C-MAC: a Configurable Medium Access Control Protocol for Sensor 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*—C-MAC is a highly configurable MAC protocol realized as an architecture 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. 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. In this work, we implemented and evaluated EPOS C-MAC in the scope of the EPOSMote project. The EPOSMote devices used in this work feature an IEEE 802.15.4 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 an architecture whose elements are more fine-grained and thus can be reused in a larger variety of scenarios.**

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].

Nevertheless, most of the optimizations proposed by existing MAC protocols 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.

C-MAC is a highly configurable MAC protocol for WSN realized as an architecture 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 produced better instances than the original implementation in terms of footprint, performance, and network usage 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 [2] used 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 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 of the three major categories [3]: channel polling, scheduled contention, and time division multiple access.

Section II describes the redesign of C-MAC in details. Section III briefly describes the EPOSMote project, within which this research was carried out. Section IV presents an evaluation of the new C-MAC, and Section V concludes the paper.

II. NEW C-MAC DESIGN

A careful analysis of channel polling (e.g B-MAC [4], X-MAC [5]); scheduled contention (e.g. S-MAC [6], T-MAC [7]); and TDMA protocols led us to the new C-MAC architecture presented in Figures 1, 2 and 3. We used activity diagrams where each activity is executed by a microcomponent which can have different implementations. These microcomponents alongside with the flow control can be combined to produce application-specific protocols. By using static metaprogramming techniques, microcomponents representing activities that do not make sense for a certain protocol can be completely removed. When an activity is removed, its inputs are forwarded to the activity targeted by its outputs, still maintaining the original flow semantics. Besides being able to accommodate representative protocols in any of the three categories, C-MAC architecture also supports hybrid protocols such as Z-MAC and IEEE 802.15.4.

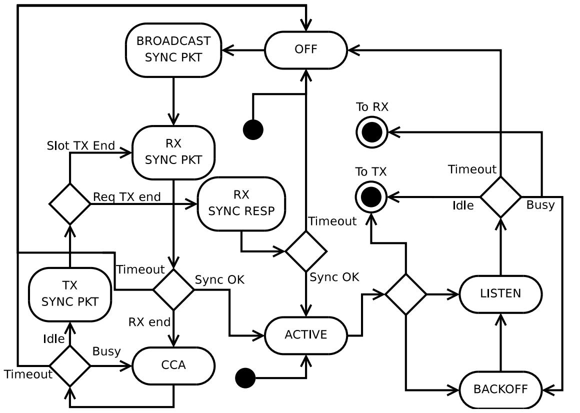


Figure 1: Synchronization Activity Diagram.

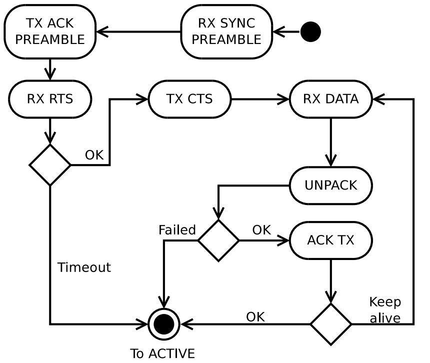


Figure 2: Reception Activity Diagram.

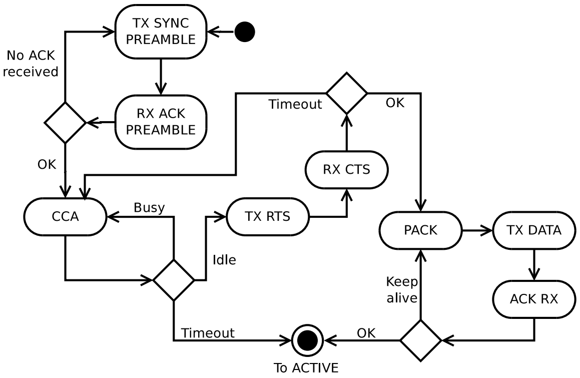


Figure 3: Transmission Activity Diagram.

C-MAC 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 `OFF`, with the radio turned off, until one of the previous events activates it. Figure 1 presents the activities used to synchronize duty cycle. A node can start the synchronization by broadcasting (`BROADCAST SYNC PKT`) a synchronization packet 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 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 activity 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 synchronized, it executes any transmission/reception request pending, otherwise it goes `OFF` at the end of its active cycle. When the nodes are ready to transmit or receive, they may go through contention mechanisms in order to avoid collisions. These contention mechanisms are defined by the carrier sense (`CCA`), and RTS/CTS (`RX RTS, TX RTS, RX CTS, TX CTS`) microcomponents.

Some protocols 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 (`TX SYNC PREAMBLE, RX SYNC PREAMBLE, RX ACK PREAMBLE, TX ACK PREAMBLE`).

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`). On the transmission side, error handling and security are appended to the data on the `PACK` microcomponent before transmission (`TX DATA`). The `ACK RX` microcomponent implements acknowledgement packets reception. Some protocols allows the transmission of bursts of data packets (e.g. X-MAC and S-MAC) without contending for the medium again, which required the `Keep alive` flows.

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

Physical layer configuration: These are the configuration points defined by the underlying transceiver (e.g. frequency, transmit power, date rate).

Synchronization and organization: Provides mechanisms to send or receive synchronization data to organize the network and synchronize the nodes duty cycle.

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.

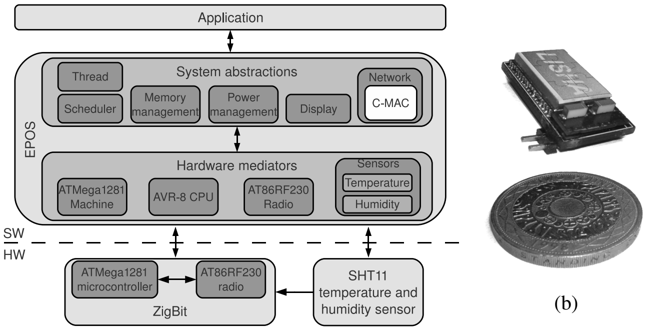
Acknowledgment mechanism: The exchange of *ack* packets to determine if the transmission was successful, including preamble acknowledgements.

Error handling and security: Determine which mechanisms will be used to ensure the consistency of data (e.g. CRC check) and the data security.

III. THE EPOSMOTE

The goal of the EPOSMote project is to develop an EPOS-based WSN node focused on environmental monitoring [8]. The nodes have the following main requirements: low energy consumption, environmental monitoring features, and environmental integration.

Figure 4a shows an overview of the EPOSMote architecture. Its hardware is designed as a layer architecture composed by a main module, a sensoring module, and a power module. The main module is responsible for processing and communication. It is based on the ATmega1281 microcontroller and the AT86RF230 radio from Atmel. We have developed a sensoring module based on the SHT11 sensor, which is used to measure the environment temperature and humidity. Yet expensive, we chose this sensor due to its very small size. Figure 4b shows the final hardware. The EPOSMote is littler than a £2 coin.



(a)

(b)

Figure 4: (a) Architectural overview of EPOSMote. (b) EPOSMote side-by-side with a £2 coin.

IV. EXPERIMENTAL RESULTS

In order to evaluate C-MAC we have implemented a configurable IEEE 802.15.4 MAC on the EPOSMote. C-MAC was configured to behave like following the MACs, which were based on IEEE 802.15.4: *No CSMA-CA / ACK*; *No ACK*; *No CSMA-CA*; *CSMA-CA / ACK enabled*; and *CSMA-CA / ACK beacons enabled*.

In the experiments, we have used a network topology that simulates a typical monitoring application, where a coordinator receives data transmitted periodically by other nodes monitoring the environment. One node is defined as the coordinator and other nodes are placed around the coordinator. The nodes are placed in a way in which they are within range of each other, so each node may potentially interfere in the communication of every other node. For all experimets we used: GCC 4.0.2 to compile EPOS and the application; the ATMEGA1281 clock set to 1 MHz; packets size of 64 bytes; 3 dBm of TX power; for the beacon-enabled configurations, beacon and superframe order were set to 7 and 4, yielding a duty cycle of 12%.

A. Results

We have used the avr-size tool, from GNU Binutils version 2.19, to analyse the size of the applications. The result for each configuration is shown in Table I. As expected, the more complex the configuration, the larger the footprint. Thus, the configuration with no beacons, CSMA, and ACK yielded the smallest footprint, while the full IEEE 802.15.4 configuration yielded the largest one. Also, we compared C-MAC with the IEEE 802.15.4 MAC provided by Meshnetics ZigBeeNet [9] for the ATMega1281/AT86RF230. C-MAC's meta-programmed implementation, along with EPOS's component architecture, delivered equivalent functionality with smaller footprint than a non-configurable, platform-optimized implementation.

Table I: Memory footprint and latency of C-MAC IEEE 802.15.4 and ZigBeeNet IEEE 802.15.4. The latency is the round-trip time between two nodes.

Configuration	Code (bytes)	Data (bytes)	RTT (ms)
No CSMA-CA / ACK	3248	185	60
No ACK	3572	185	68
No CSMA-CA	3768	202	71
CSMA-CA / ACK enabled	4092	202	79
CSMA-CA / ACK beacons enabled	5344	215	1882
ZigBeeNet MAC (non-beacon)	26776	289	62

To evaluate latency, we have measured the round-trip time of a packet between two nodes. The results in Table I show that the latency increases as more features of the protocol are enabled. On a beacon enabled network, duty cycle of 12% results in a sleep period of about 2 seconds, which is the dominating factor when this configuration is used. However, the time spent with idle listening is reduced, thus reducing the energy consumption as shown in Figure 7. Also, C-MAC latency is comparable to the ZigBeeNet MAC, which shows that C-MAC configurability does not come at expense of performance.

Figure 5 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 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.

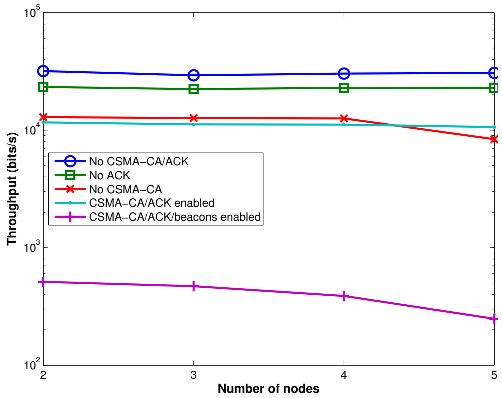


Figure 5: Average network throughput (logarithmic scale).

The low duty cycle used on the *CSMA-CA / ACK beacons enabled* configuration yielded the worse throughput. This configuration also yielded the biggest performance deterioration as the number of nodes increases. This is due to the fact that all nodes try to communicate at the same small period, increasing the chance of collisions and packet loss rate, as shown in Figure 6. For configurations without beacon synchronization, the packet loss rate varies similar to the throughput.

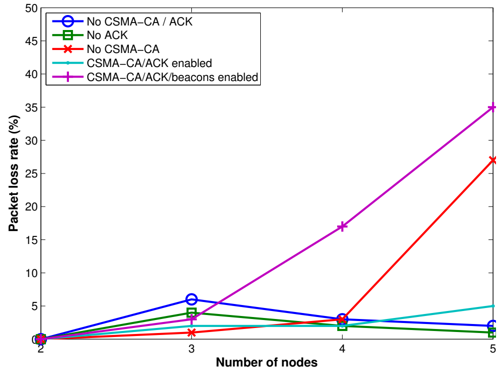


Figure 6: Average packet loss rate.

C-MAC's energy efficiency were evaluated by measuring the energy consumed per byte received at the coordinator. Figure 7 shows the results. As expected, the configurations with beacon synchronization yielded the best results. Except for the beacon-enabled configuration, the energy per byte decreases as the number of nodes increases. This happens because the main source of energy consumption is idle listening. As the network traffic increases, the average energy consumed per byte decreases. This is not the case on the beacon-enabled configuration, showing that it successfully treated the idle listening problem.

V. 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

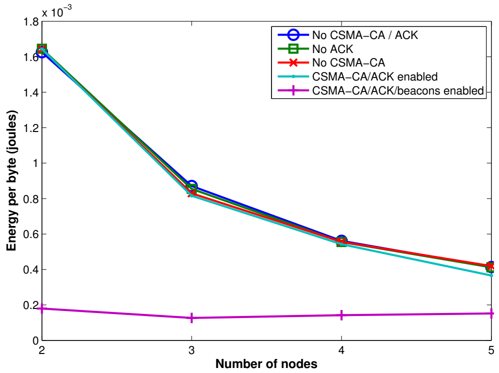


Figure 7: Energy consumed per byte received on the coordinator.

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 resulting in a component architecture that can be specialized to produce a large variety of application-specific protocols. The architecture 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 Int. Fed. For Info. Processing-Publications*, vol. 231, pp. 401–410, 2007.

[3] 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*. New York, NY, USA: ACM, 2007, pp. 59–72.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04*. 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*. 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*. New York, NY, USA: ACM, 2003, pp. 171–180.

[8] Epos project. [Online]. Available: http://epos.lisha.ufsc.br

[9] MeshNetics, "Doc. M-252 08 v.1.0: ZigBeeNet - ZigBee Stack and Software Development Kit," Technical report, 2008.