Implementation of IEEE 802.15.4 Unslotted CSMA/CA Protocol on Contiki OS

Hamadoun Tall, Gérard Chalhoub, Michel Misson
Clermont-Université
LIMOS UMR 6158 CNRS
Aubière, France

***Abstract*— IEEE 802.15.4 is a standard designed for low rate wireless personal area network. This standard uses the Carrier Sensing Multiple Access/Collision Avoidance (CSMA/CA) algorithm to manage the medium access process. CSMA/CA is a random wireless access algorithm that allows each node with equal probability to access the channel. Having a compliant implementation of CSMA/CA algorithm which follows the IEEE 802.15.4 specifications is very important in order to obtain realistic results. Some of the well-known Wireless Sensor Networks (WSNs) operating systems such as Contiki OS do not provide a compliant version. In this paper, we present an implementation of the IEEE 802.15.4 unslotted CSMA/CA protocol on Contiki OS. Simulation results using Contiki OS Cooja simulator showed that our implementation is compliant and achieves a better packet delivery rate and better throughput compared to the provided CSMA/CA version on Contiki OS.**

Keywords—IEEE 802.15.4, CSMA/CA, Contiki, performance evaluation, throughput.

I. INTRODUCTION

IEEE 802.15.4 [1] is a short range wireless communication standard that offers low complexity, low data rate transmission and low energy consumption. It is widely used in the wireless sensor networks applications and it provides a theoretical throughput of 250 Kbps for resource constraints and low cost wireless sensor nodes.

The IEEE 802.15.4 standard allows two types of channel access mechanisms: beacon enabled and non-beacon enabled. The latter case uses unslotted carrier sense multiple access with collision avoidance (CSMA/CA), whereas the former uses a slotted CSMA/CA algorithm with a super frame structure. CSMA/CA makes each node use a randomized waiting time before trying to access the communication medium. The use of a random mechanism helps to reduce collisions and interferences in the network. The CSMA/CA of IEEE 802.15.4 is very widely used in MAC protocols that are based on carrier sensing. The ZigBee standard MAC algorithm is also based on it [2]. Contiki [3] is one of the most famous low-power operating systems used for simulations and testbeds of wireless sensor network protocols. Nevertheless, Contiki does not provide a compliant implementation of this CSMA/CA algorithm [10].

The goal of this work is to provide an implementation evaluation of the IEEE 802.15.4 unslotted CSMA/CA standard version on Contiki 2.7. This implementation is based on the Rime [12] networking protocol stack of Contiki OS.

The remainder of this paper is organized as follow. In section II, we provide an overview of different implementations of the CSMA/CA on different simulators and operating systems. In section III, we describe in details the CSMA/CA algorithm and we present our modifications to the Contiki 2.7 CSMA/CA implementation. Simulation results are presented in section IV, and the paper is concluded in section V.

II. RELATED WORK

In [4] and [5], authors present a throughput analysis of the IEEE 802.15.4 CSMA/CA algorithm using OPNET simulator. A model for the IEEE 802.15.4 GTS mechanism is implemented. The work of these two papers address the maximum throughput of IEEE 802.15.4 channel but the limits imposed by the capacities of WSNs operating systems are ignored. Hence, the reported maximum channel capacity may not be an accurate estimation from an application perspective. An analytical throughput evaluation of the IEEE 802.15.4 slotted CSMA/CA algorithm is presented in [9]. Analytical models typically require simplifying assumptions to produce results. However, in real WSNs, these assumptions may not be true in many cases [8]. To support analytical results, simulations or experimentations are needed to validate the obtained results.

In [16], a model of discrete Markov chain that can dynamically represent different network loads is proposed to evaluate the IEEE 802.15.4 slotted CSMA/CA algorithm. By computing the steady-state distribution probability of the Markov chain, authors present an evaluation formula for throughput, energy consumption, and access latency. They further analyse the parameters that influence performance like packet arrival rate, initial backoff exponent and maximum number of backoffs. To validate the proposed model, NS2 simulator is used to evaluate the performance of the 802.15.4 CSMA/CA algorithm under different scenarios.

In [17], authors provide a performance evaluation of the CSMA/CA algorithm based on the NS2 simulator in several scenarios. They also conducted a performance comparison of the CSMA/CA between IEEE 802.15.4 and IEEE 802.11.

In [18], using OMNeT++ simulator, authors present the results of performance evaluation of four scheduling algorithms: Random Select (RS), Destination Priority Queueing (DPQ), Longest Queue First (LQF), and Shortest Packet First (SPF) designed for the unslotted CSMA/CA algorithm to address the issues of fairness and bandwidth efficiency. From the simulation results, they showed that under uniform traffic conditions, the LQF with buffer size of

13 and 78 bytes has the best performance in terms of fairness, while for throughput and packet delay, the DPQ with the maximum buffer size of 1538 bytes gives the best result.

Analytical studies and evaluation by simulation often fail to take into account the architecture of the operating system and the stack overhead on the network protocols. On the other hand, simulators such as Cooja [11] and TOSSIM [21] emulate the operation system constraint (Contiki and TinyOS [14], respectively) during the execution of the protocols.

In [13], authors present TKN15.4, an IEEE 802.15.4 MAC layer implementation for TinyOS 2. In this implementation, the MAC layer specifies the initial CSMA/CA parameters but the algorithm itself is implemented and executed in the radio driver. The radio driver is also responsible for performing any random backoffs and for the transmission of acknowledgements. According to the authors, on a typical mote platform, some requirements cannot be met by a platform independent MAC protocol [15], rather they should be pushed from the MAC layer to the Physical layer, ideally to the hardware level. Consequently they argue that the CSMA/CA algorithm and the transmission of acknowledgements should be exposed as services by the radio abstraction. For this reason, they consider it as a part of the execution environment rather than the TKN15.4 MAC.

In [19], authors present an experimental evaluation of the Slotted CSMA/CA algorithm by considering realistic conditions using TelosB motes and an implementation of the protocol over TinyOS. The obtained results are compared to the ones presented in [20]. Results in terms of average delay show the gap between theoretical and empirical approaches and they give some implementation considerations that need to be taken into account when designing theoretical models for evaluating the delay in WSNs in order to have a more accurate model to work with.

In [10], authors present an analytical approach and simulation results of unslotted CSMA/CA algorithm with Cooja WSN simulator. The authors showed that using Contiki OS with the default CSMA/CA implementation, a node can only transmit a maximum of 8 kbps. To enhance the network throughput, authors propose an amelioration of the default implemented version of CSMA/CA on Contiki OS. They modified the CSMA/CA algorithm waiting time before performing the Clear channel Assessment (CCA). For the null radio duty cycling algorithm, this time is 125 ms. The CSMA/CA MAC layer uses a callback timer (ctimer) of Contiki to invoke the function responsible for performing the CSMA/CA MAC layer activities corresponding to a packet transmission. The modification was to set the value of the callback timer to 0, so that, in case there is a packet in the MAC layer queue, CSMA/CA mechanism immediately performs clear channel assessment. Simulation results using Cooja simulator showed that the modification increases the throughput of each node in the network. In this implementation, node throughput can reach 45 kbps instead of 8 kbps with Contiki original CSMA/CA implementation. This modification improves the throughput for each node, but some important specifications of the standard CSMA/CA algorithm are missing. The increasing backoff window in case of a CCA busy is not respected. Moreover, when a node has only one packet in its MAC layer queue, the backoff window is not respected but instead the packet is sent immediately.

III. IMPLEMENTING CSMA/CA STANDARD VERSION ON CONTIKI 2.7

Having a standard implementation of CSMA/CA on Contiki OS is important. We need this MAC layer algorithm to be used with other upper layer protocols for performance evaluation. The aim of this paper is to present an implementation of the unslotted CSMA/CA version on Contiki OS which respects the specifications of the IEEE 802.15.4 standard version. In this section, we will describe in details the available implementation of the CSMA/CA algorithm in Contiki OS by identifying the parts that do not respect the standard specifications. We will also explain the modifications that we made in order to make it compliant to the IEEE 802.15.4 standard.

On Contiki OS [3] without radio duty cycling mechanism, when the MAC layer running CSMA/CA algorithm receives a data packet for the upper layer, the packet is enqueued in the MAC layer buffer. But, when the received packet is a broadcast packet, it is not enqueued; rather, it is diffused straight-away, without performing any carrier sensing. The unicast packet is also treated as a broadcast packet when the MAC queue is full. It means that, if no space is available in the MAC layer to enqueue the unicast packet, it is transmitted without any carrier sensing. When the MAC layer sent an unicast packet as a broadcast packet, this packet is not acknowledged and thus it is not retransmitted in case of data collision or corruption. That leads to data loss and reduces the throughput of the network. When the null radio duty cycling mechanism is activated on Contiki OS, whenever the MAC layer running CSMA/CA algorithm has a packet to transmit, it delays medium carrier sensing to 125 ms. Afterwards, it performs carrier sensing and if no activity is detected, the packet is transmitted. If packets acknowledgement (ACK) is enabled, the MAC layer waits for a predefined interval of time to detect the ACK message. If no ACK is received in the allotted time interval (192 us), the system backs off for a random amount of time. The interval of the random backoff depends on the Channel Check Interval (CCI) used by the radio duty cycling mechanism instead of the standard known value of the CSMA/CA algorithm. The backoff time is 125 ms which is equivalent to the default value of the CCI for null radio duty cycling on Contiki OS. When the MAC layer tries to transmit a data packet and it senses that the channel is busy, it backs off for a random amount of time, like already stated above. After every successful packet transmission, the MAC layer with null duty cycling and CSMA/CA algorithm waits for 125 ms before trying to transmit the next packet in the MAC layer queue.

This strange behaviour of the implemented CSMA/CA on Contiki OS limits the throughput as shown in [10]. Our implementation of the CSMA/CA algorithm respects the specifications of the standard version. The main part of the algorithm presented in Fig.1 is described by the following paragraph.

For each transmission attempt, each node holds the following tree variables: *NB*, *CW* and *BE*. *NB* is the number of times the CSMA/CA algorithm was required to backoff while attempting the current transmission. *CW* represents the number of consecutive positive CCAs that need to be done before the transmission starts. *BE* is the backoff exponent, it is the maximum number of backoff periods that a node will wait before attempting to assess the channel.

First, *NB*, *CW*, and *BE* are respectively initialized to 0, 1, and BE_{min}. At the reception of a data packet, any transmission activity is delayed (backoff state) for a random number of backoff periods in the range (0, 2^{BE} - 1) [step 1]. After this delay, channel sensing is performed for one backoff period [step 2]. If the channel is assessed to be busy [step 3], *CW* is set to 1, *NB* and *BE* are increased by 1, ensuring that *BE* is not bigger than BE_{max}, and the algorithm returns to step 1. In case *NB* reaches 4, the algorithm will unsuccessfully terminate, which means that the node does not succeed in accessing the channel. If the channel is assessed to be idle, *CW* is set to 0. When *CW* is equal to 0, the transmission may then start.

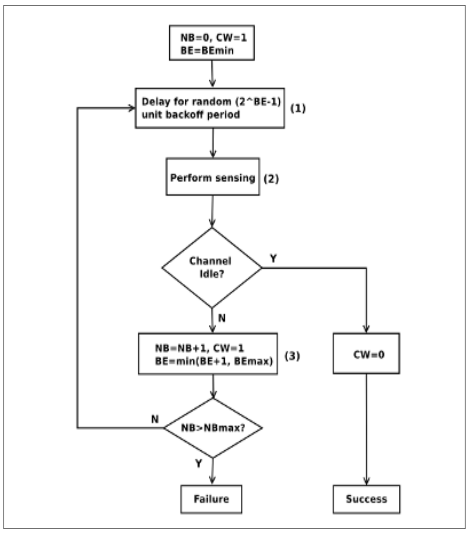

Fig. 1: IEEE 802.15.4 unslotted CSMA/CA algorithm.

IV. SIMULATIONS ENVIRONMENT AND PERFORMANCE EVALUATION

We evaluated the performance of our CSMA/CA implementation by means of simulation performed with the Cooja simulator [11]. Cooja is a flexible Java-based simulator designed for simulating sensor networks running the Contiki operating system [3].

A. *Results analysis*

To analyse the performance of our CSMA/CA implementation, we considered a wide range of simulation settings using a star topology. We chose to evaluate the performance of our implementation according to the following metrics: (i) respect of the backoff duration taking into account the Backoff Exponent (BE), (ii) packet reception rate at the sink node which gives an idea about the capacity of the CSMA/CA algorithm to efficiently deliver the data packets, (iii) the received throughput (S) at the sink node according to the offered load (G) which shows the capacity of the network to support heavy traffic.

In our results, each point of the graphic is an average value over ten simulations. We compare the results of our CSMA/CA implementation with results obtained by the default CSMA/CA version provided on Contiki OS and also with the modified CSMA/CA version proposed by [10].

B. *Respecting backoff duration according to BE value*

Using a scenario of two nodes, one sink and one sender, where the sender has to transmit one data packet per second to the sink, we note the number of backoff periods for the twenty first packets before the CCA mechanism evaluation. Each backoff period is equal to 320 us.

Fig. 2 shows that the backoff duration in terms of backoff periods before performing the CCA for each packet is less than 8 which respects the IEEE 802.15.4 unslotted CSMA/CA specifications. Since in this scenario we have only one sender transmitting to the sink, there is no interference or collision so the maximum backoff periods before each CCA should remain below or equal to 2^{BEmin} - 1 = 7.

In Fig. 3, we use a scenario of 9 nodes where 8 senders have to transmit 2 data packets per second to the sink. We note the number of backoff periods consumed by the node number 4 before the CCA. We notice that in case of CCA busy or packet collision (ACK message is not receive) the BE value is increased.

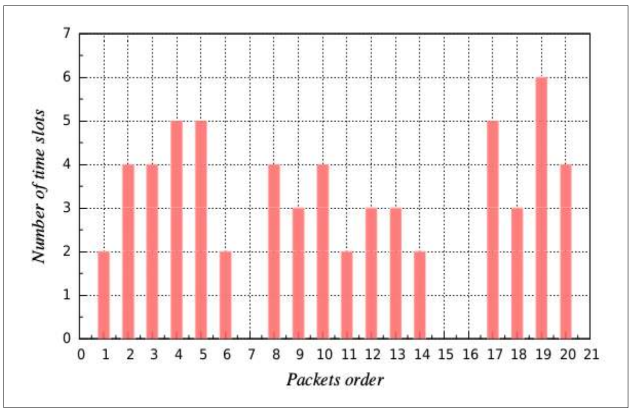

Fig. 2: Backoff duration, 2 nodes scenario.

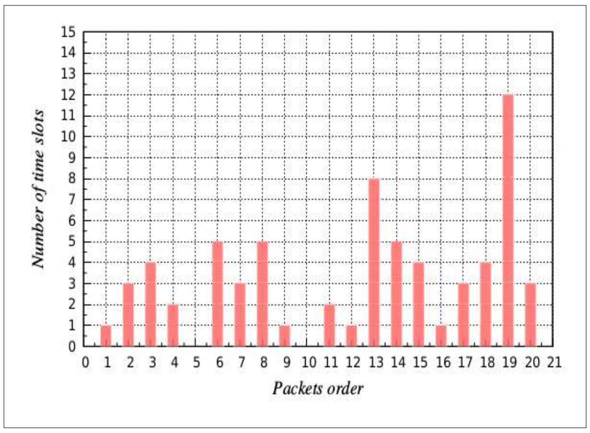

Fig. 3: Backoff duration, 9 nodes scenario.

C. *Packet reception rate at the sink node*

In this scenario, we use a star topology of 9 nodes (one sink and 8 senders) to evaluate the packet reception rate (the reception rate being the ratio of the number of received packets over the number of generated packets). We varied the packet generation rate for each node in the network (except the sink node) in {1 pkt/s, 4 pkt/s, 8 pkt/s, 12 pkt/s, 16 pkt/s, 20 pkt/s, 24 pkt/s, 28 pkt/s}. Where 1 pkt/s represents an underloaded network, whereas 28 pkt/s represents an overloaded network.

Fig. 4 shows that the packet reception rate of our version of CSMA/CA is greater than 60% even under a high traffic load. Whereas the provided CSMA/CA version on Contiki OS gives very low reception rates essentially due to the long periods of backoff that nodes under go before each transmission.

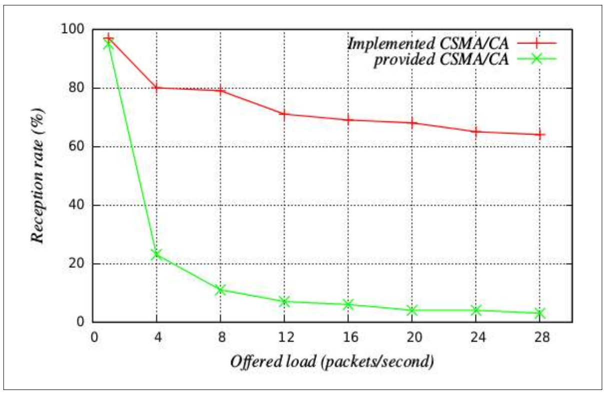

Fig. 4: Packet reception rate vs offered load.

D. *Offered load and throughput*

On Contiki OS, CSMA/CA algorithm is implemented using timers and events. The way events are handled in the operating system has an impact on the throughput. Even though the analytical approach states that IEEE 802.15.4 unslotted CSMA/CA achieves a certain throughput, we cannot conclude that the stated throughput can be achieved by user application. Contiki processes run in a cooperative context and the process of sending a data packet may not respect the schedule of needed traffic load. This makes a challenge to have each node to provide the offered load of 250 kbps. Results are obtained using the same previous topology. Each point is an average value of ten simulations.

Figures 5 and 6 show the received throughput on the sink node depending on the offered load. Fig. 5 represents the results obtained with the provided CSMA/CA on Contiki OS and Fig. 6 compares the results of our implemented version with the modified version (callbackTimeNull CSMA/CA) proposed in [10] where the callback timer for CCA mechanism evaluation is set to 0.

In fig. 5, we present the traffic that was sent in order to show the limitation of the default CSMA/CA, whereas in Fig. 6, the offered load represents the traffic that is generated and offered to the MAC layer.

Fig. 5 shows how the throughput gets a boost once the packet queues are full. Indeed, this is due to the fact that the provided version of CSMA/CA sends packets without any delay when a packet is generated or received and the packet queue is full. The throughput of our implementation outperforms the one obtained with the provided CSMA/CA on Contiki OS. That is essentially due to the delay of 125 ms used by Contiki CSMA/CA before sending a new packet [10]. In our case, only the backoff mechanism of the standard is used before the transmission.

The results of callbackTimeNull CSMA/CA version outperform our implemented version when the offered load is less than 50 kbps. This is because in callbackTimeNull CSMA/CA when the node has only one packet in the MAC buffer queue, the packet is immediately sent without backoffs. But in our standard version, in any case we fire a backoff before CCA mechanism evaluation preceding data transmission.

Notice that in both cases the offered load is less than 250 kbps. It is a challenge to achieve the offered load of 250 kbps because on Contiki OS, processes run in the cooperative context, and these processes run sequentially with respect to other processes in this cooperative context. A cooperative code must run to completion before other cooperatively scheduled code can run. This mechanism affect the execution time of an event when the timer used to schedule this event is the etimer (event timer). The timer used for packet generation on Contiki is the event timer that can explain why the offered load is limited.

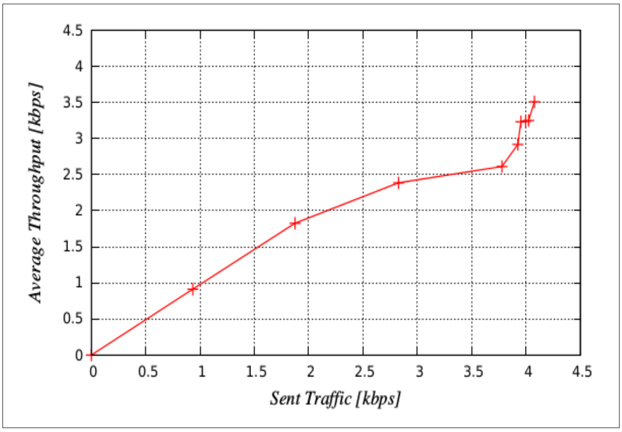

Fig. 5: Sent traffic vs throughput, provided version of CSMA/CA.

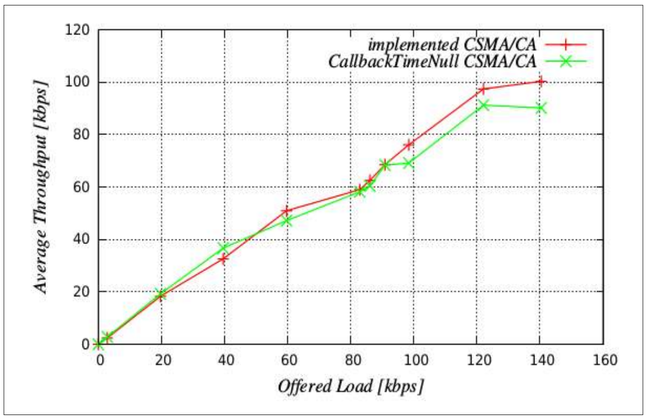

Fig. 6: Offered load vs throughput, implemented version and CallbackTimerNull version of CSMA/CA.

V. CONCLUSION

The CSMA/CA is one of the most used algorithms to access the wireless medium in loosely synchronised wireless networks. The IEEE 802.15.4 standard proposes an unslotted CSMA/CA that is not provided on Contiki OS. In this paper, we propose an implementation of unslotted CSMA/CA which respects the specifications of the IEEE 802.15.4 standard. Simulation results using Cooja simulator show that our implementation respects the IEEE 802.15.4 specifications and outperforms the CSMA/CA default version provided on Contiki OS in terms of packets reception rate and throughput. Using Contiki OS we were not able to achieve the desired offered load. This is essentially due to the fact that the timer that is used for generating data packets is interrupted by the timer used for the CSMA/CA. Our future work will focus on how to provide offered load of 250 kbps using Contiki OS. We also intend to evaluate our implementation on a testbed and implement the slotted version of CSMA/CA on Contiki OS in our future work.

REFERENCES

[1] Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std. 802.15.4, 2006.

[2] Z. Aliance, Zigbee Specifications. San Ramon, CA: Zigbee Standard Organisation, 2008.

[3] A. Dunkels, B. Gr¨onvall, and T. Voigt, "Contiki a lightweight and flexible operating system for tiny networked sensors," in Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, pp. 455–462, November 2004. K. Elissa, "Title of paper if known," unpublished.

[4] P. Jurcık, A. Koubaa, M. Alves, E. Tovar, and Z. Hanz´alek, "A simulation model for the IEEE 802.15.4 protocol: delay/throughput evaluation of the GTS mechanism," in Proceedings of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'07), pp. 109–116, IEEE, Istanbul, Turky, October 2007.

[5] S. Fan, J. Li, H. Sun, and R. Wang, "Throughput analysis of GTS allocation in beacon enabled IEEE 802.15.4," in Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10), pp. 561–565, July 2010.

[6] A. Kumar, P. G. Namboothiri, S. Deshpande, S. Vidhyadharan, K. M. Sivalingam, and S. A. V. Satya Murty, "Testbed based throughput analysis in a wireless sensor network," in Proceedings of the 18th National Conference on Communications (NCC'12), pp. 1–5, February 2012.

[7] J. Edwards, F. Demers, M. St-Hilaire, and T. Kunz, "Comparison of ns2.34's ZigBee/802.15.4 implementation to Memsic's IRIS Motes," in Proceedings of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC '11), pp.986–991, July 2011.

[8] S.W.Golomb, "Mathematical models—uses and limitations,"Simulation, vol. 4, no. 14, pp. 197–198, 1970.

[9] T.-J. Lee, H. R. Lee, and M. Y. Chung, "MAC throughput limit analysis of slotted CSMA/CA in IEEE 802.15.4 WPAN," IEEE Communications Letters, vol. 10, no. 7, pp. 561–563, 2006.

[10] Farooq, M. O., and Kunz, T. (2015). Contiki-Based IEEE 802.15. 4 Channel Capacity Estimation and Suitability of Its CSMA-CA MAC Layer Protocol for Real-Time Multimedia Applications. Mobile Information Systems, 2015.

[11] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in Proceedings of the 31st Annual IEEE Conference on Local Computer Networks (LCN '06), pp. 641–648, November 2006.

[12] Dunkels, A. (2007, January). Rime—a lightweight layered communication stack for sensor networks. In Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session, Delft, The Netherlands.

[13] HAUER, Jan-Hinrich. TKN15. 4: An IEEE 802.15. 4 MAC implementation for TinyOS. 2009.

[14] Levis, Philip, et al. "Tinyos: An operating system for sensor networks." Ambient intelligence. Springer Berlin Heidelberg, 2005. 115-148.

[15] Jan Flora and Philippe Bonnet. Never Mind the Standard Here is the TinyOS 802.15.4 Stack. Technical Report 06-10, University of Copenhagen, 2006.

[16] Wen H, Lin C, Chen ZJ et al. An improved Markov model for IEEE 802.15.4 slotted CSMA/CA mechanism. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 24(3): 495-504 May 2009.

[17] Zheng J, Lee J M. A comprehensive performance study of IEEE 802.15.4. Sensor Network Operations, IEEE Press, 2006, pp.218-237.

[18] Kim, Kyeong Soo, and Leonid G. Kazovsky. "Design and performance evaluation of scheduling algorithms for unslotted CSMA/CA with backoff MAC protocol in multiple-access WDM ring networks." Information Sciences 149.1 (2003): 135-149.

[19] F. Despaux, Ye-Qiong Song, Abdelkader Lahmadi. On the Gap Between Mathematical Modeling and Measurement Analysis for Performance Evaluation of the 802.15.4 MAC Protocol. RTN - 12th International Workshop on Real-Time Networks - 2013, Jul 2013, Paris, France. 2013. <hal-00877452>.

[20] Jelena Misic and Vojislav Misic. Wireless Personal Area Networks: Performance, Interconnection, and Security with IEEE 802.15.4. Wiley Publishing, 2008.

[21] Levis, Philip, and Nelson Lee. "Tossim: A simulator for tinyos networks." UC Berkeley, September 24 (2003).