Demonstrating the Versatility of a Low Cost Measurement Testbed for Wireless Sensor Networks with a Case Study on Radio Duty Cycling Protocols

Maite Bezunartea[1(✉)], Marie-Paule Uwase[1,2,3], Jacques Tiberghien[1,2], Jean-Michel Dricot[2], and Kris Steenhaut[1]

[1] Vrije Universiteit Brussel, ETRO, Ixelles, Belgium
{mbezunar,ksteenha}@etro.vub.ac.be
[2] Université Libre de Bruxelles, OPERA, Brussels, Belgium
Jean-Michel.Dricot@ulb.ac.be, muwase@etro.vub.ac.be, jacques.tiberghien@vub.ac.be
[3] National University of Rwanda, Butare, Rwanda

Abstract. Today, Wireless Sensor Networks (WSNs) with open source operating systems still need many efforts to guarantee that the protocol stack succeeds in delivering its expected performance. This is due to subtle implementation problems and unexpected interactions between protocol layers. The subtleties are often related to the judicious choice of parameters, in particular those related to timing issues. As these issues are often not visible in simulation studies, this paper proposes a low-cost versatile measurement testbed and demonstrates its usefulness in measuring the performance of RDC protocols. We demonstrate how the testbed helped to identify bugs in the implementation of an RDC protocol.

Keywords: Wireless Sensor Networks · Testbed · Field deployment · Measurements · Radio Duty Cycling · Contiki

1 Introduction

For studying the interactions between RDC and Routing over Low power and Lossy Networks (RPL) protocols for WSNs, Packet Delivery Ratio (PDR), packet latency and power consumption for the different RDC protocols available under Contiki on the Zolertia Z1 motes were measured, using the simple setup described in [1]. Unfortunately, this setup proved inadequate for accurately measuring packet latencies exceeding the inter-packet interval.

This was the motivation for designing a new, affordable, easily configurable testbed for exploring simple, point to point links as well as complex multi-hop networks. It is composed of two WSNs. One, called the *black* network runs the applications and protocols under study, the other called the *white* network observes the first one and transmits these observations to a sink node in which they are recorded (see Fig. 1). The *black* motes contain Z1 motes that belong to the observed network. Their built-in ceramic antenna limits its radio range to a few meters. The *white* motes are Z1 motes that belong

B. Mandler et al. (Eds.) IoT 360° 2015, Part I, LNICST 169, pp. 222–230, 2016.
DOI: 10.1007/978-3-319-47063-4_20

to the observer network. They use external antennas, allowing single hop communications with the sink. Both networks use different frequency bands, typically channel 26 for the *black* network and channel 16 for the *white* network.

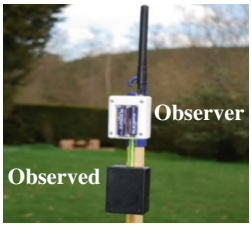

Fig. 1. A dual mote from the testbed.

2 Measurement Techniques

The *black* nodes run application programs that send short messages to each other. These messages are uniquely identified by the address of the node in which they were created and a local sequence number generated by the node. Whenever a *black* application program sends or receives a packet, it communicates the sequence number to the associated *white* node, which sends to the *white* sink a packet containing the sequence number as well as the power used by the *black* node since the last packet was transmitted or received. A computer connected to the *white* sink stores the received packets, together with the address of the sender and a timestamp that gives, according to the clock of the *white* sink, the moment the start of frame bit of the message arrives.

The recorded data allows a full inventory of the data packets transmitted and received by applications running on the *black* nodes, from which it is easy to compute the PDR, the per packet latency and the average power used in the initial sender and the final receiver.

2.1 The Link Between the *Black* and *White* Motes

Transferring data between two motes can be done via the serial USB port or via some of the parallel GPIO pins of the Z1. As transferring data through the serial port requires approximately 80 μs per character, which would disturb beyond reasonable limits the normal operation of the *black* network and might affect its power requirements, the parallel pins were chosen. A Z1 designer recommends to use GPIO pins 1.0, 1.6, 1.7, 2.3, 4.0, 4.2 and 4.3 as those are not used by the built-in features of the Z1s. Those pins of the *black* and *white* motes were directly connected with each other. Setting them requires ± 25 instructions in the *black* mote, which is negligible from a delay and power point of view. Six of those pins will transfer the least significant bits of the sequence number of a packet sent by a *black* mote. Pin 1.0 is a trigger for the *white* mote. At the very moment the toggling of pin 1.0 by the *black* mote gets detected by the *white* mote, this last one records this moment and sends a message to the *white* sink.

2.2 Measuring Technique for Packet Delivery Ratio, Latency and Power Usage

As discussed in Sect. 2.1, GPIOs carry the 6 least significant bits of the *black* packet sequence number. Matching the sequence numbers of transmitted and received packet descriptors received by the *white* sink allows to determine the PDR for each *black* end to end link.

We define the latency for a packet travelling from node A to B in the *black* network as the difference between the moment (denoted *m1*) at which the application in A offers the packet to the lower layer and the moment (denoted *m2*) at which the application in B has received the full packet. Calculating packet latency by including in the packet *m1* and subtracting it from *m2* is only correct when *m1* and *m2* are measured with synchronized clocks. Motes in the *black* network are seldom synchronized. As explained in Sect. 2.1, *m1* and *m2* are also recorded in the *white* motes, through the toggle pins. Simply deducing *m1* from *m2* is not appropriate as the clocks of the motes in the *white* network are not synchronized either, since synchronization overhead [5] would jeopardize its intended functioning. Fortunately, the sink will record the messages triggered by the toggles and timestamp their arrival. The difference between the two timestamps (denoted *m3* and *m4*) is a good estimation of the latency, on condition that both messages generated by the toggles undergo the same latency to reach the sink. This is realistic when no MAC nor RDC protocols are used in the *white* network. RDC protocols are not needed in the *white* network, as its power usage is of no importance, but a MAC protocol is needed to avoid loss of messages in the *white* network.

Therefore, a second, more accurate technique consists in measuring the latencies of the *white* messages and correcting the difference between the timestamps *m3* and *m4* by subtracting these latency measurements.

Figure 2 shows packet latency for a single hop transmission, composed of:

- The time interval between the moment the application offers the packet to the lower layer and the moment that the last bit of the start of frame is clocked out of that packet (re)transmission that will be acknowledged in the future. We call this interval the "sender latency", calculated as *tsr-tsa*. During this interval, several unsuccessful attempts to transmit can have been made.
- The time required for the physical propagation from sender to receiver. Considering the typical distances between motes in WSNs, this time can always be neglected (<100 ns).
- The time interval between detecting the start of frame byte and the arrival of the packet at the application layer in the receiver after verification of the destination address and the frame control sequence. We call this delay the "receiver latency", calculated as *tra-trr*. This delay results mostly from clocking in the packet and is almost constant for white messages.

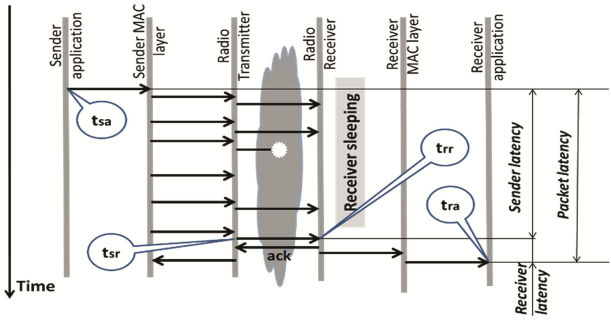

Fig. 2. Packet latency.

The sender latency can be measured by instrumenting the driver of the cc2420 radio in the white motes. It is possible to modify the contents of the last bytes of the transmitter FIFO while the first bytes are already being transmitted [2]. The modified driver polls the "start of frame detected" bit provided by the radio while the packet is being transmitted. When it goes high, the local real time clock *tsr* is read and its value is copied in the transmitter FIFO in the two last data bytes of the packet. The application program, just before passing the packet to the MAC layer has already read the local real time clock and inserted its value in another reserved location in the packet (*tsa*). When receiving a successfully transmitted packet, the *white* sink can compute the sender latency by subtracting from the time the successful packet was actually transmitted (*tsr*) the time it was passed to the sender's MAC layer (*tsa*), since these two times are both readings of the same real time clock in the *white* sender. Only when latencies exceed a full cycle of the mote's real time clock (2 s in our case), we use the Linux timestamps from the computer recording the *white* messages, to evaluate the number of clock cycles that need to be added to the computed packet latency.

For comparing the power requirements of different protocols, Contiki has four "energest" variables that are used to totalize the time during which the CPU, the sender and the receiver were active and the time during which the mote was "sleeping" [3]. By multiplying each of these times by the power required by the different components of a mote, one could compute the energy required for a given task. Hurni et al. [4] showed that this technique can result in high precision (1 %) power measurements, provided that the power requirements of the components of each mote are accurately known. They suggest to individually characterize each mote by means of current drain measurements with enough temporal resolution to distinguish the different states of the mote. A sensor node management device [5] sampling the current at 1000 Hz was used for that purpose. Another device, with even better resolution, was presented at the Como RealWSN workshop [6] in 2013.

However, many monolithic radios, and particularly the cc2420 [2] which is present in the Zolertia Z1 motes, are powered through a built-in DC to DC converter that keeps the voltage in the radio constant, regardless of the battery state. Observations by means of an oscilloscope showed that the activity type of the radio could not accurately be deduced from the instantaneous power drawn by the mote. Therefore, our performance

studies exclusively rely on power figures obtained by measuring the average current absorbed by the motes. The *black* motes are powered from the batteries of the *white* motes via a 1Ω series resistor between the ground lines, and decoupled by a 4700 μF capacitor, ensuring that no significant aliasing error will occur if the voltage across the resistor is sampled 100 times per second to record the current drawn by the *black* mote. Figure 3 shows how the *black* mote is powered and its current measurement amplified by a differential instrumentation amplifier before being sampled at 100 Hz and digitized by the built-in AtoD converter with sample and hold of the *white* mote.

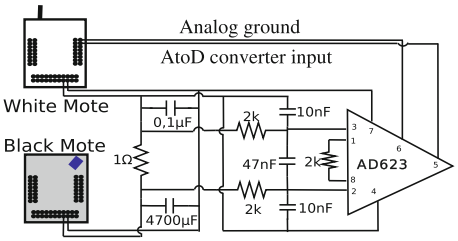

Fig. 3. Measuring black mote's power consumption.

3 Experimental Validation

As many hard to explain incidents were observed when RDC protocols on unicast single hop links were evaluated [1], a reevaluation of these protocols with better observation techniques was considered a good way to validate the new testbed and to get familiar with its possibilities and limitations.

The experimental set-up, shown in Fig. 4 consists in four dual motes and one *white* sink connected to a portable computer running the Linux Operating System (OS). A second PC with Windows is used to monitor the radio traffic by means of a Texas Instruments sniffer cc2531 and the associated free sniffer software SmartRF. This monitoring proves very helpful to quickly detect errors in the choice of communication parameters and/or detect mote malfunctioning.

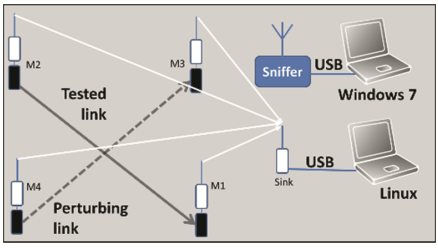

Fig. 4. Set-up for unicast tests.

The unicast link under study goes from the *black* mote M2 to the *black* mote M1. The behaviour of this link is obviously influenced by radio transmissions in the

neighbourhood. For studying unicast links, such unrelated radio-traffic is created by two communicating *black* motes M4 and M3. Sending and receiving of packets by the *black* motes is reported by the corresponding *white* motes to the *white* sink and recorded on the Linux PC. All dual motes are powered by AA alkaline batteries in the *white* motes.

3.1 The Radio Traffic on the *Black* Network

The traffic generated by the sending mote M2 consists in 59 bytes unicast messages transmitted at an average rate of 1 message per second (the inter-message interval is uniformly distributed between 10 and 1990 ms to prevent any correlation between the transmit cycle and the wake-up cycle of RDC protocols). In order to obtain statistically valid measurements, at least 1200 messages are observed per experiment. This results in 34 sets of 34 messages. Latency, PDR and power usage is computed for each set of messages allowing to calculate the 95 % confidence intervals. Perturbing traffic can be generated by M4 sending unicast messages to M3. To simulate the perturbing effect of a network where each mote has three reachable neighbors, M4, when active, sends 2 times more messages than M2. Lightly interfering traffic is generated when both M3 and M4 are active. Heavier traffic results from switching off M3 as then M4 will repeatedly try to transmit its packets to the non-responding M3. We name these three traffic classes NP for "No Perturbation", PR for "Perturbed with Receiver" and PNR for "Perturbed with No Receiver". The Contiki Rime software is used for managing the unicast transmissions and CSMA is used as MAC protocol. Four different RDC protocols, readily available on Contiki, have been tested. They are ContikiMAC [7], CXMAC (a version optimized for Contiki of XMAC [8]), Low Power Probing (LPP) [9] and, as a reference, NullRDC (this protocol leaves the radio always on). In case of LPP, tests with a perturbing sender and no receiver (PNR) should not differ from those with no external perturbation (NP), because a sender does not send unless invited by the target receiver; for that reason no "PNR" results are given for LPP tests. To the best of our knowledge, no extensive comparison of these protocols has been published. Only XMAC has recently been studied in depth [10].

3.2 The Protocols Used in the *White* Network

As in the *white* network power is not an issue while latency should be kept small, the choice of NullRDC is obvious. In Contiki, NullRDC is available with or without acknowledgements. Up to two retransmissions of unacknowledged packets by the MAC layer can reduce the risk of lost *white* messages, but at the price of a more variable latency. As differences in *white* latency can be compensated as described in Paragraph 2.2, NullRDC with acknowledgements has been chosen. Experimentally, we found that up to 10 % of the *white* messages got lost when acknowledgments were disabled, while no lost messages were observed when acknowledgments were enabled.

3.3 Packet Delivery Ratio, Latency and Power Measurements

After a series of tests (as described in Sect. 3.1) involving the four different Contiki RDC protocols the PDRs were computed and gave results above 98 %. These results confirm our previous experiments, reported in [1].

Extensive latency and power measurements had been done with much more limited technical means [1]. Repeating these measurements with the new testbed was a straightforward validation technique. Figure 5 shows some results of the latency measurements, together with the corresponding results obtained previously. These measurements were made with no perturbing traffic. The solid bars represent average values of all the measured latencies. One can observe that for ContikiMAC, CXMAC and LPP old and new measurements match very well. Only for NullRDC significant differences are visible, but, this is due to the use of Acks in NullRDC, while in the past acknowledging was deactivated as this was the default setting. The Ack activation was necessary for the fairness of the comparisons, as all other RDC protocols use Acks.

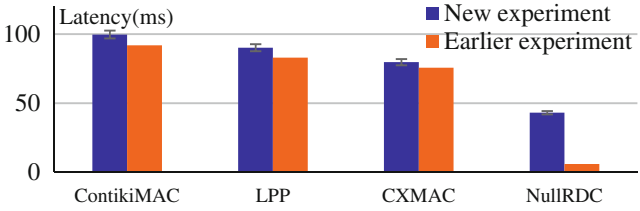

Fig. 5. Old and new measurement tool give the same results, except for NullRDC.

The next step in our validation experiments consisted in showing how RDC protocols trade latency for power under the three different traffic conditions. Figure 6(a) and (b) show respectively the latency and the power usage for ContikiMAC waking up 4, 8 and 16 times per second.

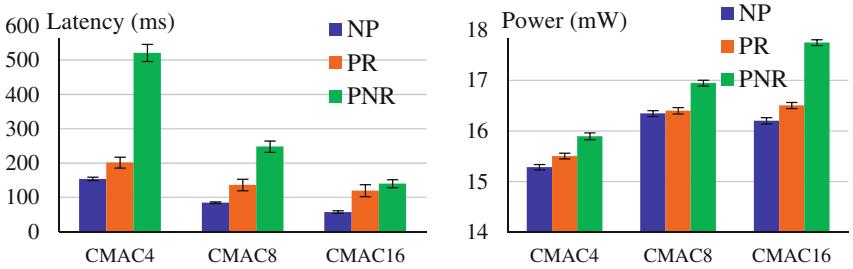

Fig. 6. Increasing the traffic density increases the (a) packet latency (b) power consumption.

For unperturbed traffic (NP) the average latency should be half the wake-up interval, augmented by the receiver delay and the occasional delays caused by transmission errors, as no collisions should occur. The observed latencies exceed on average the half wake-up interval by some 20 ms which seems normal. Perturbing traffic should increase

the sender delay through collisions. This effect is clearly visible and is very important when the wake-up interval is large as the perturbing link will almost continuously be transmitting. One could expect that the required power should grow linearly with the frequency of the wake-ups. Figure 6(b) shows this is true under heavy traffic conditions (PNR) but not under low traffic, whereas frequencies of 8 and 16 Hz have similar power needs. This observation deserves some further investigation.

The last step in our validation tests consisted in comparing latencies and power usage of the four RDC protocols available in Contiki, under three different traffic conditions. In order to be fair, we had to modify the default options in the Contiki implementation of the LPP protocol, as this was the only one that did not use a phase locking mechanism to try to predict when known neighbours would wake up. As the "Encounter Optimization" option is available in Contiki LPP, we enabled it for our comparisons with a wake-up frequency of 8 Hz.

The results presented in Fig. 7 show that RDC protocols trade latency for power. Surprisingly LPP with the encounter optimization shows worse latencies than in our initial experiments in which, by default, encounter optimization was disabled. Encounter optimization should reduce power consumption without affecting latencies.

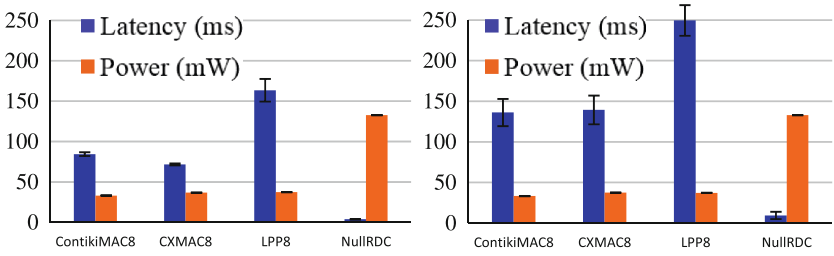

Fig. 7. RDC protocols trade latency for power (a) environment without perturbing traffic (b) with perturbing traffic.

Meanwhile, the responsible bug in the implementation of LPP has been fixed.

4 Conclusions and Future Work

The proposed testbed is versatile and low-cost. Due to the limited radio range of the *black* motes and the much larger range of the *white* ones, networks of any topology and size can be set up. The wireless observer network avoids the burden of using wired monitoring devices.

During the validation experiments, many anomalies observed in earlier tests, such as clustered packet losses in ContikiMAC and unexplained variations in the message latency with ContikiMAC and LPP showed up again, proving that they were not due to experimental artefacts but to undocumented particularities of the Contiki 2.6 implementation of the protocols.

The now validated testbed will enable us to pursue the in-depth study of these protocols. After further identifying the subtle implementation issues and fixing them, we will

start with our ultimate goal: understanding in detail the interactions between the routing layer and the RDC-MAC layer.

References

1. Uwase, M.-P., Bezunartea, M., Nguyen, T.L., Tiberghien, J., Steenhaut, K., Dricot, J.-M.: Experimental evaluation of message latency and power usage in WSNs. In: 2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 69–72. IEEE (2014)
2. Texas Instruments: Datasheet: CC2420 2.4 GHz IEEE 802.15.4/ ZigBee-ready RF Transceiver. http://www.ti.com/lit/ds/symlink/cc2420.pdf
3. Dunkels, A., Osterlind, F., Tsiftes, N., He, Z.: Software-based on-line energy estimation for sensor nodes. In: Proceedings of the 4th Workshop on Embedded Networked Sensors, pp. 28–32 (2007)
4. Hurni, P., Nyffenegger, B., Braun, T., Hergenroeder, A.: On the accuracy of software-based energy estimation techniques. In: Marrón, P.J., Whitehouse, K. (eds.) Wireless Sensor Networks, pp. 49–64. Springer, Heidelberg (2011)
5. Hergenroeder, A., Wilke, J., Meier, D.: Distributed energy measurements in WSN testbeds with a sensor node management device (SNMD). In: 23rd International Conference on Architecture of Computing Systems (ARCS), pp. 1–7. VDE, Hannover (2010)
6. Buschhoff, M., Günter, C., Spinczyk, O.: MIMOSA, a highly sensitive and accurate power measurement technique for low-power systems. In: Langendoen, K., Hu, W., Ferrari, F., Zimmerling, M., Mottola, L. (eds.) Real-World Wireless Sensor Networks. LNEE, vol. 281, pp. 139–151. Springer International Publishing, Switzerland (2013)
7. Dunkels, A.: The ContikiMAC Radio Duty Cycling Protocol (2011)
8. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems - SenSys 2006, pp. 307–320. ACM Press, New York (2006)
9. Musaloiu-E, R., Liang, C.-J.M., Terzis, A.: Koala: ultra-low power data retrieval in wireless sensor networks. In: 2008 International Conference on Information Processing in Sensor Networks (IPSN 2008) (2008)
10. Beaudaux, J., Gallais, A., Montavont, J., Noel, T., Roth, D., Valentin, E.: Thorough empirical analysis of X-MAC over a large scale internet of things testbed. IEEE Sens. J. **14**, 383–392 (2014)