Sustainable Medium Access Control: Implementation and Evaluation of ODMAC

Xenofon Fafoutis, Alessio Di Mauro, Nicola Dragoni
Department of Applied Mathematics and Computer Science
Technical University of Denmark, Denmark
{xefa,adma,ndra}@dtu.dk

***Abstract*—Harvesting small-scale ambient energy constitutes a promising source of power for wireless embedded devices. Due to the unpredictable nature of the harvested energy, adaptive radio duty cycling can lead to a long-term sustainable operation. In energy constrained conditions, very low duty cycles are vital to guarantee the sustainability of the system; whereas, in the opposite case, the system should use the energy surplus to increase the application performance. In this paper, we implement and evaluate On-Demand MAC (ODMAC), the first receiver-initiated MAC protocol specifically designed for energy harvesting applications. In particular, we provide a basic yet fully operational implementation of ODMAC for the Texas Instruments' MSP430 microprocessor family. Furthermore, we verify the theoretical results of our previous work by achieving sustainable operation of an energy harvesting node in various cases of energy input using a real test-bed.**

***Index Terms*—Medium Access Control, Energy Harvesting, Wireless Sensor Networks**

I. Introduction

Energy Harvesting (EH) technologies have led to the possibility of powering wireless embedded devices by small-scale energy that is harvested from the surrounding environment. Depending on the specific application, several energy sources can be used, such as solar power and wind power in outdoor deployments or heat from radiators in indoor situations. Traditionally, wireless embedded devices are powered by batteries. This power source constitutes a limitation of the operational lifetime of the devices. Energy harvesting can potentially produce and provide the system with an infinite amount of energy. Therefore, the continuous operation of the system is solely limited by hardware or software failures. Furthermore, energy harvesting constitutes an environmental friendly means of providing power to embedded devices, as it uses renewable energy and reduces battery wastes.

Energy harvesting introduces several challenges in wireless sensing systems. Among these, it is possible to find the MAC (Medium Access Control) layer. The unpredictable, ever-changing and small-scale nature of the energy input makes efficient radio duty cycling the only means to achieve sustainable operation. The duty cycles need to adapt to significantly different levels of energy input. In particular, in energy constrained conditions, the MAC protocol must allow very low duty cycles in order to guarantee the longer-term sustainability of the system. On the other hand, when energy is abundant, the same protocol has to efficiently use the energy surplus to increase the performance of the application.

Radio duty cycling introduces the problem of finding a moment in time where both the sender and the receiver are active so that a link can be established. Traditionally, MAC protocols for Wireless Sensor Networks (WSN) use either synchronous or asynchronous approaches to handle this challenge. MAC protocols based on synchronization, like S-MAC [1], T-MAC [2] and DSMAC [3], require a notion of globally synchronous clock. The coupling of nodes' activity periods hinders their ability to have a fully independent duty cycle preventing the nodes from adapting to the energy they can harvest. This crucial requirement makes an asynchronous approach to this problem mandatory. The establishment of a link between two nodes can be initiated either by the sender, via preamble sampling (e.g. WiseMAC [4], B-MAC [5], X-MAC [6], or by the receiver, via beaconing (e.g. RICER [7], RI-MAC [8]). Within the asynchronous approach, the receiver-initiated paradigm has proven to be more energy efficient when compared to the sender-initiated scheme [8] [9].

On-Demand MAC (ODMAC), is the first receiver-initiated scheme specifically designed for energy harvesting applications [10]. It extends the receiver-initiated paradigm towards the Energy Neutral Operation (ENO) [11] concept. Unlike other protocols, ODMAC employs dynamic duty cycling, specifically for the purpose of regulating the power consumption. Furthermore, each node can be made sustainable while the excess of energy, should there be any, can be used for improving application performance metrics, such as throughput, delay or security [12]. ODMAC has been shown, through analysis and simulation, to efficiently support the system requirements of sustainability and application performance. The main goal of this work is to extend the evaluation of the previously proposed ODMAC [10] in a real test-bed by means of implementation. In particular, the contribution of this paper can be summarized as follows. First, we provide a basic yet fully operational implementation of ODMAC for the Texas Instruments' MSP430 microprocessor family [13]. Furthermore, we focus on a specific case study and we verify the theoretical results of our previous work by achieving the sustainable operation

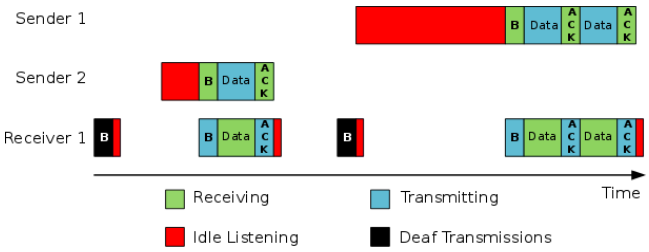

Fig. 1. ODMAC communication scheme

of the whole system in various cases of energy input using a real test-bed.

The remainder of the paper is structured as follows. In Section II, we provide a detailed presentation of ODMAC, which is followed by the details of our implementation in Section III. In Section IV, we experimentally evaluate the protocol in various cases of input energy. Lastly, Section V concludes the paper.

II. On-Demand MAC (ODMAC)

The receiver-initiated paradigm constitutes the foundation of all the receiver-initiated asynchronous protocols. According to the paradigm, a node willing to receive data, wakes up periodically and checks for incoming transmissions. To do so, a Clear Channel Assessment (CCA) is performed immediately after waking up, and a special message called *beacon* is broadcasted if the channel is free. Then, the node performs a Binary Exponential Backoff (BEB) and transmits the beacon. After the beacon has been transmitted, the receiver continues to listen to the channel for a short predetermined period of time. Meanwhile, whenever a node with data ready to be sent enters the active state, it listens silently for a beacon from the intended receiver. Once the beacon is received, the sender transmits its data packet, and waits for another beacon which acknowledges (ACK) the reception of the data. Conversely, if there is no incoming data after transmitting the beacon, the receiver enters a sleep state. At this point both the sender and receiver resume their cycles normally.

The receiver-initiated paradigm significantly reduces the amount of time the channel is occupied, allowing more contending nodes to communicate with each other, thus increasing the capacity and throughput of the network. It is also more efficient in detecting collisions, because access to the channel is mainly controlled by the receiver. Since receivers only wait a short period of time for incoming data, after beacon transmission, idle listening is greatly reduced.

ODMAC builds upon the receiver-initiated paradigm, as shown in Fig. 1. To adapt to the ever-changing unpredictable nature of the energy input, nodes dynamically adjust their duty cycle in a completely independent and distributed manner. Nodes in the network have the double role of *receivers* for forwarding tasks and *senders* for measuring tasks. ODMAC decouples the duty cycles of these two jobs

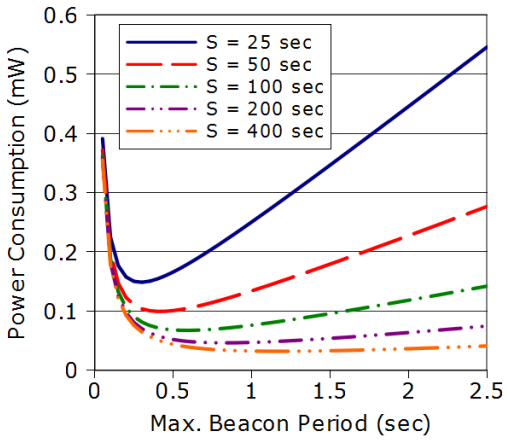

Fig. 2. Average node consumption for various sensing periods (S) [12]

within a single node. Hence, a node has a *beacon duty cycle* and a *sensing duty cycle*. The beacon duty cycle controls the trade-off between energy consumption and end-to-end delay, while the sensing duty cycle controls the trade-off between energy consumption and throughput. Thus, ODMAC grants the network administrator the ability to establish the trade-off depending on the particular application.

ODMAC uses an adaptive duty cycle mechanism based on the ENO principle [11]. According to the ENO principle, a node is sustainable if, over a time period that its energy buffers can support, the energy consumed is less than or equal to the energy harvested. All nodes in the network dynamically adjust the beacon and sensing duty cycle, in order to achieve and maintain an ENO-Max state, which is defined as an ENO state with maximum performance. This means that when the node is consuming more energy than is harvested, the duty cycles are decreased to reduce the energy consumption. In the same manner, when the energy consumed is less than the energy harvested, the duty cycles are increased.

Increasing the beacon period has two contrasting effects in the power consumption. On one side, the power consumption connected to the beaconing process is decreased. On the other side, the nodes waiting for beacons in order to perform data transmission have to spend more time in idle listening, wasting energy. Reducing the beacon period produces opposite results. Previous analytic results in random multi-hop topologies, suggest that the system has an operating state where the average power consumption is minimized, as shown in Fig. 2 (see [12] for the details of the analysis). Thus, the system has the following operating alternatives. In case of delay-sensitive applications, it can trade power for shorter delays by adapting the beaconing period between zero and the local minimum. Similarly, in cases of applications where throughput is the main priority, the system can trade power for throughput by adapting the sensing period while maintaining the beaconing period at the local minimum. Alternatively, the system can operate

at the minimum power consumption level and expend the energy elsewhere (e.g. security-sensitive applications).

For scenarios with multiple duty-cycling receivers, ODMAC defines a forwarding policy for anycast routing support, named *Opportunistic Forwarding*. Instead of waiting for a specific receiver to wake up, a sender opportunistically forwards data to any eligible receiver, based on the beacon obtained first. This mechanism requires a routing protocol that assigns each sender a list of approved receivers. The more the energy a receiver harvests, the higher the probability of receiving beacons from it. Thus, this policy creates a more robust network, that is adaptive to changes in energy, by maintaining the load balanced. Furthermore, the idle listening time of senders, and therefore their energy consumption, is significantly reduced [9].

III. Implementation of ODMAC

Our design is based upon the holistic approach, which claims that the whole system should be designed and function as a whole, rather than being organized in layers. This approach sacrifices the versatility of the system toward the efficient use of resources, as all parts of the system, from the hardware to the firmware (i.e. protocols and system services), need to be specifically designed for the desired application. As a result, the implementation of ODMAC constitutes integral part of a complete firmware.

The protocol was implemented on Texas Instruments' eZ430-rf2500 sensor nodes [14]. The nodes consist of an MSP430 microcontroller (MCU) and a CC2500 radio, operating in the 2.4 GHz band. In addition to batteries, the nodes can be powered by external energy harvesting boards. In particular, we use Cymbet's CBC-EVAL-10 [15] solar energy harvester board and CBC-EVAL-09 [16] general energy harvester board, that can harvest energy from various sources. The boards store the harvested energy into embedded batteries (100 μAh capacity). The boards can also accommodate external rechargeable batteries for scenarios that require larger energy buffers.

Duty cycles are implemented through wake-up interrupts using the timer of the MCU. A time quantum is defined. It controls the sleeping time between two subsequent wake-up events. On top of that, the two independent duty cycles for the sensing and the forwarding tasks are implemented as multiples of the basic time quantum. In each wake-up interrupt the MCU checks if it's time for one of the two tasks, sets up the next wake-up interrupt and goes to the sleeping state. While in the sleep state, the MCU is configured to Low Power Mode 3 (LPM3), in which only the auxiliary low-frequency clock, used to schedule the interrupts, is active. In LPM3, MSP430 consumes less than 1 μA at 1 MHz [17]. Additionally, the time quantum is periodically adjusted, by adding a uniformly random number of cycles in $[-2^{r-1}, 2^{r-1}]$ to the defined value. This randomization prevents unfortunate synchronizations and decreases the collisions by enforcing random channel access between different nodes. Even though the period

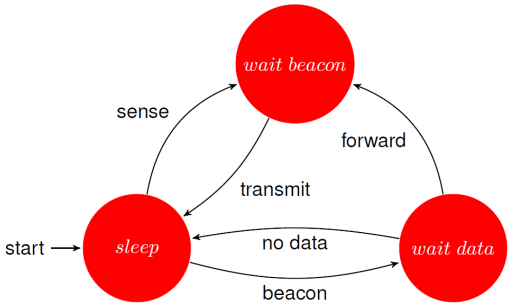

Fig. 3. ODMAC as a Finite State Machine

of the time quantum randomization can be individually configured, it is currently set to the period of the sensing tasks. The level of the randomization, r, can be configured in accordance to the desired behavior.

The heart of ODMAC is implemented as a finite state machine (FSM), Fig. 3. Its functionality is mainly based upon two routines, namely *Send* and *Receive*. Unless one of these two handler is invoked, ODMAC is in sleeping state and the radio is turned off. The *Send* routine generates and formats a packet around the payload (i.e. the result of a sensing operation). When the packet is ready, the radio is switched on into listening mode and the state machine awaits for an interrupt signaling the reception of an appropriate beacon. Should this happen, ODMAC continues its execution and the data packet is transmitted. At the end of a packet transmission, the radio is switched back off.

Different packet types might be received when waiting for a suitable beacon. While non beacon packets are simply discarded, potentially appropriate beacons are evaluated. The decision of whether or not a specific beacon is appropriate, is taken by a routine that implements the *routing functionalities* and that will be described further on. The *Receive* handler is invoked during the forwarding duty cycle. In particular, it generates a beacon packet and transmits it to the node's neighborhood. At this point, the radio is switched into listening mode and the protocol awaits for a data packet for a defined amount of time. If no incoming data is received during this period, the radio is set back to sleep mode and the routine ends. On the other hand, upon receiving a data packet, the information contained is extracted and the radio set back to sleep mode. In order to forward the newly received packet toward the sink, a new invocation of *Send* is performed.

To accompany and improve ODMAC, a security suite inspired by TinySec [18] has been designed and implemented. It provides four modes: *No security, Authentication, Encryption, Both* allowing to choose among them on a per-message basis. Both confidentiality and integrity are provided through the same encryption primitive. The algorithm of choice is *Skipjack* [19], but other implementations are underway (e.g. AES, PRESENT [20]). Encryption

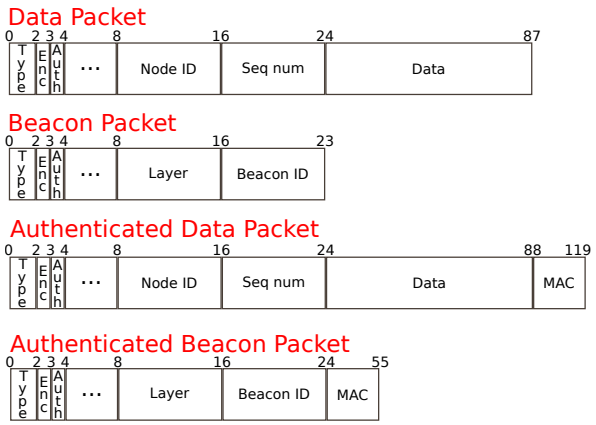

Fig. 4. Packet Formats (field size in *bits*).

is always performed by using secure modes of operation, specifically *Cipher-block Chaining* with *Cyphertext Stealing* to keep the message size unmodified. Authentication of beacons guarantees that these messages cannot be forged. The suite works at the same layer of ODMAC, therefore *Message Authentication Codes* and encryptions are verified and reconstructed at each hop. This requires more CPU intensive work, but allows for forged or malformed packet to be identified and discarded right away, thus saving transmissions of useless data and assuring greater energy savings. Key managements schemes and adaptive security that take into account the energy harvesting rate of the nodes are considered future extensions of the security layer.

Additionally, we implemented a simple and fully distributed routing protocol that exploits the benefits of *opportunistic forwarding* by selecting multiple appropriate forwarders. Specifically, we define $layer(u)$ as the distance of node u form the sink, expressed in number of hops. The sink is initialized at layer 0. All nodes advertise their layer through their beacons and nodes update their layer upon beacon reception. Let B be the set of layers received by node u then $layer(u) := \min(B) + 1$. Additionally, layers are reset ($layer(u) = \infty$) if no beacon is received after a predefined amount of time. *Appropriate* beacons (i.e. beacons useful for transmission of data) are considered those advertising a layer lower than the one of the receiving node, thus leading towards the sink. More formally, a beacon b is *appropriate* for node $u \iff layer(b) < layer(u)$. By using the beacons to distribute information required for routing decisions, we avoid transmitting extra control packets and save energy. Additional routing metrics that account for energy harvesting and application performance are planned to be incorporated in the routing layer.

Fig. 4 depicts the packet formats of the implementation. The packet headers contain fields for all the services implemented in the firmware beyond the MAC protocol. In particular two basic packets are defined, namely the beacon and the data packet. The first byte is common in all packets. The first two bits in the header are used to determine the packet type. The two next bits are used to state the security options of the specific packet. Hence, the receiver of the packet can determine if the packet needs to be decrypted and whether or not it is authenticated. In the case of authenticated packets, a 4-byte footer contains the Message Authentication Code (MAC[1]). The next 4 bits are reserved for future extensions. In case of a beacon packet, the header also includes the layer of the node and a sequence number, named *Beacon ID*. In case of a data packet, the header includes the ID of the node and a sequence number of the packet used to track lost packets. Finally, the payload of the data packet follows.

IV. Experimental Evaluation

In our previous work, we showed through analysis [9] and simulations [10] that, by using ODMAC, a node can adapt its activity in a completely distributed manner and independently from other nodes. As a result, the operation of the node is sustainable, adapting to different levels of input energy. Furthermore, by decoupling the duty cycle of sensing and forwarding tasks, any energy surplus can be used to improve different performance metrics, such as throughput or delay, and can be configured in accordance to the running application. In this section, we attempt to verify this finding. In particular, we focus on the case study of applications that prioritize the throughput. Monitoring an area for long-term statistical off-line analysis constitutes an example of a delay-tolerant application, in which the amount of measurements is the first priority. Thus, we expose a node to different levels of energy input and we demonstrate how the node can find a sustainable operation state while the energy is used towards the selected performance metric, i.e. throughput.

Even though ODMAC is a multi-hop MAC protocol, in this work we focus on a single transmitting node, u, which is part of a single link to a receiver node. Hence, the routing layer is not used. From the perspective of u, the activity of the receiver is unknown. We consider two identical receivers, one with high and one with low Duty Cycle (DC). We programmed the sleep time between the transmission of two beacons at 400 cycles for the *High DC Receiver* and 800 cycles for the *Low DC Receiver*, which corresponds to approximately 33 ms and 66 ms respectively. Additionally, we turned off the forwarding duty cycles of u, focusing entirely on the sensing duty cycles. Specifically, node u periodically interrupts its sleeping to execute an active period, which consists of the following actions: a) sense the MCU temperature using the internal temperature sensor, b) generate a packet, c) encrypt and authenticate the packet, d) wait for a beacon from the receiver, e) transmit the packet. The consumption of a typical activity period is shown in Fig. 5. Specifically, the figure shows the voltage of a 10 Ω shunt resistor, connected between the load and

[1]MAC in Fig. 4 stands for Message Authentication Code. Not to be confused with Medium Access Control.

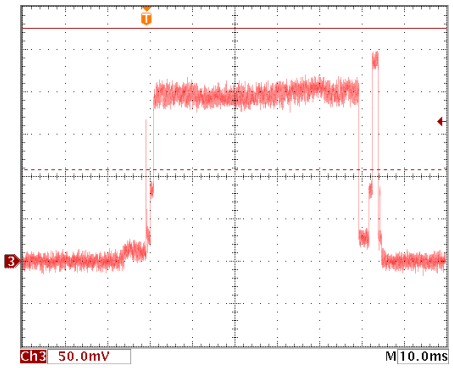

Fig. 5. Consumption of a typical active period. The current drain is obtained by dividing the shown voltage by the shunt resistor's value (10 Ω).

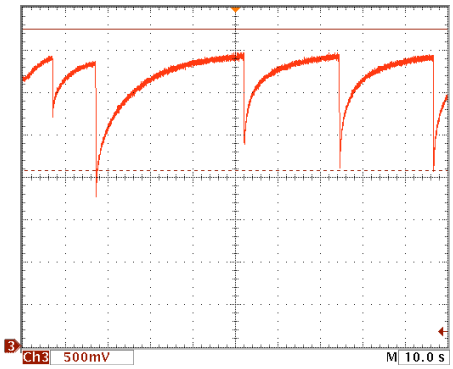

Fig. 6. The output capacitor voltage demonstrates a typical series of activity periods.

the power source. In the figure, one can clearly notice the time the node is listening for a beacon, which follows some initial MCU activity, that that includes using the node's temperature sensor, the ADC and an LED. After the beacon reception, it is possible to see the consumption spike related to the packet transmission. As expected, the main source of power consumption comes from the time the radio spends in listening mode, waiting for a beacon. Given this specific configuration and network topology, the average duration of an active period was found to be $43\ ms$ with a standard deviation of $11\ ms$ in the case of the *High DC Receiver* and $61\ ms$ with a standard deviation of $23\ ms$ in the case of the *Low DC Receiver*.

Node u is powered by a photo-voltaic panel connected to a CBC-EVAL-09 energy harvester board. Due to the specific hardware used, the duty cycle implementation, described in Section III, had to be slightly adapted. While the node still has a hard-coded value for the wake-up interrupts, not all of these are actually used to transmit a packet. The harvester in use is designed around factory specifications that support relatively short high-consumption activity periods (e.g. whenever the radio is on). The energy accumulated in the solid state batteries of the board is used to charge the following stage, composed of a $1000\ \mu$F capacitor that is then used as the final energy output. Such component is designed to handle long drains of low current, but short pulse current drain would fully deplete its charge, without giving time to the batteries to recharge it. According to [21], the embedded solid state batteries cannot charge the capacitor in less than approximately 10 seconds. Depleting the capacitor resets the node and triggers a protection mechanism, that disconnects the load until the capacitor is fully charged. According to our measurements, the capacitor can support activity periods with duration in the order of tens of ms (up to $\approx 150\ ms$).

Considering that the possibility of replacing this critical component would only solve the problem for this specific application, we decided to devise a solution that could be applicable in different scenarios. The idea is to measure the voltage across the capacitor and, after opportune conditioning, to feed it back to the node through an A/D conversion channel. Being able to know the actual charge state of the capacitor allowed the application to use this value to decide whether or not the energy available would be enough to enter an activity period. We empirically found such threshold and validated it by matching the energy available in the capacitor against the energy required by the node to send an packet in the worst case scenario. Whenever a wake-up event arises, the node can simply read the actual capacitor voltage and compare it to the aforementioned threshold, sending a packet only if the value is high enough.

As a result, the duty cycle of the node is a multiple of the sensing duty cycle, based on the state of the capacitor. Specifically, we set the wake-up interrupts every 12048 cycles ($\approx 1\ s$). Given the minimum time required for the capacitor to be charged [21], we check the state of the capacitor every 10 wake-up events ($\approx 10\ s$) and transmit when the voltage across the capacitor is above $3.3\ V$. In turns, this solution allows us to dynamically adapt the duty cycle (and therefore the amount of packets sent) according to the amount of energy harvested, making the application energy aware. Fig. 6 depicts the voltage of the capacitor in a succession of packet transmission. Observe how the energy required for different transmissions varies with respect to the duration of the listening period, while the time for the capacitor to recover changes accordingly.

In this setting, we conducted the following experiment. We exposed the board to different levels of input power, by adjusting the distance between the light source and the photo-voltaic panels, and we measured the amount of packets that the node managed to successfully transmit in 30 minutes. The input power is estimated using the voltage measured across the photo-voltaic panel and the current measured through a $10\ \Omega$ shunt resistor. Fig. 7 shows the results of several experiments. All experiments were initiated after the depletion of all the stored energy. Given the fact that the capacitor can not store enough energy for more than very few transmissions, the 30-minute continuous operation proves the sustainability of the node.

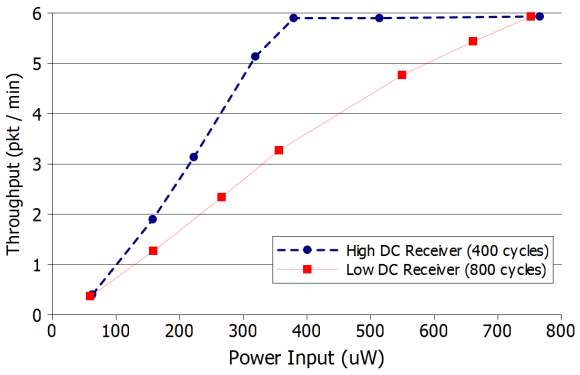

Fig. 7. Sustainable performance at different levels of input power.

Furthermore, the excess of harvested energy is used to improve the throughput of the application. As expected, the throughput increases linearly with the amount of available energy, while it is capped by the maximum throughput supported by the energy harvesting board, i.e. 1 transmission every 10 seconds. The difference in throughput, in the cases of the high and low duty cycle receivers, shows how u was able to adapt to different environmental conditions in terms of energy consumption.

V. Conclusion and Future Work

We presented a basic implementation of ODMAC [10], a MAC protocol that is based on the receiver-initiated paradigm and is aiming to support completely independent duty cycles, so that the nodes can adjust to the energy that they can harvest. Furthermore, we present experiments that show how an ODMAC-compliant transmitter is able to independently choose its duty cycle to find a sustainable state of operation and use the available energy to promote throughput, which is the selected performance metric in our case study. The implementation difficulties we encountered suggest that hardware can constitute a major limitation on implementing features at firmware level. As a result, we highlight the importance of designing and optimizing the hardware and the firmware together for a target application.

As next steps, we plan to extend the implementation of ODMAC with active collision avoidance mechanisms and evaluate it in scenarios with many nodes. Furthermore, we plan to enrich the security extensions of the protocol with additional encryption algorithms and dynamic key distribution mechanisms. Apart from applications that prioritize throughput, we also plan to consider applications that prioritize other performance metrics, such as delay and security, aiming to verify our previous theoretical work and show how the system is able to operate in a sustainable manner, while promoting different performance priorities.

References

[1] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings on the 21st Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM.*, vol. 3. IEEE, 2002, pp. 1567–1576.

[2] T. V. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, no. 5-7. ACM, 2003, pp. 171–180.

[3] P. Lin, C. Qiao, and X. Wang, "Medium Access Control With A Dynamic Duty Cycle For Sensor Networks," in *Wireless Communications and Networking Conference. WCNC.*, vol. 3, 2004, pp. 1534–1539.

[4] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. L. Roux, "Poster abstract: WiseMAC, an Ultra Low Power MAC Protocol for the WiseNET Wireless Sensor Network," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 302–303.

[5] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 95–107.

[6] 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.

[7] E.-Y. A. Lin, J. M. Rabaey, and A. Wolisz, "Power-Efficient Rendezvous Schemes for Dense Wireless Sensor Networks," in *International Conference on Communications*, vol. 7. IEEE, 2004, pp. 3769–3776.

[8] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 1–14.

[9] X. Fafoutis and N. Dragoni, "Analytical Comparison of MAC Schemes for Energy Harvesting-Wireless Sensor Networks," in *The International Workshop on Algorithms and Concepts for Networked Sensing Systems Powered by Energy Harvesters (EnHaNSS'12)*, 2012.

[10] ——, "ODMAC: An On-Demand MAC Protocol for Energy Harvesting - Wireless Sensor Networks," in *Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. ACM, 2011, pp. 49–56.

[11] A. Kansal, J. Hsu, S. Zahedi, , and M. B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 32, 2007.

[12] X. Fafoutis and N. Dragoni, "Adaptive media access control for energy harvesting - wireless sensor networks," in *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, june 2012, pp. 1–4.

[13] Texas Instruments, "MSP430 Microcontroller." [Online]. Available: www.ti.com/msp430

[14] ——, "eZ430-RF2500 Development Tool," SLAU227E, 2009. [Online]. Available: http://www.ti.com/lit/ug/slau227e/slau227e.pdf

[15] Cymbet Corporation, "EnerChip CC Energy Harvester Evaluation Kit," CBC-EVAL-10, DS-72-20 Rev A, 2011. [Online]. Available: http://www.cymbet.com/pdfs/DS-72-20.pdf

[16] ——, "EnerChip EP Universal Energy Harvester Eval Kit," CBC-EVAL-09, DS-72-13 Rev E, 2012. [Online]. Available: http://www.cymbet.com/pdfs/DS-72-13.pdf

[17] Texas Instruments, "MSP430x2xx Family User's Guide," SLAU144E, 2008.

[18] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 162–175.

[19] "Skipjack and kea algorithm specifications," Tech. Rep., May 1998.

[20] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in *Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '07, 2007, pp. 450–466.

[21] Cymbet Corporation, "Using the EnerChip in Pulse Current Applications," AN-1025, 2012. [Online]. Available: http://www.cymbet.com/pdfs/AN-1025.pdf