Performance Evaluation of Energy Saving MAC Protocols in WSN Operating Systems

Mike Ojo, Davide Adami and Stefano Giordano
Department of Information Engineering
University of Pisa
Via Girolamo Caruso 16 - 56122 Pisa, Italy
mike.ojo@ing.unipi.it; (d.adami; s.giordano)@iet.unipi.it

***Abstract*—Sensing, computing and communication are the main features of a wireless sensor network (WSN) which serves a wide range of applications. Despite its versatility and simplicity, it brought up various challenges such as limited storage, power consumption from radio activities, just to mention a few. The distinguishing traits of sensor networks have a direct impact on their protocol design at each layer, especially at the Medium Access Control (MAC) layer since it manages transmission scheduling as well as duty cycling for energy conservation. To maximize energy efficiency of WSNs, a critical analysis of the radio duty cycle of the WSN operating systems were carried out with experimental evaluation. Moreover, we focus on the energy consumption by conducting experimental measurements on different platforms i.e. OpenMote-CC2538 on various operating systems. Results shows that IEEE 802.15.4e Time Synchronized Channel Hopping (TSCH) has great impact on the energy consumption with respect to other radio duty cycles.**

***Index Terms*—Wireless Sensor Networks; Radio Duty Cycle; Operating Systems; Energy Consumption; Sensor Node; TSCH**

I. Introduction

Over the last decade, there have been tremendous advances in Micro-Electro Mechanical Systems (MEMS) based technology which has led to the rise and development of small shaped, low cost, low power, wide spread distributed multifunctional devices. These devices are called sensor nodes with wireless communication capabilities that interact together to form a network called Wireless sensor networks (WSN). WSN are composed of small nodes with sensing, data processing and communication components [1]. It also enables interaction between persons or computers and the surrounding environment which can be used for different types of applications [2]. This has made WSN to be popular, demanding and indispensable in different industrial sectors. Moreover, since most sensor nodes are battery powered, limited battery capacity can constrain the overall lifetime of the wireless sensor network.

Energy efficiency in wireless sensor networks is a big challenge and has drawn significant attention to both academic and industrial areas over the past years as technology solutions nowadays do not require frequent replacement of batteries. Also, the replacement of embedded batteries can be a difficult process once the nodes have been installed due to large network size and also deployment to hazardous environment. Hence, it is very necessary to estimate and determine the energy consumption of a wireless mote before deployment. In addition to that, sensor nodes need to optimize their energy usage for network lifetime.

Several efforts in the literature have addressed energy efficiency in sensor networks, through the design of radio duty cycle (RDC) mechanism which schedules the data transmission, data reception and inactive sleeping periods at regular interval. The node's duty cycle is a function of the MAC layer and has a significant factor in reducing the energy consumption in wireless sensor networks. We investigate into the RDC protocols of Contiki [3], RIOT [4], OpenWSN [5] operating systems (OS) which are perhaps the most recent and well known systems in the sensor network research community.

The rest of this paper is organized as follows. Section II discusses the network stack of the WSN operating system used in our experimental evaluation. Section III describes the radio duty cycle used in the WSN OS above and it also captures their comparison illustrating their advantages and disadvantages. Energy consumption of a mote by using different operating systems were derived in section IV. Section IV also describes the results and the comparison and finally we conclude this paper in Section V along with the discussion of the future work.

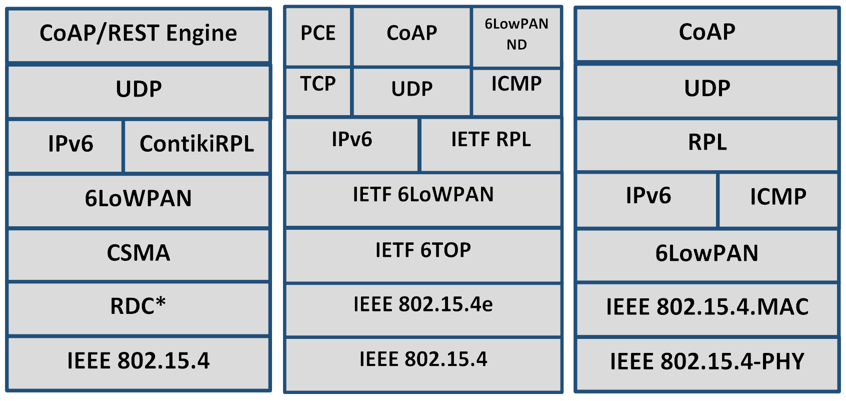
II. Operating Systems

We take a look at the network stack of OpenWSN, Riot and Contiki Operating system

A. Contiki OS Network Stack

Contiki is an open source, highly portable, multi-tasking operating system for memory-efficient networked embedded systems and wireless sensor networks. Contiki OS network stack is shown in Fig 1a. At the lower layer of the stack comprises of the Physical layer and the MAC layer. Traditionally, duty cycling mechanisms are built into the MAC layer of the operating systems networking stack. Contiki, however, employs duty cycling tactics in a separate level of the networking stack known as the RDC layer. Currently, Contiki provides various RDC protocols that follow the asynchronous paradigm, which relies heavily on low-power probing and low-power listening (LPL) such as ContikiMAC [6], X-MAC [7] etc.

The default Contiki MAC layer is a CSMA/CA mechanism that places outgoing packets on a queue. Packets from the



(a) Contiki OS (b) OpenWSN OS (c) RIOT OS

Fig. 1: Network Stack of Contiki, OpenWSN and RIOT operating systems

queue are transmitted in order through the RDC layer. The RDC layer in turn transmits the packets through the radio link layer which are IEEE 802.15.4 compliant. The MAC layer will retransmit packets until it sees a link layer acknowledgement from the receiver. If a collision occurs, the MAC layer does a linear back-off and retransmits the packet. Outgoing packets have a configurable threshold for the maximum number of transmissions. Outgoing IPv6 packets flow from the IPv6 layer to the 6LoWPAN layer for header compression and fragmentation and moreover, the 6LoWPAN layer sends outgoing packets to the MAC layer. ContikiRPL [8] implements the RPL protocol, as specified in version 18 of the IETF RPL draft [9]. The ContikiRPL implementation separates protocol logic, message construction and parsing, and objective functions into different modules. In addition, ContikiRPL does not make forwarding decisions per packet, but sets up forwarding tables for IPv6 layer and leaves the actual packet forwarding to IPv6. Contiki uses CoAP which enables interoperability at the application layer through RESTful Web services but with a lower cost in terms of bandwidth and implementation complexity than HTTPbased REST interfaces. Unlike HTTP over TCP, CoAP uses UDP. This makes it possible to use CoAP in one-to-many and many-to-one communication patterns.

B. *OpenWSN Network Stack*

OpenWSN project's goal is to construct and provide a complete protocol stack based on open-source implementations of Internet of Things standards while being hardware independent. OpenWSN uses IEEE 802.15.4e Time Synchronized Channel Hopping (TSCH) as its Medium Access Control MAC which is an amendment to the well-known and widely used IEEE 802.15.4-2001 standard. The IEEE 802.15.4e standard presents time-slotted channel hopping that achieves ultra-high reliability through frequency agility; and low-power through tight time synchronization. IEEE 802.15.4e TSCH is located at the foundation of the protocol stack as shown in figure 1b. Synchronization accuracy directly relates to power consumption, and can vary from microseconds to milliseconds, depending on the hardware and implementation.

Due to the constraints of the IEEE 802.15.4e TSCH frames, the IPv6 header does not fit, thus the use of a mechanism to translate the addresses from IPv6 to something lighter is mandatory. OpenWSN like most WSN OS uses the IETF 6LoWPAN for this purpose as shown in figure 1b. OpenWSN implements a low-power border router (LBR), a device that sits between the mesh and an Internet connection for the purpose of a full IPv6 header which is required to support functionality on the Internet. The LBR inflates 6LoWPAN headers to normal IPv6 header on packets leaving the mesh and compacts the IPv6 headers on incoming packets. Furthermore, it is mandatory to use a protocol that will find the multi-hop path connecting the nodes in the network with a small number of destination nodes, this functionality is achieved with the use of IETF RPL [9] protocol. Finally, on the upper layers we see both TCP and UDP with HTTP and CoAP respectively, with the CoAP over UDP being the most used combination due to the constraint resources.

C. *RIOT OS Network Stack*

RIOT is a real-time, multithreading, open source operating system aiming to ease development across a wide range of IoT devices. It enables programmers to develop applications on typical IoT devices, with no learning curve (assuming prior experience with POSIX and Linux). It bridges the gap between OSs for WSNs and traditional full-fledged OS currently running on internet hosts.

RIOT OS Network Stack is shown in figure 1c. At the bottom of the stack consists of IEEE 802.15.4 complaint radio such as OpenMote-CC2538, Atmel SAM R21 etc. RIOT provides a TCP/IP network stack support for resource-constrained devices using 6LoWPAN and RPL as well as full support for IPv6, UDP, TCP and CoAP as shown in figure 1c. CoAP seeks to apply the same application transfer paradigm and basic features of HTTP to constrained networks, while maintaining a simple design and low overhead. CoAP uses UDP as transport protocol. This choice would enable CoAP to have a low impact on the limited bandwidth of the 802.15.4 wireless links. However, since UDP is an unreliable protocol, CoAP has to implement its own mechanisms in order to guarantee reliability to those applications that use it.

RIOT implements a microkernel architecture. In addition, RIOT add native support for C/C++. Advantages of the RIOT architecture this include: (i) high reliability and (ii) a developer-friendly Application Program Interface (API). The modular microkernel architecture of RIOT makes it robust against bugs in single component.

III. Radio Duty Cycle

Duty cycle is considered the most essential technique among other energy conservation technique due to its flexibility of implementation that does not require special capabilities of sensor node platforms and are suitable for any type of applications. Nodes turn their radios off for most of the time and wake up periodically to check for incoming packets. An efficient power-saving mechanism for WSN nodes thus relies on finding the better-tradeoff between minimizing the RDC while keeping networking efficiency at the highest level. This is achieved by MAC/RDC protocols. In this section we will investigate the RDC protocols in OpenWSN, Contiki and RIOT operating systems.

A. RDC in CONTIKI

In this section, we take a look at the radio duty cycle used in Contiki and we make a comparison between them using experimental results. The RDC (Radio Duty Cycle) layer is the most important layer of the stack in the Contiki OS. This is because it is responsible for the period during which the nodes are inactive and also responsible for the time that the packets must be transmitted, as well as ensuring that the node will be active when packet arrives. The main protocols at this layer are: (1) ContikiMAC (2) X-MAC (3) Low Power Probing.

1) ContikiMAC: ContikiMAC is a RDC protocol that repeatedly transmits a full data packet until it is acknowledged by the receiver. The frame destination field allows to reduce overhearing: a node that is not the destination of the frame can immediately go back to sleep. In the opposite case the receiver acknowledges the correct reception of the frame. When an ACK is received, the sender stops sending the data frame and the transmission is successful. A transmission will fail if no ACK is received after a duration equal to a wake-up interval. In this case, it is the responsibility of the above layers to schedule a retransmission. Broadcast transmissions are achieved in the same way than unicast, except that no data-ACKs are expected.

2) X-MAC: The X-MAC is a RDP protocol older than ContikiMAC and more energy intensive. It uses a preamble to achieve timing synchronization and maximum throughput for the data exchange. Moreover, X-MAC uses addressing and structure of the frames of the 802.15.4 protocol which makes it 802.15.4 compatible. X-MAC sends a stream of short-sized preambles (strobes) to wake up receivers. Nodes turn off the radio for most of the time to reduce idle listening. They wake up shortly at regular intervals to listen for strobes. When a receiving node wakes up and receives a strobe destined to it, it replies with an acknowledgment indicating that it is awake. After receiving the ACK, the sender transmits the data packet.

3) LPP: Low Power Probing (LPP) is a power-saving MAC protocol where receivers periodically send small packets, so called probes, to announce that they are awake and ready to receive a data packet. After sending a probe, the receiver keeps its radio on for a short time to listen for data packets. A node willing to send a packet turns on its radio waiting for a probe from a neighbor it wants to send to. On the reception of a probe from a potential receiver, the node sends an acknowledgment before the data packet.

4) Other RDC Protocols: There are other RDC protocols for Contiki such as CX-MAC; NullRDC. CX-MAC is a RDC protocol, based on X-MAC. The basic difference between CX-MAC and X-MAC is that the former is not as strict as latter in terms of timing, which makes CX-MAC appropriate for

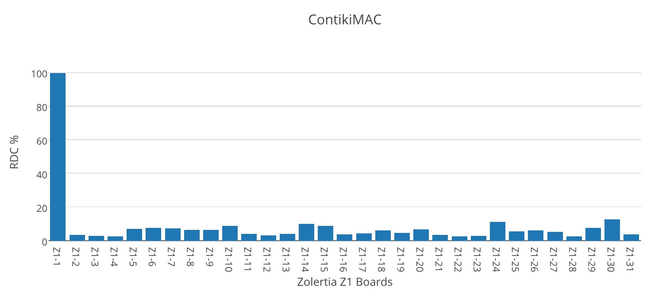


Fig. 2: Chart of ContikiMAC RDC

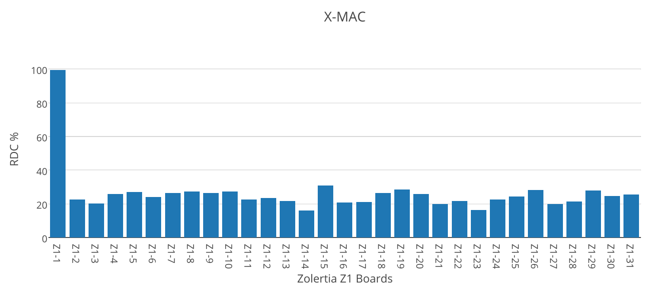


Fig. 3: Chart of X-MAC RDC

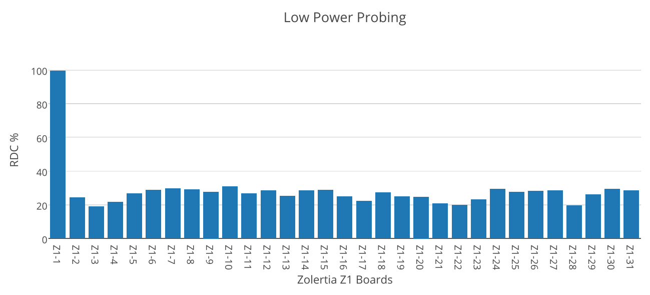


Fig. 4: Chart of LPP RDC

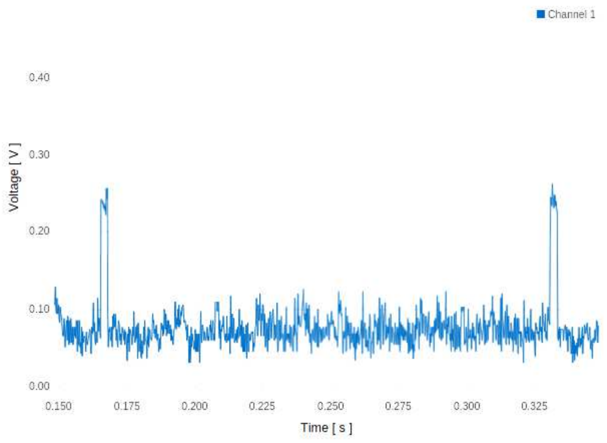


Fig. 5: Duty cycle of IEEE 802.15.4eTSCH in OpenWSN

a broader set of radio platforms. NullRDC uses the Framer functions for header creation/parsing. It does not save energy and works as a pass-through layer that only transmits a packet and returns the results of such transmission (success or collision).

Experimental Results

In this experiment, we present some numerical results showing the performance of the most famous $RDCs$ of ContikiOS namely ContikiMAC, X-MAC, LPP based on their duty cycle as shown in figure 2,3,4 respectively. In the simulation, we consider Zolertia Z1 board [10] as the hardware with a random topology laid in a WSN network. We made use of 31 nodes with node 1 $(Z1-1)$ indicating as the server and the remaining 30 nodes $(Z1-2...31)$ as clients. We made use of a firmware based on IPv6 and RPL for UDP packet transmission with a transmission interval of 1s. The duration of each simulation was 10mins. The results shows that contikiMAC provides the lowest RDC compared to X-MAC and LPP denoting a drastically reduced energy consumption state.

From the simulation results, ContikiMACs average duty-cycle is 8.73% while X-MAC provides an average duty-cycle of 26.22%. This means we have a 17.49% reduction on duty-cycle using ContikiMAC with a large impact on the energy consumption of the nodes. Finally, the Low Power Probing RDC achieved a duty-cycle of 28.35% which makes it the least energy consumption oriented RDC.

B. RDC in OpenWSN

In OpenWSN, the IEEE 802.15.4e MAC makes RDC a part of the standard. It defines three RDCs namely receiver-initiated RIT (Receiver Initiated Transmission); a sender-initiated CSL (Coordinated Sampled Listening); and a synchronous TSCH (Time-Synchronized Channel Hopping). TSCH is mostly used because of its special capabilities providing ultra-low power and reliability.

Furthermore, we carried out an experiment in order to calculate the duty-cycle of IEEE 802.15.4e TSCH in the OpenWSN using the OpenMote microcontroller with OpenBase and OpenBattery boards. In our experimental setup, we made use 9 OpenBattery boards (the maximum number of nodes that a sink can handle) and one OpenBase board. The firmware used was the out-of-the-box OpenWSN firmware and only RPL messages were exchanged during the experiment as shown using the oscilloscope in figure 5. This provided us with an average duty cycle of 2.3% that gives a remarkable low impact of energy consumption on WSN. A comparison with ContikiMAC RDC shows that the sink node in ContikiMAC has a duty-cycle of 99.63% while the sink node in OpenWSN has 3.2%.

C. RDC in RIOT

In RIOT OS, there is no MAC/RDC layer implemented on the network stack but the duty cycle mechanism is part of the IEEE 802.15.4 MAC standard. However, it still lacks high-performance MAC / RDC layer protocols. RIOT OS is suitable for implementing high performance MAC / RDC protocols, thanks to its real-time features (especially hardware timers management).

IV. ENERGY CONSUMPTION MEASUREMENTS AND RESULTS

There have been tremendous advancement in the development of different areas in WSN but regrettably the energy densities of the batteries did not follow the same trend. In order to ensure the expected lifetime in a WSN, it is important to properly define the work-flow of the nodes, evaluating and measuring their energy consumption. Such evaluation may provide feedback during application design phase, consenting to improve the overall energy efficiency [11]. The energy consumption profiling of a node is also an important stage in the deployment of a WSN, since it consents to properly configure the duty-cycle and the number of transmissions as a function of the available energy. There are several methods to estimate the energy consumption of a WSN node, including theoretical estimation, direct measurements and usage of simulation tools [12].

Theoretical estimation relies on the abstraction of the network, including the surrounding environment [13]. It can simplify the modeling procedure, but they cannot accurately represent the inherent complexity of sensor networks. The accuracy in providing a realistic model is limited in describing the environment, moreover the trade/off between latency, cost and portability essentially limits models to testing.

Direct measurements on the other hand offer the best accuracy on energy consumption measurement estimation and evaluation, and widely used. It is feasible and all the functions are set correctly, also no inaccurate presumption is made. Direct measurements can be carried out on hardware using power meters, oscilloscope or specific instrumentation under fixed conditions.

Usage of simulation tools such as AVOVRA etc. can also be used to measure the energy consumption of a WSN node. By using simulation tools, various scenarios of the real environment can be modeled. In this section, we

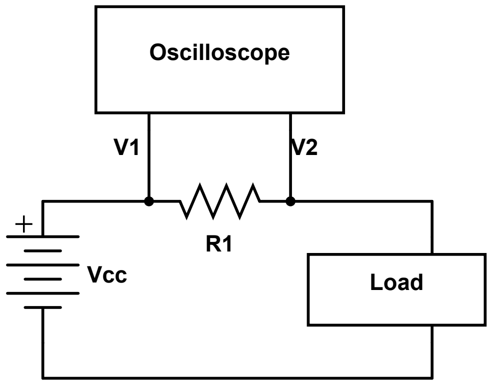


Fig. 6: Measurement Setup with an Oscilloscope

will demonstrate the power/energy consumption of a WSN node with experimental evaluation focusing on the direct measurement.

A. Experimental Setup

The measurement presented in this paper were done with an OpenMote platform which includes a CC2520 radio module and a cortex-M3 microcontroller. OpenMote-CC2538 [14] board features 4LEDS with an operational voltage of the platform to be 3.6V. Openmote platform consists of an Open-Battery which provides power and basic sensing capabilities. It is powered by a 2 AAA batteries, it also contains a three sensors: a temperature/humidity sensor (SHT21), a 3-axis accelerometer (ADXL346) and a light sensor (MAX44009). A very common series insertion of a small resistance (10Ω) is connected with the device to measure the consumed current as shown in figure 6. An oscilloscope was used to measure the current generating very accurate measurements that can be used to visualize all the steps of data transmission. This setup allows us to monitor the power consumption when different components of the board are active, and its time dependence. A key role in the reliability of such measurement is played by the oscilloscope, the tolerance on the shunt resistor value, the stability of the supply voltage, and the measurement rate, that should be compatible with the analyzed phenomena.

Since embedded systems usually operate at constant supply voltage, power consumption measurements can be carried out indirectly, by measuring and monitoring the absorbed current. Then, by measuring the voltage drop $V_2 - V_1$ across the resistor, current I can be measured indirectly, using Ohms law, as shown in the equation below. The power measured is the multiplication of the absorbed current and the constant voltage supply. Thus, the energy consumed for a specific time duration, can be easily calculated as shown in applying the following equation below.

$$I = \frac{\Delta V}{R_1} = \frac{V_2 - V_1}{R_1} \quad (1)$$

$$E = P \times t = \left(I \times V\right) \times t \quad (2)$$

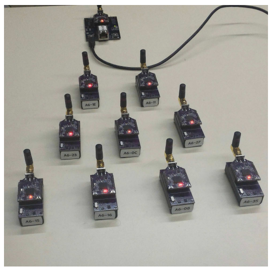


Fig. 7: A tree topology using OpenMote-CC2538

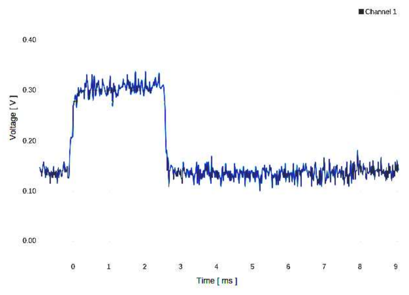
TABLE I: States of TX/RX slots in IEEE 802.15.4eTSCH

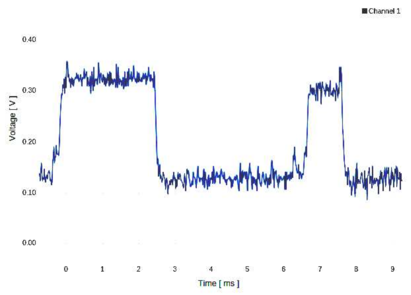
Tx	Rx
TxDataOffset	RxDataOffset
TxDataPrepare	RxDataPrepare
TxDataReady	RXDataReady
TxDataDelay	RxDataListen
TxData	RxData
RxAckOffset	TxAckOffset
RxAckPrepare	TxAckPrepare
RxAckReady	TxAckReady
RxAckListen	TxAckDelay
RxAck	TxAck
TxProc	RxProc
Sleep	Sleep

B. Results and Discussions

This section measures the power consumption of OpenMote platform main modules in various operating systems. The main objective of this section is to provide consolidated information extracted from these measurements in useful and practical manner to facilitate a modeling approach.

1) Energy Consumption in OpenWSN: Several measurements are presented in order to highlight the energy consumption of the examined board in different scenarios. At first, we determine the power consumption using OpenWSN operating system with OpenMote boards (both OpenBase and OpenBattery) as shown in figure 8. We made use of a tree-topology with 9 motes (open battery) and one OpenBase boards acting as the sink (DAGRoot) shown in figure 7. The power consumption of the 4 LEDs in OpenBase and and the sensors in OpenBattery also have their effect on the overall power consumption. Figure 8a shows an idle listening slot in which the radios wake up to listen for transmission. Then it sleeps back since there are no transmissions. In the active slots with 10ms as the slot duration, the motes transmits RPL messages to the sink and waits for an ACK packet from the sink as shown in figure 8b. Figure 8c shows the open battery motes receiving the RPL messages. The different states of the 10ms slot duration used are shown in table 1. The power consumption measurements consider the following different radio states :receive/listen, transmission under different transmission power thresholds and idle state as presented in table 2. The TX and RX state measurements confirm the respective upper and lower limits presented in the TI-CC2538 datasheet. During packet transmission, the current consumes 22mA and during reception, it consumes 21mA.


(a) idlelistening


(b) tx

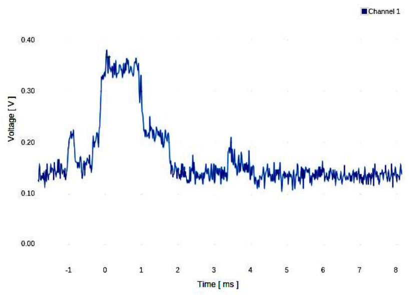

(c) rx

Fig. 8: Current drawn of OpenMote-CC2538 running the OpenWSN protocol stack

2) Energy Consumption in Contiki: We measure the energy consumption of the OpenMote with OpenBattery using Contiki operating system. In this experiment we made use of a simple UDP network topology with 9 clients and one sink. We made use of the 6LoWPAN netstack combined with ContikiMAC and CSMA. ContikiMAC is an asynchronous protocol and the duty cycle of the sink must be around 100% because the sinks radio is always on. Figure 9 shows when the motes sends RPL messages to the sink. The Clear Channel Assessments (CCAs) are shown in the figure is to determine if the channel is ready and able to receive data, so that the transmitter may start sending. When the motes is about to transmit a packet, it performs a CCA in two consecutive slots. If the channel is found to be idle in both these slots, the node goes ahead with its transmission. Otherwise, the node attempts CCA again after a random backoff, which it repeats a certain number of times before reporting an access failure to the upper layer. The results for the power consumption in the sink is 0.14572W while the average power consumption on the normal motes is 0.10404W.

3) Energy Consumption in RIOT: We measured the energy consumption of the sink using the Border Router. In this experiment we measured the energy of a simple UDP network topology with 9 clients and one sink. The measurement showed that the radio of the sink is always on with a 100% duty-cycle. The energy consumption of the RIOT Border Router is 0.1534W while the average energy consumption of the motes is 0.1092W.

The most energy efficient WSN operating system using the OpenMote platform is OpenWSN because of IEEE 802.15.4e TSCH implementation at its MAC layer as seen from the experiment. The radio in both OpenBase and OpenBattery is switched on only when it is necessary, which makes the average RDC of OpenWSN to be 2.5% and translates to 0.0882W and 0.09252W in the idle listening state for both OpenBase and OpenBattery respectively. Furthermore, the power consumption in OpenWSN using OpenMote is low as 0.10296W in the RX and 0.10584W in the TX. On the other hand using Contiki with ContikiMAC yields an average RDC with power consumption of 0.14572W for OpenBase and 0.10404W for OpenBattery. Moreover, the radio of the sink is always on while the rest of the nodes switched their radios on only to transmit packets for a finite time. With RIOT OS on OpenMote, the power consumption achieved in OpenBattery is 0.1092W making it the least energy efficient among others described.

TABLE II: Average Power Consumption of the motes

OpenWSN	OpenBase	OpenBattery
IdleListen	0.0882W	0.09252W
Rx	0.1029W	0.1029W
Tx	0.10584W	0.10584W

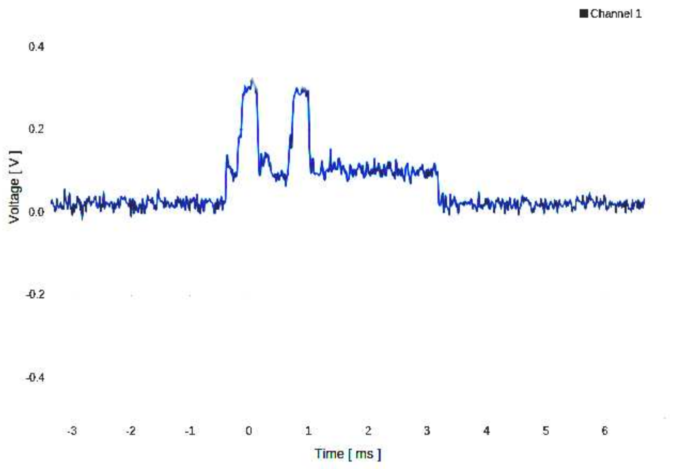


Fig. 9: Current drawn of OpenMote-CC2538 running the Contiki OS stack

V. Conclusion

We provided a critical analysis of the radio duty cycle of each with experimental evaluation. The duty cycle can be designed appropriately by inspecting the network topology and settings nodes close to the sink node to have more activity. The RDC have serious impact on the power consumption of the WSN nodes. Moreover, since many factors influences the energy consumption in wireless sensor networks, we presented the design and implementation of an efficient measuring setup based on a well-known WSN platform, able to provide accurate measurements concerning the current demands of the mote. A sufficient number of experiments were carried out aiming to evaluate the Wireless Sensor Network power consumption under specific scenarios using various operating systems. It was observed that the IEEE 802.15.4e TSCH has great impact on the power consumption on the motes in relation to other radio duty cycles.

VI. Acknowledgment

This work has been partially supported by the PRA 2016 research project 5GIOTTO funded by the University of Pisa.

References

[1] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang, and R. Lemmens, "New generation sensor web enablement," *Sensors*, vol. 11, no. 3, pp. 2652–2699, 2011.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] A. Dunkels, B. Grönvall, 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.

[4] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. Schmidt, "Riot os: Towards an os for the internet of things," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, April 2013, pp. 79–80.

[5] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "Openwsn: a standards-based low-power wireless development environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[6] A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne, "The announcement layer: Beacon coordination for the sensornet stack," in *Wireless sensor networks*. Springer, 2011, pp. 211–226.

[7] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "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*. ACM, 2006, pp. 307–320.

[8] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-power wireless ipv6 routing with contikirpl," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010, pp. 406–407.

[9] T. Winter, "Rpl: Ipv6 routing protocol for low-power and lossy networks," 2012.

[10] W. Zolertia, "platform, z1 datasheet."

[11] A. Prayati, C. Antonopoulos, T. Stoyanova, C. Koulamas, and G. Papadopoulos, "A modeling approach on the telosb wsn platform power consumption," *Journal of Systems and Software*, vol. 83, no. 8, pp. 1355–1363, 2010.

[12] A. Moschitta and I. Neri, "Power consumption assessment in wireless sensor networks," *ICT-Energy-Concepts Towards Zero-Power Information and Communication Technology*, 2014.

[13] J. Alonso, S. Gómez, M. Alejandrez, M. Gil, and N. Navarro, "Experimental measurements of the power consumption for wireless sensor networks," *Computer Architecture Department Universitat Politècnica de Catalunya June*, vol. 26, 2006.

[14] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "Openmote: Open-source prototyping platform for the industrial iot," in *Ad Hoc Networks*. Springer, 2015, pp. 211–222.