In this way we can obtain much better performance than the Deluge standard version running on LPL. Indeed, our experiments show that the image dissemination time is 7 times smaller than Deluge with LPL, while the energy consumption falls 2.6 times. The advantage of our solution is that it perfectly integrates in Deluge with LPL and can be complementary to other solutions seen in section II that are focused on image size reduction.

REFERENCES

[1] R. K. Panta, S. Bagchi, and S. P. Midkiff, "Zephyr: Efficient incremental reprogramming of sensor nodes using function call indirections and difference computation," in *Proc. of USENIX Annual Technical Conference*, 2009.

[2] W. Munawar, M. Alizai, O. Landsiedel, and K. Wehrle, "Dynamic tinyos: Modular and transparent incremental code-updates for sensor networks," in *Communications (ICC), 2010 IEEE International Conference on*, May 2010, pp. 1–6.

[3] W. Dong, Y. Liu, X. Wu, L. Gu, and C. Chen, "Elon: enabling efficient and long-term reprogramming for wireless sensor networks," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1. ACM, 2010, pp. 49–60.

[4] R. Panta, I. Khalil, and S. Bagchi, "Stream: Low overhead wireless reprogramming for sensor networks," pp. 928–936, May 2007.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *ACM SIGOPS operating systems review*, vol. 34, no. 5. ACM, 2000, pp. 93–104.

[6] M. Paoli, A. Lo Russo, U. M. Colesanti, and A. Vitaletti, "Magonode: Advantages of rf front-ends in wireless sensor networks," in *Real-World Wireless Sensor Networks*. Springer, 2014, pp. 125–137.

[7] D. Moss and P. Levis, "Box-macs: Exploiting physical and link layer boundaries in low-power networking."

[8] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[9] S. Michiels, W. Horré, W. Joosen, and P. Verbaeten, "Davim: a dynamically adaptable virtual machine for sensor networks," in *Proceedings of the international workshop on Middleware for sensor networks*. ACM, 2006, pp. 7–12.

[10] K. Langendoen and A. Meier, "Analyzing mac protocols for low data-rate applications," *ACM Trans. Sen. Netw.*, vol. 7, no. 2, pp. 19:1–19:40, Sep. 2010. [Online]. Available: http://doi.acm.org/10.1145/1824766.1824775

[11] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," pp. 81–94, 2004. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031506

[12] G. Tolle and D. E. Culler, "Design of an application-cooperative management system for wireless sensor networks." pp. 121–132, 2005. [Online]. Available: http://dblp.uni-trier.de/db/conf/ewsn/ewsn2005.html#TolleC05

[13] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, ser. NSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251175.1251177

[14] 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*. ACM, 2009, pp. 1–14.