A Configurable Medium Access Control Protocol for IEEE802.15.4 Networks

Rodrigo Vieira Steiner, Tiago Rogério Mück, and Antônio Augusto Fröhlich

Software/Hardware Integration Lab
Federal University of Santa Catarina
PO Box 476, 88040-900 - Florianópolis, SC, Brazil
{rodrigo,tiago,guto}@lisha.ufsc.br

***Abstract*—This paper presents the new design of C-MAC, a highly configurable, low-overhead Medium Access Control protocol for Wireless Sensor Networks. This new design was developed within EPOSMote, a project targeted at enabling application-specific deployment scenarios for IEEE 802.15.4 networks. The new C-MAC arose from a careful decomposition of several preexisting MAC protocols aiming at obtaining their state machines. These individual state machines were subsequently merged into a generalized one and captured as a component framework that can be specialized to produce a large variety of application-specific protocols. The framework was implemented in C++ using static metaprogramming techniques, thus ensuring that configurability does not come at expense of performance or code size. The experimental results presented in the paper corroborate the new design with figures comparable to non-configurable, platform-optimized implementations.**

I. Introduction

Wireless sensor networks (WSN) are highly dependent on efficient Medium Access Control (MAC) protocols to make effective use of the few resources available on traditional motes, bandwidth and energy in particular, but also memory and processing power. This assertion is confirmed by the large number of MAC protocol proposals available from the literature [1].

Most of the optimizations proposed by existing MAC protocols, however, focus on specific segments of the design space. What is considered an optimization by one class of applications can represent a strong limitation for others. For instance, a protocol optimized for massive data dissemination during a firmware update operation (i.e. short, reliable, low-latency multicast) is certainly not the best choice for sporadic environment monitoring (i.e. long-lasting, sporadic unicasts). A MAC protocol aiming at covering a large fraction of the application universe for sensor networks must feature configuration or adaptation mechanisms directly controlled by applications. Fully automated decision making at system level will never be able to match the knowledge programmers have about their applications.

In this article, we implemented and evaluated EPOS C-MAC in the scope of the EPOSMote project. C-MAC is a highly configurable MAC protocol realized as a framework of medium access control strategies that can be combined to produce application-specific protocols. By selecting the proper strategies and configuring their parameters, programmers can instantiate MAC protocols that closely match their applications' requirements. C-MAC relies on static metaprogramming techniques to achieve high configurability without compromising size and performance. Indeed, a previous implementation of C-MAC for the Mica2 mote produced B-MAC-like instances that are smaller, faster, and make better use of the network than the original TinyOS B-MAC [2]. The EPOSMote devices used in this work feature an IEEE802.15.4 compliant radio. This motivated us to evaluate additional configuration parameters, including synchronization (e.g. beacon-based), contention, and data handling (e.g. error detection). As a result, C-MAC has undergone a major redesign and now features a framework whose elements are more fine-grained and thus can be reused in a larger variety of scenarios.

In Section II we discuss MAC protocols for wireless sensor networks. In Section III we describe the redesign of C-MAC in details. In Section IV we briefly describe the EPOSMote project, within which this research was carried out. In Section V we present an evaluation of the new C-MAC, followed by our conclusions in Section VI.

II. MAC Protocols for WSN

A Medium Access Control (MAC) protocol decides when a network node may access the medium, trying to ensure that different nodes do not interfere with each other's transmissions. In the context of wireless sensor networks, MAC protocols get the additional duty of ensuring an efficient usage of the radio, often the most critical component in terms of power consumption. A MAC in this scenario usually places latency, throughput, and reliability after energy efficiency. Therefore, the main sources of power overhead in radio-based communication (i.e. idle listening, collisions, overhearing, and traffic fluctuations) [3] define guidelines that are followed by virtually all MACs in this realm.

B-MAC [4] is a carrier sense MAC protocol for WSN. It provides an interface which enables on-the-fly reconfiguration, allowing network services to adjust its mechanisms like: enabling and disabling the use of clear channel assessment

(CCA) or acknowledgments, setting preamble length and listening intervals. One problem with B-MAC is that a receiver will have to wait until the entire preamble is sent in order to exchange data, even if it was awake at the start of the transmission. This also results at the overhearing problem, where receivers stay awake until the end of the preamble and find out that the packet was not destined for them. Both these problems are solved by X-MAC [5]. X-MAC uses a short preamble and piggybacks the receiver address into it. Since a receiver can detect if the packet is destined to itself before actually receiving it, it can either turn off the radio (case the packet is not destined to it) or send an acknowledgment to the sender, which will stop sending the preamble and start to send the data. As both these protocols are CSMA based, they suffer from the hidden terminal problem.

S-MAC [6] is a MAC protocol for WSN based on time slots. Its also a CSMA based protocol, but it uses the RTS/CTS (Request To Send/Clear To Send) mechanism to avoid the hidden terminal problem. Neighboring nodes trade synchronization information in order to wake up simultaneously to communicate. A problem with S-MAC is that it does not allows any kind of configuration, neither static nor dynamic, and thus have a fixed duty cycle which can lead to waste of energy (idle listening). T-MAC [7], which is an improvement of S-MAC, addressed this problem and dynamically adapts its duty cycle through fine-grained timeouts. Despite the RTS/CTS mechanism solve the hidden terminal problem, it introduces the external terminal problem, to which both these protocols are exposed. Also the information needed to keep neighboring nodes synchronized produces a certain ammount of overhead.

Z-MAC [8] is a hybrid MAC protocol for WSN, which combines TDMA and CSMA. It uses a TDMA schedule allocating slots for all nodes, but allows nodes to contend for other nodes slots using CSMA. Slots owner are given chances to transmit earlier, so a node can only steal slots from nodes who would not use them. Z-MAC calculates the slot assignements at the time of deployment, which introduces a high overhead at the beginning.

The IEEE802.15.4 MAC layer [9] controls the access to the physical layer in two different ways. On it's basic operating mode, it uses CSMA-CA and acknowledgement packets to handle collisions. On the other operating mode, known as beacon-enabled network, the IEEE802.15.4 uses frame beacons to synchronize the attached devices. These beacons may define an active and an incactive period. The active period is formed by time slots which can be divided in two portions: contention access period (CAP) and contention free period (CFP). During the CAP nodes compete with each others using CSMA-CA. The CFP is formed by guaranteed time slots (GTSs) which are slots allocated to specific nodes, thus no competition for the channel is done during this period. The use of beacon frames allows for a better synchronization between devices thus reducing energy consumption, but it comes at a price which is lower throughput.

III. C-MAC FRAMEWORK

C-MAC is a highly configurable MAC protocol for WSN realized as a framework of medium access control strategies that can be combined to produce application-specific protocols [2]. It enables application programmers to configure several communication parameters (e.g. synchronization, contention, error detection, acknowledgment, packing, etc) to adjust the protocol to the specific needs of their applications. Although highly configurable, C-MAC instances configured to mimic B-MAC proved lighter than the original implementation in terms of footprint, performance, and network use efficiency. This is due to the static metaprogramming techniques used for the implementation of C-MAC in C++, which enable aggressive compiler optimizations.

Nonetheless, the original C-MAC defined configurable protocol elements that were relatively coarse-grained. For instance, synchronization was taken as a single large component that had to be reimplemented for any new protocol even if aspects such as preamble generation and timer synchronization are common factors to virtually any protocol. The redesign presented here aimed at making C-MAC more fine-grained, thus enabling the reuse of microcomponents in a larger variety of application-specific protocols. The starting point for this new design was a decomposition of traditional protocols [10] to obtain a generalized state machine for each of the three major categories: channel polling, scheduled contention, and time division multiple access.

Protocols based on *channel polling* follow a state machine similar to the one presented in Figure 1. They periodically turn on the radio to check for channel activity (`LISTEN`). If activity is detected, the the radio is kept on to receive a packet, otherwise it is immediately turned off (`OFF`). In order to send a packet, the sender node first waits for the channel to be free (`BACKOFF, LISTEN`) and then starts transmitting a synchronization preamble (`TX PREAMBLE`). Receivers listening to the channel will detect the activity and use the preamble (`RX PREAMBLE`) to synchronize themselves with the sender before receiving the packet's payload (`RX DATA`). Note that the state machine presented is non-deterministic, this is because some transitions will only take place depending on the protocol. For example, once a receiver has listened the preamble in X-MAC, if it was destined to it then it will send an acknowledgement to the sender (`TX ACK PREAMBLE`) and only then keep listening to receive the data, otherwise it will go to sleep. However in B-MAC after receiving the preamble the node has no other option than keep listening to receive the data. Furthermore, even the way the protocol is configured affect the transitions that can be taken. For example B-MAC can enable or disable the use of acknowledgements, backoffs and even CCA.

Scheduled contention protocols program the time in which neighboring nodes must wake up to check for channel activity. S-MAC and T-MAC (see section II) are examples of scheduled contention-based protocols. They are generalized by the state machine in Figure 2. At the beginning of each active

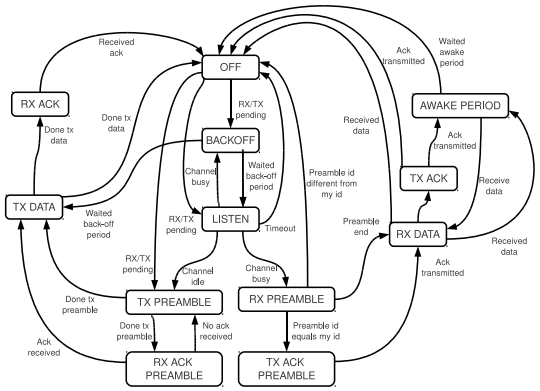

Figure 1. State machine of protocols based on channel polling.

period, neighboring nodes trade additional information to keep themselves synchronized, like the reception (`RX SYNC PKT`) and/or transmission (`TX SYNC PKT`) of schedules in S-MAC. After that, nodes go to the `ACTIVE` state where they are ready to transmit or receive data. Nodes willing to transmit data can contend for the channel using a carrier sense mechanisms (`CCA`) along with a RTS/CTS mechanism (`RX RTS, TX RTS, RX CTS, TX CTS`). After these steps they are ready to transmit/receive with the possibility of using acknowledgment packets (`ACK TX, ACK RX`).

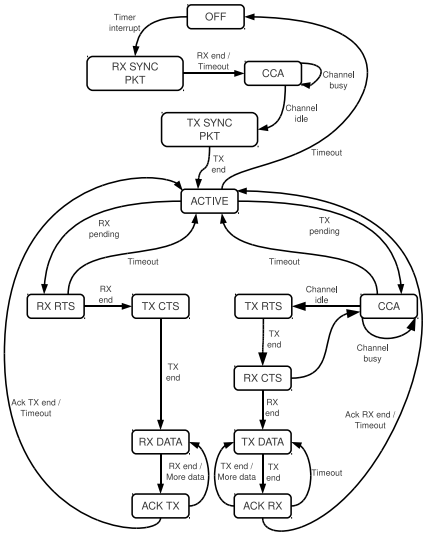

Figure 2. State machine of protocols based on scheduled contention.

Time Division Multiple Access (TDMA) protocols also program the time in which a node must wake up and listen the channel. The difference to scheduled contention lays in the fact that each node (instead of a group of nodes) has a specific time slot to transmit. This way, nodes do not contend for the channel and there are no collisions. However, getting rid of collisions come at the price of lower throughput. Since a node can only transmit during its own time slot, it must remain silent even if other nodes are not transmitting during their slots. Furthermore, this kind of protocol is sensitive to topology changes, requiring a reallocation of time slots whenever it happens. Figure 3 depicts the generalized state machine for TDMA-based protocols. This state machine is rather similar to the one of scheduled contention. Note that the synchronization steps are different if a node is a master (i.e. the coordinator of the network, responsible for time slots allocation) or slave, which is a common configuration for this kind of protocol.

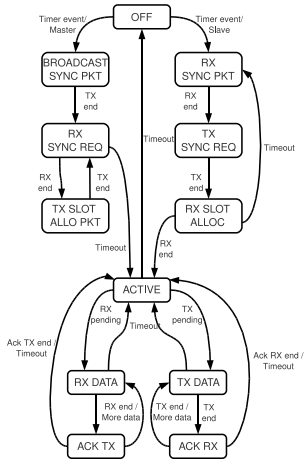

Figure 3. State machine of TDMA-based protocols.

A careful analysis of the state machines of the main classes of MAC protocols led us to the new C-MAC state machine presented in Figure 4. Each state represents a microcomponent which can have different implementations. These microcomponents alongside with states transitions can be combined to produce application-specific protocols. By using static metaprogramming techniques, microcomponents representing states that do not make sense for a certain protocol can be completely removed. When a state is removed, its input transitions are forwarded to the state targeted by its output transitions, still maintaining the original transitions semantics. Besides being able to accommodate representative protocols in any of the three categories, C-MAC state machine also supports hybrid protocols such as Z-MAC and IEEE 802.15.4, thus covering all the protocols discussed in section II.

C-MAC state machine can be triggered either by send/receive events (i.e. when the target protocol has a full duty cycle) or periodically by time events (i.e. when a sleep/active duty cycle is required). The protocol remains in the `OFF` state, with the radio turned off, until one of the previous events triggers the transition to the `SYNCHRONOUS SYNC` state. The dashed states, in Figure 4, represents sets of states related to macro-operations which were encapsulated into a single macro-state for better visualization. The states which compose the `SYNCHRONOUS SYNC` macro-state are shown in Figure 5. These states are related to operations used to synchronize nodes duty cycle and were obtained by merging the synchronization steps shown in the generalized state machines for scheduled contention and TDMA described previously. A node can start the synchronization by broadcasting (`BROADCAST SYNC PKT`) a synchronization packet

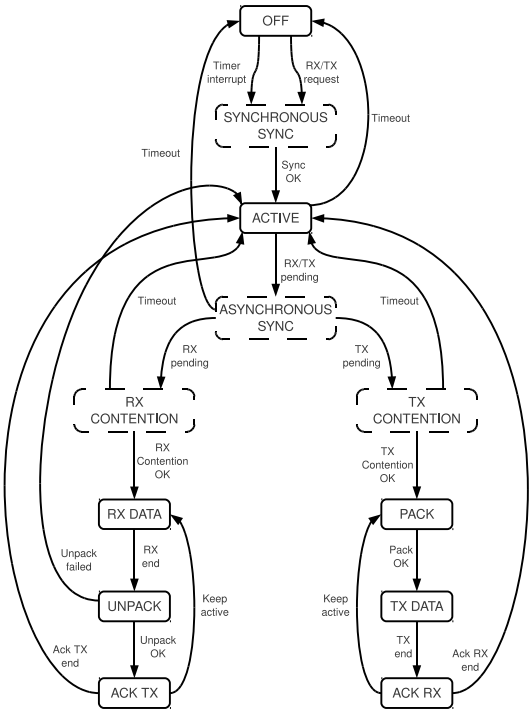

Figure 4. C-MAC state machine.

containing synchronization information (e.g. its schedule in a scheduled contention protocol) which is then followed by the reception of other nodes information (`RX SYNC PKT`) like time slots allocation requests in a TDMA based protocol. At this point a node can either end its synchronization (states in bold belong to C-MAC top view state machine) or perform additional information exchange (`TX SYNC PKT`, `RX SYNC RESP` with the possibility of using a contention mechanism to avoid collisions in this process (`CCA`). Note that in the same protocol each state can be enabled, disabled or perform different operations depending on the node configuration (e.g. the node can be the master, slave or both in a TDMA based protocol). After the node has been properly sinchronized, it goes to the `ACTIVE` state. If there is any transmit/receive request pending, it will go to the related states, otherwise it will go to the `OFF` at the end of its active cycle.

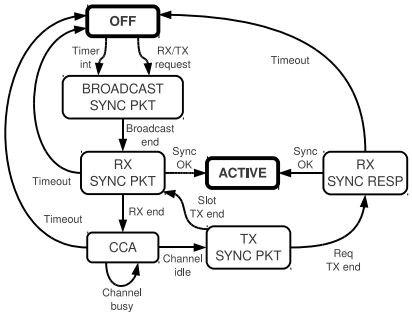

Figure 5. `SYNCHRONOUS SYNC` macro-state state machine.

The `ASYNCHRONOUS SYNC` macro-state, illustrated in Figure 6, is present for protocols that do not require nodes to have a synchronized duty cycle, and thus does not exchange synchronization data in order to communicate (e.g. B-MAC, X-MAC). This is done through the transmission of a large preamble, or a sequence of short preambles. The result is a subset of the states present in the channel polling state machine (Figure 1), as this mechanism is a part of this kind of protocol.

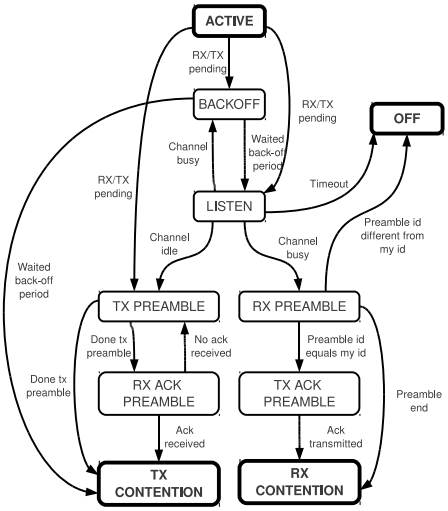

Figure 6. Asynchronous Sync state machine.

When the nodes are ready to transmit or receive, they may go through contention mechanisms before performing these operations in order to avoid collisions. These contention mechanisms are defined by the `RX CONTENTION` and `TX CONTENTION` macro-states whose states machines are shown in Figures 7 and Figure 8, respectively. The states were designed in order to support a carrier sense mechanism (`CCA`), RTS/CTS (`RX RTS`, `TX RTS`, `RX CTS`, `TX CTS`), or both.

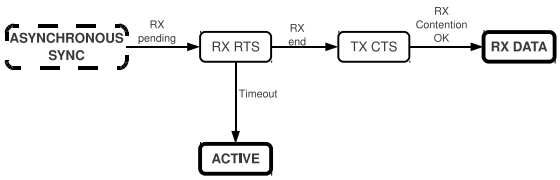

Figure 7. RX Contention state machine.

After going through the contention mechanisms the nodes are ready to transmit or receive data. When the data is received (`RX DATA`), it goes through the error handling and security mechanism (`UNPACK`) and an acknowledgement packet can be transmitted (`ACK TX` state). On the transmit side, error handling and security are appended to the data on the `PACK` state before transmission (`TX DATA`). The `ACK RX` state implements acknowledgement packets reception. Some protocols allows the transmission of bursts of data packets (e.g. X-MAC and S-MAC) without contenting for the medium again, which required the `Keep alive` transitions.

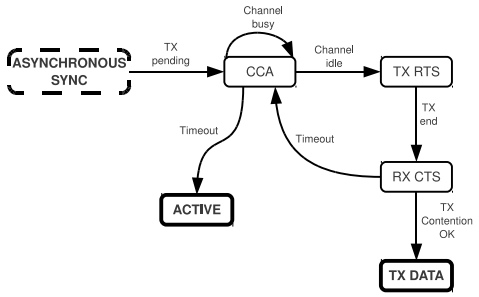

Figure 8. TX Contention state machine.

Through these new state machines we were able to expand C-MAC and provide a larger range of configurable points, while achieving a higher level of components reuse. The main C-MAC configuration points now include:

Physical layer configuration: These are the configuration points defined by the transceiver under utilization (e.g. frequency, transmit power, date rate). This is hardware dependent and was already present into the previous version.

Synchronization and organization: Provides mechanisms to send or receive synchronization data to organize the network and synchronize the nodes duty cycle. The new state machines allows the use of mechanisms that could not be used in the previous version.

Collision-avoidance mechanism: Defines the contention mechanisms used to avoid collisions. May be comprised of a carrier sense algorithm (e.g. CSMA-CA), the exchange of contention packets (*Request to Send* and *Clear to Send*), or a combination of both. These mechanism were present in the previous version, but with a lower level of reusability.

Acknowledgment mechanism: The exchange of *ack* packets to determine if the transmission was successful. The previous version had only support to payload acknowledgements, support for preamble acknowledgement is now a possibility.

Error handling and security: Determine which mechanisms will be used to ensure the consistency of data (e.g. CRC check) and the data security. Already present in the previous version, but with a lower level of reusability.

IV. The EPOSMote

The goal of the EPOSMote project is to develop an EPOS-based WSN node focused on environmental monitoring. The nodes have the following main requirements:

Low energy consumption: Usually it is not practicable to often send teams to the field to replace the nodes batteries, or the nodes may become inaccessible for long time periods (e.g. nodes inside bags of coffee monitoring its storage conditions).

Environmental monitoring features: The nodes must feature sensors to measure the environmental conditions, such as temperature, humidity, etc.

Environmental integration: The environment should not be affected by the nodes presence and vice-versa, so the nodes must be as small, salubrious and strong as possible.

Figure 9 shows the overview of the EPOSMote's architecture. Its hardware is based on the ATmega1281 [11] and AT86RF230 [12] platforms from Atmel. The ATmega1281 microcontroller has 128 Kbytes of programmable flash and 8 Kbytes of data memory. It offers a low-power and low-cost solution with reasonable performance. The AT86RF230 radio is a low-power IEEE802.15.4 compliant hardware that works on the 2.4 GHz frequency range with O-QPSK modulation. AT86RF230 integrates most of the components required for communication, thus allowing a small hardware project. A SHT11 sensor is used to measure the environment temperature and humidity. Yet expensive, this sensor is used due to its very small size. Figure 10 shows the final hardware. The EPOSMote is littler than a £2 coin.

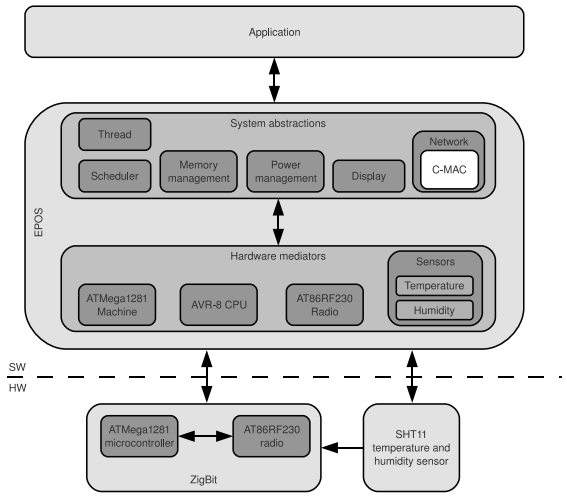

Figure 9. Architectural overview of EPOSMote.

Figure 10. EPOSMote side-by-side with a £2 coin.

V. Experimental results

In order to evaluate the new configuration points introduced into C-MAC we implemented a configurable IEEE802.15.4 MAC on the EPOSMote. C-MAC was evaluated by varying the following parameters available on IEEE802.15.4:

- Full Beacon-enabled.
- Full Non-beacon.
- Non-beacon without CSMA-CA.
- Non-beacon without ACK.
- Non-beacon without both CSMA-CA and ACK.

We used an network topology where one node is defined as the coordinator and the other nodes are placed around the coordinator in a way that each node is within range of other

nodes, and may potentially interfere in the communications of every other node. This topology was configured to simulate a typical environmental monitoring application, where the coordinator receives data transmitted periodically by other nodes monitoring the environment. Other important parameters used in the experiments setup are shown in Table I. The *beacon order* and *superframe order* parameters are used only on the beacon-enabled configuration to control the nodes duty cycle according to the specifications in the IEEE802.15.4 standard [9].

Table I
CONFIGURATION PARAMETERS USED ON THE EXPERIMENTS

Parameter	Value
Compiler	GCC 4.0.2
Microcontroller clock	1 MHz
Packet size	64 bytes
TX power	3 dBm
Beacon order	7
Superframe order	4
Duty cycle	12%

To analyze memory footprint we used the avr-size tool, from GNU Binutils version 2.19. The results for each configuration used can be seen in Table II. As expected, the more complex the configuration, the more worse was it footprint. Thus, the configuration with no beacons, CSMA and ACK yielded the best footprint, while the Full/Beacon-enabled configuration yielded the worst. Nevertheless, C-MAC's meta-programmed implementation, along with EPOS's component architecture also delivered equivalent functionality with smaller footprint than other known MAC protocols for WSN [4], [5], [6], [7].

Table II
MEMORY FOOTPRINT OF C-MAC ON EPOS

Configuration	Code (bytes)	Data (bytes)
No CSMA-CA / ACK	3248	185
No ACK	3572	185
No CSMA-CA	3768	202
Full	4092	202
Full/Beacon-enabled	5344	215

To evaluate latency, we measured the round-trip time of a packet between two nodes. The results in Table III shows that the latency increases as more features of the protocol are enabled. On a beacon enabled network, the beacon order and superframe order parameters shown in Table I results in a duty cycle of 12% with a sleep period of about 2 seconds, which is the dominating factor when this configuration is used. On the other side the time spent with idle listening is reduced, thus reducing the energy consumption as can be seen in Figure 13.

Figure 11 shows the variations of the average throughput as the number of nodes on the network increases. The overall throughput improves as the features of the protocol are removed and presents small variations as the number of nodes increase, this is due to the low network traffic and the non-coincidence of the period of transmission of the nodes. The

Table III
ROUND-TRIP TIME OF A PACKET BETWEEN TWO NODES

Configuration	RTT (ms)
No CSMA-CA / ACK	60
No ACK	68
No CSMA-CA	71
Full	79
Full/Beacon-enabled	1882

exception is when we enable the use of ACK packets and disable CSMA-CA. With this configuration there is a high packet loss due to collisions and the retransmissions ends up reducing the protocol's performance.

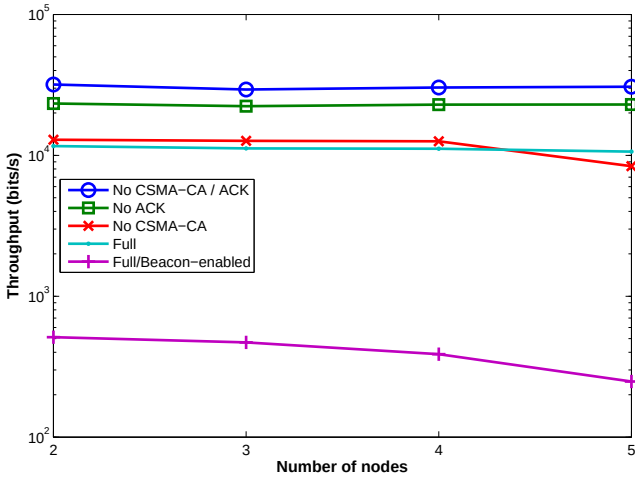

Figure 11. Average network throughput (logarithmic scale).

The low duty cycle used on the Full/Beacon-enabled configuration yielded the worse throughput. This configuration also yielded the biggest performance deterioration with the increase on the number of nodes. This is due to the fact that all nodes try to communicate at the same small period, increasing the chance of collisions and the packet loss rate, as can be seen in Figure 12. For the configurations without beacon synchronization, the packet loss rate varies similar to the throughput.

C-MAC's energy efficiency were evaluated by measuring the energy consumed per byte received at the coordinator. Figure 13 shows the results. As expected, the configurations with beacon synchronization yielded the best results through the synchronization of the nodes duty cycle. Except for the beacon-enabled configuration, the energy received per byte decreases as the number of nodes increases. This happens because the main source of energy consumption is the idle listening, keeping the total energy consumed uniform as the topology changes. As the network traffic increases, the average energy consumed per byte also decreases. This is not the case on the beacon-enabled configuration, showing that it successfully treated the idle listening problem.

With C-MAC, it's possible to adjust the protocol according to the application's requirements. The results showed that we

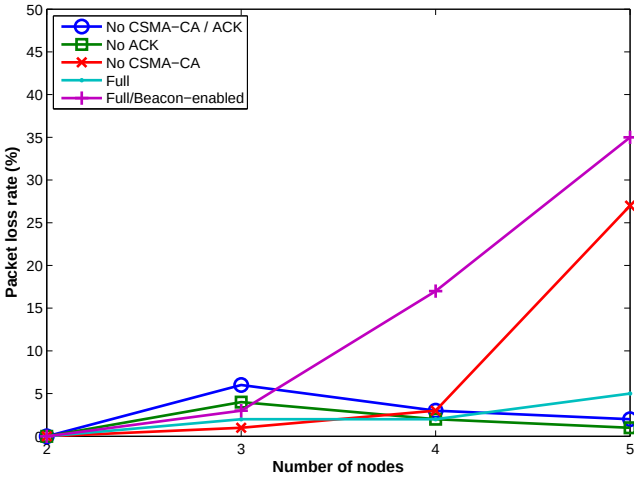

Figure 12. Average packet loss rate.

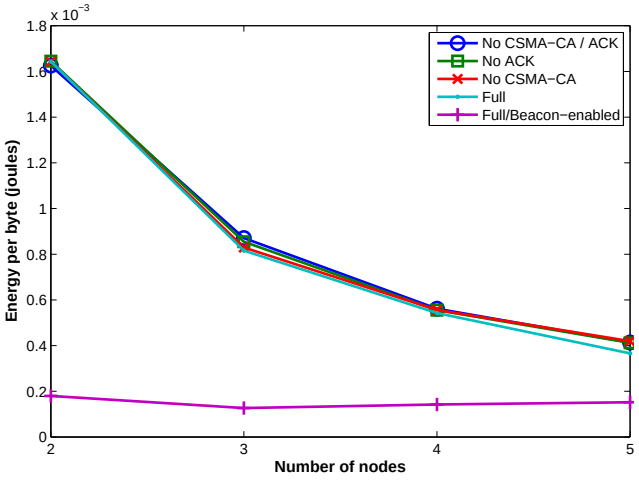

Figure 13. Energy consumed per byte received on the coordinator.

can easily control the trade-offs among memory footprint, latency, throughput, reliability and energy consumption, by using different configurations of IEE802.15.4. For example, on environments where there are a big number of nodes transmitting very often, but there is no need to guarantee the delivery of messages, an implementation without acknowledgment packets can be used to increase the network throughput. On environments that contains a big number of nodes and they will transmit as little as possible to save energy, a configuration with CSMA-CA and acknowledgment packets can be used to guarantee the message exchange. On applications with more tolerant energy consumption requirements, the beacons can be disabled to obtain higher throughput and lower latency.

VI. Conclusion

This paper presented the new design of C-MAC, a highly configurable, low-overhead Medium Access Control protocol for Wireless Sensor Networks. This new design was developed within EPOSMote, a project targeted at enabling application-specific deployment scenarios for IEEE 802.15.4 networks. The new C-MAC arose from a careful decomposition of several preexisting MAC protocols aiming at obtaining their state machines. These individual state machines were subsequently merged into a generalized one and captured as a component framework that can be specialized to produce a large variety of application-specific protocols. The framework was implemented in C++ using static metaprogramming techniques (e.g. templates, inline functions, and inline assembly), thus ensuring that configurability does not come at expense of performance or code size.

We experimentaly evaluated C-MAC in terms of memory footprint, latency, throughput, packet loss rate, and energy consumption by varying IEEE 802.15.4 main configuration aspects. The results corroborate the new design with figures comparable to the non-configurable, platform-optimized implementation provided by Meshnetics. Applications using EPOSMote can now easily configure a MAC protocol to closely match their requirements.

References

[1] A. Bachir, M. Dohler, T. Watteyne, and K. Leung, "Mac essentials for wireless sensor networks," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 2, pp. 222 –248, second 2010.

[2] L. Wanner, A. de Oliveira, and A. Frohlich, "Configurable medium access control for wireless sensor networks," *IFIP International Federation For Information Processing-Publications*, vol. 231, pp. 401–410, 2007.

[3] K. Langendoen and G. Halkes, *Embedded Systems Handbook*. CRC Press, 2005, ch. Energy-Efficient Medium Access Control.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 95–107.

[5] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2006, pp. 307–320.

[6] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *IEEE INFOCOM*, vol. 3. Citeseer, 2002, pp. 1567–1576.

[7] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003, pp. 171–180.

[8] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-mac: a hybrid mac for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, 2008.

[9] IEEE Computer Society, "IEEE Standard 802 Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," The Institue of Electrical and Electronics Engineers, Technical report, 2006.

[10] K. Klues, G. Hackmann, O. Chipara, and C. Lu, "A component-based architecture for power-efficient media access control in wireless sensor networks," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007, pp. 59–72.

[11] Atmel, *8-bit Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash*, 2007.

[12] ——, *Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM Applications*, 2007.