WaCo: A Wake-Up Radio COOJA Extension for Simulating Ultra Low Power Radios

Rajeev Piyare[1,2], Timofei Istomin[2], Amy L. Murphy[1],
[1] Bruno Kessler Foundation, Italy {piyare, murphy}@fbk.eu
[2] University of Trento, Italy timofei.istomin@unitn.it

Abstract

Radio communication remains the primary battery consuming activity in wireless systems. Advances in MAC protocols have enabled significant lifetime improvements, but in systems with low data rate, idle listening, and other communication artifacts can begin to dominate costs. One proposal to combat this is the addition of a second, extremely low power radio component that is always-on. As a consequence of the extremely low power, such radios are incapable of decoding general data, and thus are often delegated the task of listening for a trigger, leading to the terminology *wake-up radio*, as this extremely low power radio is used to wake up a higher power radio, which is then used for data communication. While wake-up technology has been steadily evolving over the last decade in the hardware arena, few protocols have been developed to exploit it. In this work, we present `WaCo`, our wake-up radio COOJA extension that allows exploration of the capabilities of the wake-up radio from the desktop environment. We also use our extended simulator to concretely show the potential benefits of the wake-up radio hardware with two, standard data collection protocols. Our results simultaneously confirm that wake-up technology has tremendous potential and that our simulator extension provides an effective mechanism for such exploration.

Categories and Subject Descriptors

I.6.4 [**Computing Methodologies**]: Simulation and Modeling, Model Validation and Analysis; D.2.5 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Design, Simulation, Performance, Reliability

Keywords

Wake-up Radios, Energy Efficiency, Wireless Sensor Networks, Contiki Collect, RPL, Data Collection

International Conference on Embedded Wireless Systems and Networks (EWSN) 2017
20–22 February, Uppsala, Sweden

ISBN: 978-0-9949886-1-4

1 Introduction

Extending the battery life of sensor nodes has been one of the primary research focuses since the introduction of wireless sensor networks (WSNs) and its integration with the Internet of Things (IoT) paradigm.

Most approaches to energy savings focus on the software side, addressing expensive communication activities. For example, work at the MAC layer [6] trades off energy consumption, latency, throughput and fairness. While radio duty cycling significantly decreases consumption, several shortcomings remain such as idle listening, overhearing and costly continuous retransmissions at the sender.

Recently, a new hardware technology promises to directly combat these problems [9] by offering an ultra low-power radio, with consumption three orders of magnitude less than the typical low power radios used in WSNs. To offer an example, the radio we consider here consumes 1.94 μW for listening, in contrast to the CC2420's 56.4 mW. To reach such low numbers, these novel radios have very limited capabilities. For example, most prototypes are unable to transmit arbitrary data, and are instead used to exchange a simple signal, which is used to trigger the primary radio, thus the terminology *wake-up radio* (WuR), indicating that the receipt of a signal on the ultra-low power radio is used to wake-up a primary radio for regular data exchange.

WuRs are quite promising to extend lifetime, yet the technology remains in its relative infancy. While several prototypes exist in the laboratory environment, it remains difficult to experiment with these devices. Further, the exact potential of this technology on the WSN system as a whole has not been demonstrated.

Therefore, the goal of this paper is to offer a software-only, simulation environment to enable the systematic exploration of the potential of WuR technology. This exploration must allow for the development, evaluation and evolution of protocols across all layers of the software stack as well as the evaluation of the applications in the target environments. To this end, we present `WaCo`, a set of extensions to the COOJA simulator [10] and Contiki operating system that allow prototyping of protocols and applications that can exploit a standard WSN mote extended with a new, simulated WuR hardware module. Our design allows the specification of the WuR to be easily modified, allowing researchers to plug in their own WuR module.

We are not the first to propose simulation of WuR, as

connectivity requirements have been explored analytically in MATLAB [11] and the relationship between hop counts, effective-WuR range, and PDR was explored in [13]. Further, OMNET++ [9] and GreenCastalia [1] provide simulation models for WuR based systems. Nevertheless, none of these consider the entire stack. WaCo, instead, directly uses binary, deployment-ready code, and therefore provides the ability to move between simulated and real experiments. Further, it allows simulation of multiple embedded operating systems.

This paper presents the relevant background in WuRs and simulation (Section 2), then outlines the design of WaCo's COOJA and Contiki extensions (Section 3). We also provide a straightforward MAC module, called W-MAC (Section 4), which uses the WuR as a trigger for the standard CC2420 radio, and offers the same interface as other popular MAC protocols, allowing it to easily sit below standard routing protocols. We then offer an evaluation of a standard data collection system (Section 6) with two primary goals: to demonstrate the benefits of WuR and to show effectiveness of WaCo at providing this evaluation. Finally Section 7 provides concluding remarks and possible future directions.

2 Background

We begin by offering background on the characteristics of WuRs and how they can be combined with a traditional mote. We also provide details of COOJA, as it is the core simulation environment for WaCo.

2.1 Ultra Low-power Wake-up Radios

Recent developments in CMOS power consumption have led to the new design paradigm of WuRs that greatly reduces power consumption. A typical WuR is composed of both a wake-up receiver (WuRx) and a wake-up transmitter (WuTx). The former consumes several orders of magnitude less power than a traditional low-power radio, allowing it to be always-on without significantly depleting the battery.

Characteristics. The most critical feature of the WuRx is its low power consumption. Design for this at the hardware level has the concrete effect to limit signal detection capabilities and operations that can be preformed by the WuRx. Further, such constraints limit the choice of modulation schemes and receiver complexity, and, as a consequence receiver sensitivity, ultimately reducing the communication range.

Most prototype WuRxs are limited to receiving a very small packet of up to 16 bits, and, in some cases, interpreting whether or not the contents of the packet match the address assigned to the node [8], offering a unicast wake-up mode. This is motivated by the observation that decoding the address consumes energy, typically at an external low-power micro-controller (MCU). With few address bits, the demodulation at the WuRx is very simple, reducing the power budget as no control or formal check on the demodulated packet is usually required.

Operation. A typical system couples a WuR with a higher-power radio, resulting in a dual-radio architecture. When used as a wake-up device, a remote node transmits a wake-up signal (WuS) using the WuTx. The WuRx detects this WuS and generates an interrupt to the main nodes MCU to switch it from sleep to an active mode. Then, the MCU turns on the primary transceiver to exchange data packets with the initiator node in a conventional manner.

Benefits. The idle listening of a standard, low-power radio contributes significantly to the overall energy consumption of duty-cycling nodes. The on-demand WuR approach minimizes this unnecessary energy wastage, as the main radio and the node will be activated only when there is actually data to transmit. Further, WuRs with an addressing mechanism can be used to reduce overhearing. Finally, since the WuRx can be kept always-on, continuous transmissions will be avoided as the sender will know precisely when the receiver will be awake, ready to receive the data.

2.2 WSN Simulation Environment: COOJA

Our goal is to provide an environment in which to explore the potential benefits from the WuR technology without requiring significant hardware investment. COOJA offers a solid environment for such testing, however it does not support multiple radios on a single hardware platform, nor does it offer modules to simulate WuR. At the same time, COOJA directly supports the integration of multiple hardware models in the form of the platform, offering a clear place for expansion to incorporate the WuR. Further COOJA exploits MSPSim to allow software designed and compiled for real hardware such as the TMote Sky platform to be directly used for simulation. MSPSim emulates the MSP430 MCU and the CC2420 transceiver at the instruction level, offering very fine-grained, low-level simulations.

COOJA also allows developers to test and run their applications using different radio models on fully emulated hardware devices, a functionality not available in other simulators such as Castalia [2] and MiXiM [5].

Above the hardware, COOJA also enables simulation of systems with various operating systems designed for resource constrained devices. For this work, we chose ContikiOS [3], which itself contains several protocols from the application to the physical layers, as shown in Figure 1b and two networking stacks: Rime [4] and uIPv6.

3 WaCo: Enabling Support of WuR in Simulation

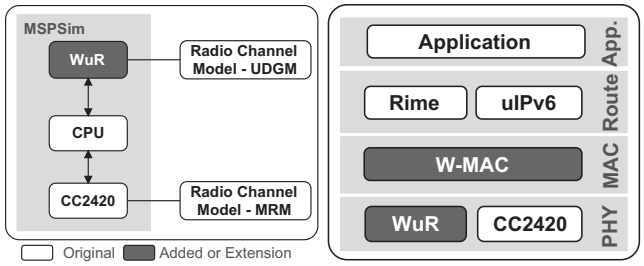
To enable simulations with WuR technology, modifications of both the simulator and the mote software are required. The simulator must be extended with a simulated hardware module (WuR chip) and support for multiple radio channels, able to be used simultaneously and independently. Additionally, the visual simulator plug-ins that simplify debugging and measuring system performance must be added.

The mote software requires changes to the OS and protocols that interface with the newly added hardware modules. For this, we modify Contiki, providing a new physical layer interface for the WuR and a new MAC protocol that uses both the new WuR and the standard CC2420 data radio.

This section details the hardware and software extensions that form the core of WaCo, summarized in Figure 1, while Section 4 provides details of our MAC protocol.

3.1 COOJA Platform Extensions

In COOJA, node hardware is represented as a platform description, containing models for the MCU, radio, sensors,



(a) Node and radio modules in COOJA

(b) ContikiOS networking stack

Figure 1: The WuR module extensions of `WaCo` to COOJA and ContikiOS.

memory, and other hardware components. We chose to extend the existing TMote Sky platform, a popular platform in the community. Further, because COOJA integrates MSPSim, simulations exploiting the TMote platform can be executed directly with code compiled for real nodes, simplifying the development process. Our description of the components specific to the WuR appear in three places in the COOJA system: the platform, the simulator core, and in its plugins.

First, the emulated TMote platform of COOJA is extended with a module, represented by WuR in Figure 1. The module implements the hardware interface between the WuR and the CPU, based on I/O ports and a memory buffer. Furthermore, it defines the operational aspects of the wake-up transceiver, e.g., its transmission rate. Conceptually, this interface is similar to that of the pre-existing CC2420, with interrupt processing to receive an incoming WuS and a transmit operation to send either an addressed unicast or broadcast signal. Additional operations enable reading and writing to the memory of the processor to communicate the destination address received or to be transmitted. While these operations cover the functionality of most WuRs found in the literature, our interface can easily be extended. For example, some radios additionally support duty cycling, necessitating extensions to turn on and off the WuR.

Second, in the core of COOJA we implemented support for multiple independent radio channels to be used by multiple-transceiver platform. The extension allows simulating an arbitrary number of radio channels, with a separate signal propagation model associated to each of them. For our extended platform we use two radio channels.

Third, COOJA uses a plugin architecture to extend its core functionality. Our wake-up extension affects two plugins: the power tracker and timeline. Accurate power profiling requires augmenting COOJA with a second instance of the power tracker plugin associated with the WuR. Thanks to our extension it is possible to record, for each radio, the time spent in different states. Efficient debugging requires a visual representation of the network behaviour on a unified timeline showing events generated by both radio transceivers.

Regarding the radio channel model, most COOJA users opt for the Multi-path Ray-tracer Medium (MRM), which we also apply to the CC2420. Nevertheless, for the WuR, we use the Unit Disk Graph Medium (UDGM) with constant loss. While UDGM is simplistic, its use reflects our current goal of creating a generic model of WuRs.

3.2 Contiki OS extensions

We now turn our attention to extensions for `WaCo` on the mote software side, as shown in Figure 1b. Most significantly, we wrote a new physical layer module for Contiki to wrap the interface of the WuR. This interface follows that of the analogous module for the CC2420, with operations to support two-way data communication, i.e., handling interrupts to the CPU when the node receives the signal over the wake-up channel and triggering transmission of wake-up signals, optionally specifying the destination address. We also implemented a MAC protocol, outlined in the next section.

4 W-MAC in a Nutshell

W-MAC, our MAC protocol for the dual-radio setup, has been implemented to offer the same interface to higher layer protocols as the popular ContikiMAC, making it seamlessly interoperable. The major contribution of W-MAC is its coordination of the dual radios, using an always-on WuRx as a trigger for activating the CC2420.

W-MAC assumes all network nodes host both radios, naturally forming a multi-hop network in which nodes alternately act as senders and receivers. Further, W-MAC is a sender-initiated protocol in which the message source triggers the receiver to wake up.

The basic operation of W-MAC is as follows: when a node has data to send, either generated from the upper layers of the protocol stack or forwarded by another node, W-MAC first transmits a WuS containing the address of the destination. It should be noted that the two networking stacks of Contiki use different layer 2 address sizes, specifically the Rime stack uses 2 B while uIPv6 requires 8 B. W-MAC allows these different network address sizes, with corresponding consequences on the costs due to increased transmission and demodulation times.

On the WuRx side, the receipt of the WuS matching the receiver's address triggers the activation of the CC2420 in receive mode. If no data packet is received within a predefined time due to interference or collisions, the receiver switches the CC2420 back into sleep mode, keeping the WuRx actively listening for subsequent signals. Instead, if a packet is received on the CC2420, an acknowledgment (ACK) is sent, then the receiver's primary radio is turned off. Upon receipt of the ACK, the transmitter's primary radio is similarly turned off. On the sender side, after the WuS is transmitted, the node waits for a short period of time during which it expects the receiver's CC2420 to be switched into receive mode. It then transmits the data on the main, data radio, switches into receive mode to receive the ACK, then goes back to sleep. If the sender does not receive the ACK within a certain time interval, it will return to the beginning of the sequence, re-transmitting the WuS.

Thus far we have only discussed unicast, addressed transmission, however our WuR also supports broadcast. In this case, the receipt of the WuS causes all receiving nodes to switch on their primary radios. As before, the sender waits a short period, then transmits the data. However, unlike unicast mode, broadcast transmissions are not acknowledged, allowing the sender to immediately switch off the CC2420 after transmission and the receiver to switch off after receipt.

As is typical, our WuR and main radio use different channels, eliminating the possibility of collisions between the WuS and the data packets. However, collisions can still occur between concurrent wake-up signals or concurrent data transmissions. In our current implementation, channel sensing is only performed by the main radio before transmission, allowing us to avoid most data packet collisions. For this, we use Contiki's default CSMA mechanism and if the data channel is busy due to an on-going transmission or reception, the transmitting node backs off for a random period before retransmitting the WuS. The drawback of performing carrier sense just before data transmission is the penalty of increasing the on-time for the main radio at the receiver, which was awakened by the WuS, but due to the data channel being busy, the data transmission cannot proceed. To overcome this, an alternative solution could be to use WuR for channel sensing and reservation rather than the main radio. We plan to investigate the benefits of this in the future.

5 WaCo Evaluation Settings

To evaluate WaCo, we demonstrate its use through the simulation of a data collection application, a common scenario for low-power WSNs. First, however, we recall that our objective is to answer the following two questions: to what extent can battery-powered networks be made energy efficient by equipping them with WuR technology and can WaCo be used to demonstrate this.

We describe our simulation settings, beginning with the lowest physical network topology and progressing up through the routing and application layers.

Network Topology. To avoid the bias of network density, we established a 10 by 10 grid of 100 nodes. The horizontal and vertical distance between nodes is 30 m and we place a single sink node close to the center of the grid.

Routing Protocols. The architecture of WaCo and W-MAC allows us to run different routing protocols, as long as they utilize the standard MAC interface. In this work, we experiment with *Contiki Collect* and *RPL* [12]. Together they represent the current state-of-the-art in data collection and are staple references for many-to-one scenarios.

MAC Protocols. The routing protocols described above sit on top of the MAC layer, whose primary objective is to control the radios, including duty cycling. For ContikiMAC, the sleep interval was set to 125 ms with the phase lock-mechanism enabled. In our case, NullRDC serves as a comparison baseline. The most significant difference between W-MAC and the others is its ability to exploit the WuR, while the others use only the CC2420.

Simulation Parameters and Metrics. The protocols mentioned above are highly customizable with parameters such as buffer sizes, timeouts, retries and maximum hop count. We used the default values unless otherwise noted. The CC2420 radio chip is configured using MRM with a mean noise value of -80 dBm and noise variance of 4 resulting in a $\sim$65 m transmission range. For UDGM, associated with WuR, the transmission range is fixed to 50 m with a success ratio of 100% and interference range of 50 m.

In each of our simulation scenarios, every source node generates 6 B data packets at a fixed inter packet interval (IPI). The WuR signals are sent at 100 kbps, and consist of 2 B of address data for Rime and 8 B for uIPv6. Each run simulates 40 minutes of runtime with the first 10 minutes serving as a burn-in time for the routing and MAC protocols. The subsequent 30 minutes are analyzed and results reported for i) *power consumption*, ii) *reliability* and iii) *end-to-end packet latency*. Each point on our plots in Section 6 represents the network-wide average of five simulated runs.

Table 1: Power consumption of the CC2420 and WuR. Idle reflects listening to the channel, but not actively receiving.

	Power Consumption (mW)		
Parameters	RX	TX	Idle
CC2420	56.4	52.2	56.4
WuR	0.144	8.38	0.001944

WuR Hardware Parameters. For evaluation purposes, the WuR solution that we adopted is a custom designed ultra low-power module [7], representative of most WuRs found in the literature. Its performance parameters are set to values obtained from actual, in-lab measurements. The WuR board includes two modules: the addressable WuRx and a WuTx. The WuRx module operates in the ISM 868 MHz band and has a high receiver sensitivity of -55 dBm with a maximum communication range of 50 m. The power consumption of the WuRx is 144 μW in receiving mode and of 1.944 μW when it is in an idle state, listening for the signal. The power consumption of the WuTx is measured to be 8.3 mW when transmitting at +10dBm.

While the experiments presented here reflect the specific properties of our prototype WuR, the results can be generalized to the class of WuR. Further, as new technologies develop, WaCo can be easily extended with the hardware parameters of other modules, as outlined in Section 3.1.

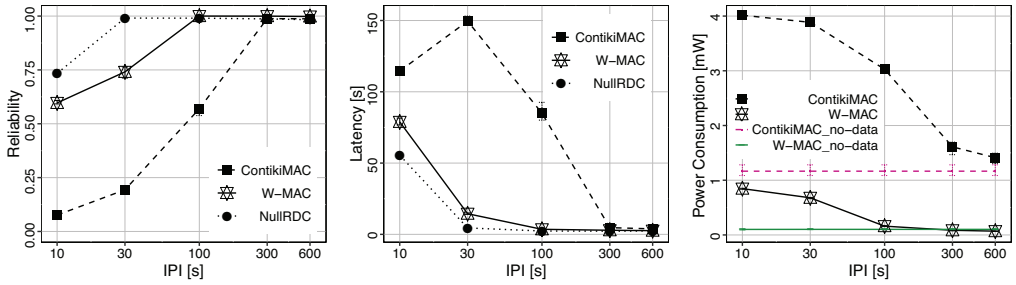
6 WaCo for Network Performance Analysis

WaCo enables evaluation of systems with WuRs in two ways. First, W-MAC, as a part of WaCo, can be seamlessly substituted for the stock ContikiMAC. Further, other MAC protocols employing WuRs or cross-layer solutions can be evaluated in WaCo by exploiting its WuR model and the plugins for power profiling and visual debugging.

Here, we concretely demonstrate how WaCo can be used to investigate the performance gains provided by W-MAC when integrated in an networking stack. Our tests vary the network load to determine how the choice of a MAC affects network capacity by exploring a range of IPIs from 10s to 600s, with the lower IPIs showing performance in a highly stressed setting. We also experiment with no data traffic, zero-IPI, observing the network without any application load to offer a baseline for the maximum power savings achievable. We first focus on the results of Contiki Collect, then briefly discuss the performance differences of the uIPv6 stack with RPL.

6.1 Contiki Collect with WuR

We begin our evaluation by considering the data collection reliability, shown in Figure 2a. At the smallest IPI, the highest load, all the protocols suffer from overload showing reliability below 75%. As expected, ContikiMAC demonstrates the smallest network capacity due to its duty-cycling



(a) Reliability (b) Latency (c) Power consumption.

Figure 2: Contiki Collect evaluation collected with WaCo.

Table 2: Summary of network-wide performance gain of W-MAC over ContikiMAC

Network Stack	Power	Reliability	Latency
Contiki Collect	4.75–20.40×	1.00–7.85×	1.45–23.94×
RPL	3.00–8.40×	1.00–3.69×	9.18–228.50×

nature. The nodes only have one chance to receive a single packet in each sleep interval, configured as 125ms in our simulations. NullRDC, instead, is on the other end of the spectrum, showing the highest capacity. Packets are tightly packed into the available bandwidth with no additional delays besides CSMA backoffs. W-MAC stands in the middle, albeit closer to NullRDC, due to delays imposed by the signaling of the WuR and the activation of the data radio. These delays are much smaller than the duty cycle period of ContikiMAC, allowing it to push more packets through. The same reasoning explains the differences in average latency shown for the three MAC protocols.

The high load negatively affects both reliability and latency as higher contention means longer backoffs and longer packet queues. In overload conditions the queues overflow and packets are dropped. As we decrease traffic with higher IPIs, all MAC protocols achieve perfect reliability and the latencies in line with those of NullRDC. Notably, each MAC protocol achieves perfect reliability at a different IPI, reflecting the ability to handle higher bandwidth due to the different times required to transmit each packet. W-MAC improves the network capacity over ContikiMAC, handling the IPI of 100s, while the latter delivers all packets only at IPI $\geq$ 300s. At low load, the latency of all protocols become very similar.

We now turn our attention to the power consumption shown in Figure 2c. As NullRDC keeps the main radio transceiver always on, its power consumption is several orders of magnitude higher than the others, averaging $56mW$. Therefore our plots only show power consumption for W-MAC and ContikiMAC. We also offer two lines showing the consumption for zero-IPI, or no data. This concretely shows the baseline consumption required to maintain the collection topology, separating out the additional cost to transmit data.

Most importantly, we note that in *all* scenarios, the addition of the WuR yields significant savings, as summarized in Table 2. With no data, consumption drops dramatically from 1.01mW to 0.1mW. In the setting when both protocols achieve high reliability, namely the IPI of 300, the data traffic of ContikiMAC pushes consumption above the baseline to 1.6mW while W-MAC is very close to its baseline. The reason for this significant difference is that W-MAC reduces idle listening of the power-hungry main radio and does not need long packet trains to deliver the packet as in duty-cycling MAC protocols, thus achieving short transmission times. While ContikiMAC does mitigate this by starting the packet train close to the end of the sleep interval of the destination node, this estimation is not perfect. Instead, the signaling of the WuR ensures tight timing, except in overloaded scenarios.

When the system is overloaded, both protocols show increased power consumption because of channel contention that leads to collisions and retransmissions. As the IPI decreases the effect of data traffic gradually reduces until the system lifetime is dominated by protocol overhead.

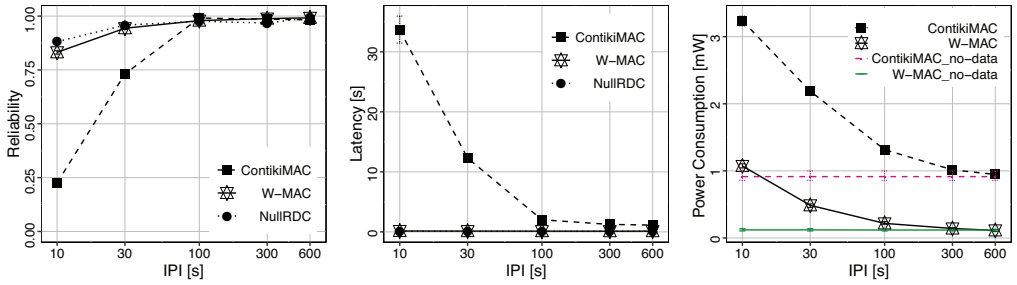
6.2 RPL with WuR

To demonstrate the capabilities of WaCo, we offer a brief evaluation of RPL, shown in Figure 3 on top of ContikiMAC, W-MAC, and NullRDC. Interestingly, the overall network performance achieved by RPL is better than that of Contiki Collect. The reliability recorded for all three MAC protocols is higher at the respective IPIs, moreover ContikiMAC shows perfect reliability starting from the IPI of 100s.

Similarly, the latency and power consumption in absolute terms are lower than those of Contiki Collect. However, in relative terms W-MAC achieves less improvement over ContikiMAC. One of the reasons is that the wake-up address in RPL is configured to 8B, which is 4× longer than the Rime node address. Therefore, the WuR spends more time transmitting the addresses due to longer bit duration. On average the power consumed by WuRs is 1.1× more than that of Contiki Collect in the no-data scenario.

6.3 Validation

Finally, we validate our simulation results against a small, hardware setup with two Tmote Sky motes, one connected to a WuRx and the other to a WuTx. Figure 4 shows a COOJA timeline trace and a trace collected by an oscilloscope. The traces show the exchange of the WuS followed by the exchange of a data packet on the main radio. In the COOJA timeline, blue indicates transmission and green reception, with the WuR exchange lasting longer than data exchange. This is due to the fact that we ran our simulation at 1Hz, so that it matches the restriction of the research prototype we compare to here. Note that in the simulations above, we ran the WuR at 100Hz, a value more common to WuR prototypes found in the literature. As can be seen in the table,



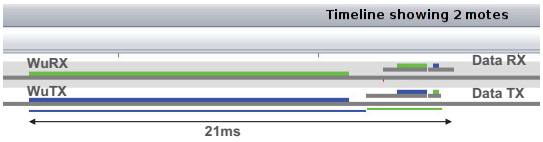
(a) RPL Reliability (b) RPL Latency (c) RPL Power consumption.

Figure 3: RPL evaluation collected with WaCo.

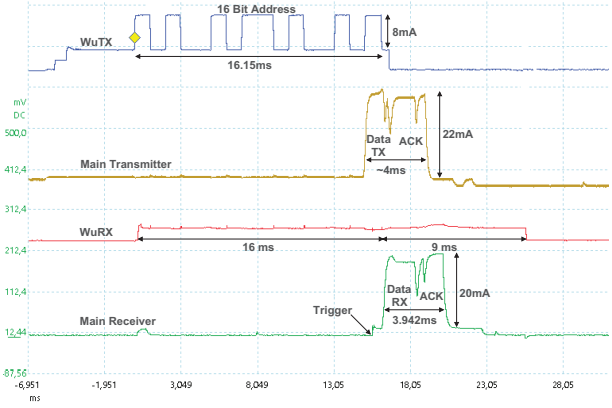
the consumption we measure in COOJA is in line with that measured in the lab, validating our simulation environment. Small differences account for variations in hardware configurations, such as keeping the WuRx active for a short period after reception ends.

Events	WaCo (mW)	Hardware (mW)
WuTX	384	387
WuRX	202	129
Data TX	250	264
Data RX	210	236

(a) Table 3: Measured power consumption using WaCo and hardware



(b) Cooja-WaCo timeline trace and radio events



(c) Oscilloscope trace for radio events

Figure 4: WaCo Validation.

7 Summary and Future Directions

Our stated goals were two fold: demonstrate the potential of the novel WuR technology, and offer a tool to provide this evaluation. Toward the first goal, we have shown that with a representative WuRx, data collection networks can be built using a standard network stack. The results show that the WuR technology has the potential to offer significant energy savings without compromising on reliability and latency. Actually, in terms of reliability, due to the brief transmission duration fostered by using the WuR as a signaling technology for a higher power radio, systems can support higher throughput. For the latter, we have demonstrated that WaCo provides an effective environment for analysis of protocols proposed for WuR technology.

Future work will involve further testing of the WaCo environment with different WuR hardware specifications from the literature, the development of novel MAC and routing protocols, possibly incorporating an acknowledgment mechanism at the WuR level, and cross layer protocols to better exploit the features of the WuR through the routing and application layers.

8 References

[1] D. Benedetti, C. Petrioli, and D. Spenza. GreenCastalia: An Energy-Harvesting-enabled Framework for the Castalia Simulator. In *Proceedings of the 1st International Workshop on Energy Neutral Sensing Systems*, page 7. ACM, 2013.

[2] A. Boulis. Castalia: Revealing Pitfalls in Designing Distributed Algorithms in WSN. In *SenSys*, pages 407–408. ACM, 2007.

[3] A. Dunkels, B. Gronvall, and T. Voigt. Contiki- A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *IEEE LCN*, pages 455–462, 2004.

[4] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *SenSys*, pages 335–349. ACM, 2007.

[5] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Haneveld, T. E. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating Wireless and Mobile Networks in OMNeT++ the MiXiM Vision. In *Simutools*, page 71, 2008.

[6] K. Langendoen. The MAC Alphabet Soup Served in Wireless Sensor Networks, 2004.

[7] M. Magno and L. Benini. An Ultra Low Power High Sensitivity Wake-up Radio Receiver with Addressing Capability. In *IEEE WiMob*, pages 92–99, 2014.

[8] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl. Performance Evaluation and Comparative Analysis of SubCarrier Modulation Wake-up Radio Systems for Energy-Efficient Wireless Sensor Networks. *Sensors*, 14(1):22–51, 2013.

[9] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl. Has Time Come to Switch From Duty-Cycled MAC Protocols to Wake-up Radio for Wireless Sensor Networks? *IEEE/ACM Transactions on Networking*, 24(2):674–687, 2016.

[10] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *IEEE LCN*, pages 641–648, Nov 2006.

[11] R. Su, T. Watteyne, and K. S. Pister. Comparison between Preamble Sampling and Wake-up Receivers in Wireless Sensor Networks. In *IEEE GLOBECOM*, pages 1–5, 2010.

[12] T. Winter, T. Pascal, B. Anders, W. H. Jonathan, and K. Richard. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *IETF*, 2012.

[13] Y. Zhang and G. Dolmans. Wake-up Radio Assisted Energy-aware Multi-hop Relaying for Low Power Communications. In *IEEE WCNC*, pages 2498–2503, 2012.