WiDom: A Dominance Protocol for Wireless Medium Access

Nuno Pereira, *Member, IEEE*, Björn Andersson, *Member, IEEE*, and Eduardo Tovar, *Member, IEEE*

***Abstract*—In this paper, we address the problem of sharing a wireless channel among a set of sporadic message streams where a message stream issues transmission requests with real-time deadlines. We propose a collision-free wireless medium access control (MAC) protocol which implements static-priority scheduling, supports a large number of priority levels and is fully distributed. It is an adaptation to a wireless channel of the dominance protocol used in the CAN bus. But, unlike that protocol, our protocol does not require a node having the ability to receive an incoming bit from the channel while transmitting to the channel. The evaluation of the protocol with real embedded computing platforms is presented to show that the proposed protocol is in fact collision-free and prioritized. We measure the response times of our implementation and show that the response-time analysis developed for the protocol offers an upper bound on the response times.**

***Index Terms*— Wireless LAN, Medium Access Control, Schedulability Analysis.**

I. Introduction

Many emerging embedded applications are designed to respond to stimuli from the environment. Typically, these events are triggered sporadically; that is, the exact time of a transmission request is unknown but a lower bound on the time between two consecutive transmission requests from the same message stream is known. Such traffic is called *sporadic message streams*. Given such setting, we address the problem of sharing a communication channel such that timing requirements are satisfied. More specifically, and given the eagerness for wireless communication in emerging embedded systems, including those for the industrial automation, in this paper we address the problem of sharing a wireless communication channel, and providing timeliness guarantees.

While many scheduling algorithms and analysis techniques for wireless communications are available for periodic messages, the case of sporadic messages is less studied. Most of the current wireless protocols cannot be analyzed to offer pre-run-time guarantees that sporadic messages meet deadlines, and the protocols that do offer such guarantees rely on polling, which is inefficient when the deadline is short and the minimum time between two consecutive requests is long.

In wired networks, sporadic messages can be efficiently scheduled using the Controller Area Network (CAN) bus [1], and this has already proven to be useful in industry, as witnessed by the pervasive use of the CAN bus. It has a medium access control (MAC) protocol which is collision-free and prioritized, and hence it is possible to schedule the bus such that if message characteristics (minimum inter-arrival times, transmission times, jitter, etc.) are known, then it is possible to compute upper bounds on message delays [2, 3]. This MAC protocol belongs to a family called *dominance* or *binary countdown protocols* [4].

In this paper, the approach to solve the problem of sporadic message scheduling on a wireless channel relies on adapting dominance/binary countdown protocols to a wireless channel. This adaptation is non-trivial. Firstly, implementations of dominance protocols for a wired medium are based on a wired-AND behaviour of the bus, where the dominant signal overwrites the recessive signal. Secondly, these implementations require that nodes are able to monitor the medium while transmitting. Clearly this does not easily extend to the case of wireless channels. Moreover, due to non-idealities of transceivers and nature of the wireless medium, it is not obvious how a dominance protocol should be implemented. An implementation of the proposed dominance protocol is presented. This implementation is named WiDom.

We evaluate WiDom experimentally in short-range communication, assuming that all computer nodes obey to the protocol. To demonstrate that our protocol supports a large number of priorities, the experimental evaluation was performed with 2^{10} priority levels. Although this number of priorities introduces overhead, the application developer has the freedom to choose the number of priority levels required, and thus possibly reduce the overhead introduced. Such a large number of priorities can be supported by other prioritized protocols (see e.g.,[5]) only at the cost of an overhead several orders of magnitude higher. The experimental evaluation of WiDom shows that the probability that a message is transmitted collision-free, correctly prioritized and received (neither lost nor corrupted) by all

Manuscript received September 14, 2006; revised January 31, 2007. This work was supported in part by the Portuguese Foundation for Science and Technology (FCT – Fundação para a Ciência e Tecnologia) and the Network of Excellence on Embedded Systems Design ARTIST2 (IST-004527).

Nuno Pereira is with the IPP-HURRAY Research Group, at the Polytechnic Institute of Porto, Portugal (e-mail: nap@isep.ipp.pt).

Björn Andersson is with the IPP-HURRAY Research Group, at the Polytechnic Institute of Porto, Portugal (e-mail: bandersson@dei.isep.ipp.pt).

Eduardo Tovar is with the IPP-HURRAY Research Group, at the Polytechnic Institute of Porto, Portugal (e-mail: emt@dei.isep.ipp.pt).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

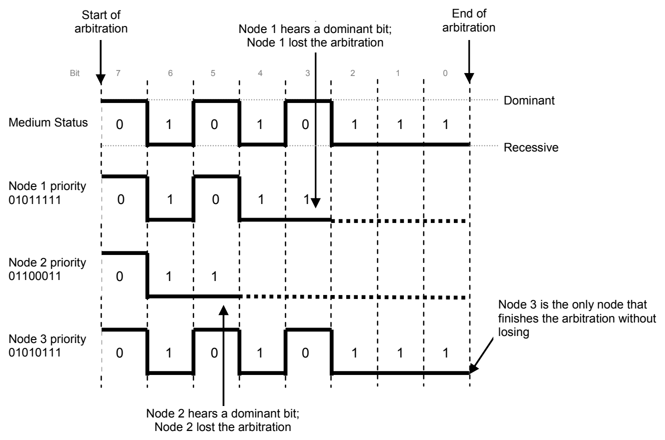
Start of arbitration
Node 1 hears a dominant bit; Node 1 lost the arbitration
End of arbitration
Bit 7 6 5 4 3 2 1 0
Dominant
Medium Status
Recessive
Node 1 priority 01011111
Node 2 priority 01100011
Node 3 is the only node that finishes the arbitration without losing
Node 3 priority 01010111
Node 2 hears a dominant bit; Node 2 lost the arbitration

Fig. 1. Arbitration in Dominance/Binary Countdown Protocols.

other nodes is at least 99.9%. This reliability justifies the study of schedulability analysis techniques for sporadic messages in wireless networks; hence a response-time analysis for WiDom is developed and tested in this paper as well. The industrial relevance of this work is that we show that is possible to engineer industrial applications (from the timeliness perspective) in a similar fashion to what engineers do currently for CAN-based systems in industrial environments.

The remainder of this paper is structured as follows. Section II provides the necessary background on dominance/binary countdown protocols, which are the basis for the proposed MAC mechanism. Section II also states the system model, assumptions and terminology employed throughout the rest of the paper. In Section III, the proposed dominance protocol for wireless media is presented, and the rationale behind its design is discussed. Section IV unveils details of the protocol implementation. The pre-run-time schedulability analysis for the protocol is introduced in Section V, and Section VI evaluates the protocol implementation through experimentation with real embedded computing platforms. Section VII discusses previous related work, and finally, conclusions are drawn in Section VIII.

II. Background

A. Dominance/Binary Countdown Protocols

Dominance/binary countdown protocols [4] are the main inspiration for the proposed protocol. In such protocols, nodes (it can be messages) are assigned unique priorities. A node that requests to transmit waits for a pre-determined time interval until the channel is idle. Then it starts a conflict resolution phase – the arbitration – where each node sends its unique priority bit-by-bit starting with the most significant bit, while monitoring the medium at the same time. The medium must be devised in such a way that nodes will only detect a recessive bit if no node is transmitting a dominant bit. If any node is transmitting a dominant bit, then every node will detect a dominant bit regardless of what the node itself is sending. During the arbitration, if a node contends with a recessive bit but hears a dominant bit, then it will refrain from transmitting any further bits and will only monitor the medium. Finally, only one node reaches the end of arbitration without hearing a dominant bit, and therefore will proceed with transmitting the data part of the message.

The arbitration performed in dominance/binary countdown protocols is illustrated through an example in Fig. 1. Three nodes with different priorities contend for the channel. If a bit is “0” then it is dominant and if a bit is “1” then it is recessive. Thus, low priority numbers represent higher priorities. When a node with a recessive bit detects a dominant bit, then it knows it has lost the arbitration.

B. System Model

Consider n message streams $\tau_1, \tau_2, \tau_3, ..., \tau_n$ and m computer nodes $N_1, N_2, ..., N_m$. A message stream is assigned to one node only. But many message streams can be assigned to one node.

Workload. Message stream τ_i makes an infinite sequence of requests to transmit. The exact time of a transmission request is unknown, but a lower bound on the time between two consecutive transmission requests from the same message stream is known. This lower bound is denoted as T_i. Every message from τ_i requires C_i contiguous time units to transmit. The maximum time elapsed from the time instant of a request from τ_i to the completion of the transmission of that message is called the *response time* of τ_i, and it is denoted as R_i.

Success and failure. If there is an overlap between a pair of transmitted data bits, then a collision has occurred and both transmissions have failed. Every time a message from τ_i is requested to be transmitted it needs to finish the transmission at most D_i (the relative deadline of τ_i) time units after it was requested. The goal of the proposed protocol is to schedule all messages in all message streams such that all transmissions are accomplished before their relative deadlines, and without any collision of data bits. Then, the protocol has succeeded.

Priorities. Priorities are assigned univocally to message streams; these priorities are non-negative integers. *npriobits* denotes the number of bits used to represent the priorities.

Propagation. The time-of-flight between two arbitrary nodes N_i and N_j is unknown, but it is non-negative and there is an upper bound α on the time-of-flights. A single broadcast domain is assumed. When a node transmits a message and there is no collision, then all nodes receive exactly one copy of the message; that is, no hidden terminals exist.

Nodes. Nodes are equipped with real-time clocks. These are not synchronized; that is, their values may be different. Therefore, we consider that for every unit of real-time, the clock increases by an amount in the range $[1-\varepsilon, 1+\varepsilon]$, $0 < \varepsilon < 1$. Let *CLK* denote the granularity of the clock.

A message may have one intended node (unicast) or all nodes (broadcast) as receivers; the proposed protocol deals with both. It is assumed that when a node receives a message it does not send an acknowledgement. This assumption could easily be removed for unicast, by adding the acknowledgement time to the message transmission time. A node can sense other transmissions only if it is not transmitting. No particular modulation technique or coding scheme for the data bits is assumed, but when data bits are transmitted, there is no interval of continuous idle time that exceeds F time units. (F is a design parameter, see Section III). Nodes can transmit a carrier wave, and all nodes are able to detect that carrier if they do not transmit themselves. *TFCS* denotes the time to detect that a carrier wave was transmitted. *SWX* denotes the time to switch from transmission to reception or vice-versa (it also includes the time until the first channel assessment can be made after changing to receive).

The protocol will be described using timed-automata like notation. States are represented as vertices and transitions are represented as edges. An edge is described by its guard (a condition which has to be true in order for the protocol to make the transition) and an update (an action that occurs when the transition is made). In figures, "/" separates the guards and the updates; the guards are before "/" and the update is after. Let "=" denote test for equality and ":=" denote assignment to a variable. When a timeout transition is enabled, it occurs immediately. The corresponding update of that transition and a continuing path of enabled transitions occur at most L time units later. Intuitively, L represents the delay due to executing on a finite-speed processor.

III. A Dominance Protocol for Wireless Medium Access

In the proposed dominance protocol, when messages contend for the channel, a conflict resolution phase, similar to the dominance/binary countdown arbitration, is performed. In our protocol, this conflict resolution phase is named *tournament*. During the tournament, nodes transmit the priority of the message contending for the medium bit-by-bit. But, wireless transceivers can hardly be transmitting and receiving at the same time. Thus, when the transmitted bit is dominant there is no need to sense the medium, whereas, when the bit to transmit is recessive, nothing has to be effectively sent, instead only the medium state has to be sensed.

In this protocol, a bit of the tournament is different from a data bit. Each bit in the tournament has a fixed duration of time, which is considerably longer than a data bit. But, when a node wins the access to the medium, it may transmit at the full bit rate allowed by the specific radio transceiver.

Fig. 2 depicts the three main phases of the protocol: *synchronization*, *tournament* and *receive/transmit* phases. Nodes have to agree on a common reference point in time. This phase is called *synchronization* and happens before every collision resolution phase (named *tournament*). After the tournament, nodes enter into the *receive/transmit* phase.

The following sections describe the protocol. Section A presents a detailed view of it. The rationale for the design of the protocol is addressed in Section B.

A. *Details of the Protocol*

The protocol is formally presented in Fig. 2, using timed-automata like notation. Note, however, that the actual behaviour is slightly different due to clock imperfection, time-of-flight of the carrier-signal and delays in the transitions. States are numbered from 0 to 10. State 0 is the initial state. Associated to each node the following variables are considered: a clock `x`; an integer `i` within the range 0..*npriobits*−1; an integer `prio` occupying *npriobits* bits; an integer `winner_prio` occupying *npriobits* bits and a boolean variable `WINNER`. Let `winner_prio[i]` denote the bit `i` in the variable `winner_prio`, and analogously for `prio[i]`.

Seven functions can be called in a node: `initRadio()`; `setRadioDataRxMode()`; `setRadioDataTxMode()`; `carrierOn()`; `carrierOff()`; `setCarrierSenseOn()`; `setCarrierSenseOff()` and `dequeueHPMsg()`. The function `initRadio()` is used to perform any initialization on the radio chip and to set it into a known starting state. The function `setRadioDataRxMode()` prepares the radio to receive a data packet. `setRadioDataTxMode()` sets the radio to packet transmission mode. The function `carrierOn()` starts transmitting a carrier wave and continues doing so until the function `carrierOff()` is called. Function `setCarrierSenseOn()` is used to set the radio into receive and start detecting carrier pulses, while `setCarrierSenseOff()` is called to stop detecting carrier pulses. To get the highest-priority message from the local queue of message requests, a node calls `dequeueHPMsg()`. The symbol "`carrier?`" is used with the following meaning: sense for a carrier and if there is a carrier then "`carrier?`" is true. Several different timeout values are used. These timeouts (F, G, H, ETG, E and SWX) are constants, and their values and meaning will be defined and reasoned out later.

States 1-4 in Fig. 2 establish a common reference point in time between all nodes that request to transmit. In State 1, nodes wait for a long period of silence (F) such that no node disrupts an ongoing tournament. Then, nodes with a pending message perform transition 2→3 after E time units. This design is such that the duration of E encompasses possible clock differences between the nodes and guarantees that all nodes have time to listen for F time units of silence. Nodes that take 2→3 start sending a carrier pulse that signals the start of a tournament and establishes a common time reference. Other nodes may take one of the two following sequence of state transitions: (i) a node is in State 2 with pending messages and it did not hear a carrier for E time units, and so it makes the transition 2→3; or (ii) a node in State 2 (either because it is waiting to make transition 2→3, or it does not have any pending messages) detects the carrier pulse being sent by other nodes and performs transition 2→4. Nodes making transition 2→4 reset their timers. However, nodes making transition 2→3 wait *SWX* time units to reset their timers because only at that time the carrier pulse is actually transmitted. And then stay in State 4 sending the synchronization carrier. Once in State 4, nodes make transition 4→5 after H time units. At this point, the synchronization ends with nodes resetting their timers.

The States 5-7 relate to the actual tournament. During the tournament, if a node loses the contention of a bit, then it will

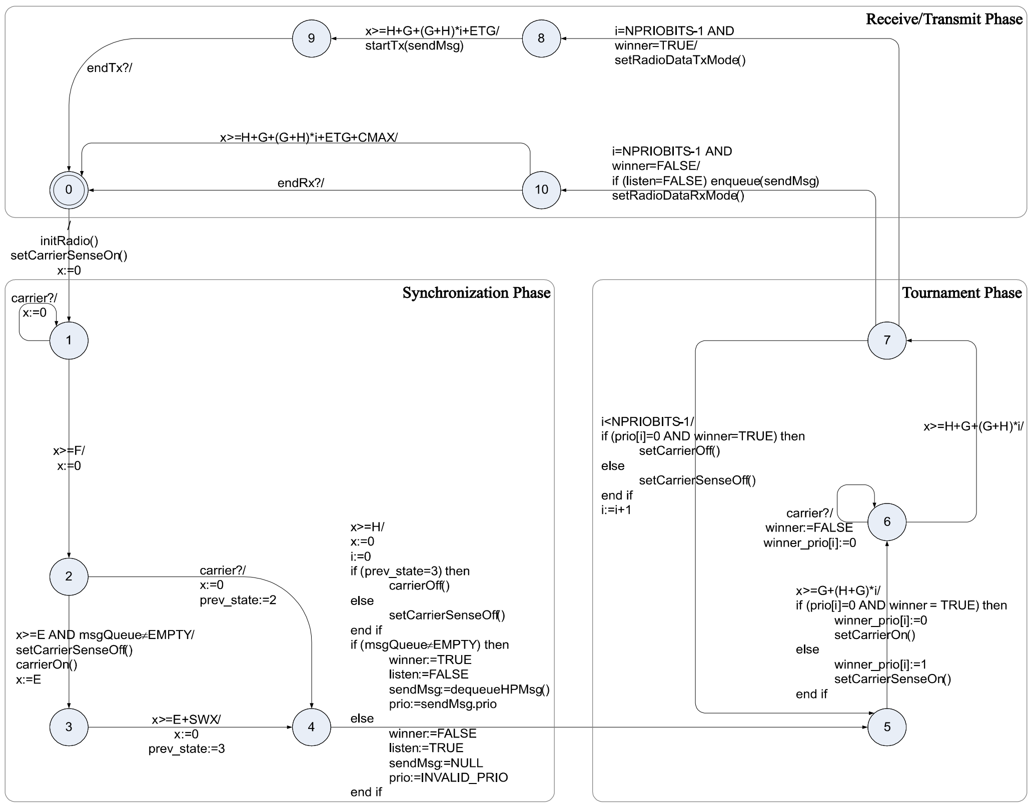

Fig. 2. Details of the WiDom Protocol.

only proceed listening to find out which priority (also message identifier) wins the tournament. If a node does not lose the contention during this bit, it continues with the contention for the next bit. If the node contends with a dominant bit ("0") then it starts transmitting a pulse of the carrier in transition 5→6. If the node contends with a recessive bit ("1"), then in transition 5→6 the node starts performing carrier sensing. While at State 6, if a node contended with a recessive bit ("1") but heard a carrier wave, it has lost.

After the tournament, the winning node (and there is only one winner of the tournament) makes the transition to State 8, waits for a while so that the radios of the other nodes can go into receive mode and then, at State 9, transmits the data part of the message. Then, it goes back to State 0.

Consider now a node which has lost the tournament. The node continues in the tournament and if it has a recessive bit, then it acts in the same way as if it had not lost. The reason for this is that with a recessive bit it just listens; it does not transmit a carrier wave. However, if a node has a dominant bit and it has lost (the boolean variable `WINNER` is `FALSE`), then the protocol acts differently from the case when it had won; no carrier wave is transmitted. After the end of the tournament, the node goes to State 10 waiting to receive the message or timeout.

A node only receiving acts like a node losing the tournament from the start because the variable `WINNER` is assigned `FALSE` before the tournament (transition 4→5).

In order to understand the timeout parameters *F*, *G, H, ETG* and *E*, let us consider the activity of N_1 in Fig. 3b. N_1 enters State 1 (denoted in Fig. 3b with the symbol ①) at time t_1. From this time instant on, node N_1 starts monitoring the medium until it detects the initial idle time period, denoted by *F*. Every time N_1 sends or tries to detect a carrier, it does so for *H* time units, representing the duration of a pulse of the carrier wave. The "guarding" time interval to separate pulses of carrier waves is denoted by *G*. This "guarding" time interval makes the protocol robust against clock inaccuracies, and takes into account that signals need a non-zero time to propagate from one node to another. *ETG* is the gap that a winner must introduce at the end of the tournament. Finally, *E* is a timeout used to improve the reliability introduced by imperfections imposed by the hardware during the synchronization (such as clock inaccuracies and transmit/receive switching times).

Consider the automaton in Fig. 2 again. Traverse the path of the transitions of the winning node and observe the last timeout (the transition 8→9). Based on this, one can compute the transmission time of a message taking the overhead of the protocol into account. The time to transmit a message and perform the tournament when nodes are already synchronized is denoted as $C_i^{'}$, and is given by:

$$\begin{aligned} C_i^{'} &= C_i + 2H + G + (G+H)\times(npriobits-1) + \\ & ETG + E + \max\{TFCS, SWX\} + 2L \end{aligned} \quad (1)$$

where C_i denotes the time required to transmit a message from message stream τ_i. The time to transmit a message and perform the tournament when nodes are not yet synchronized is denoted $C_i^{''}$, and takes into account the initial idle time:

$$C_i^{''} = C_i^{'} + F \quad (2)$$

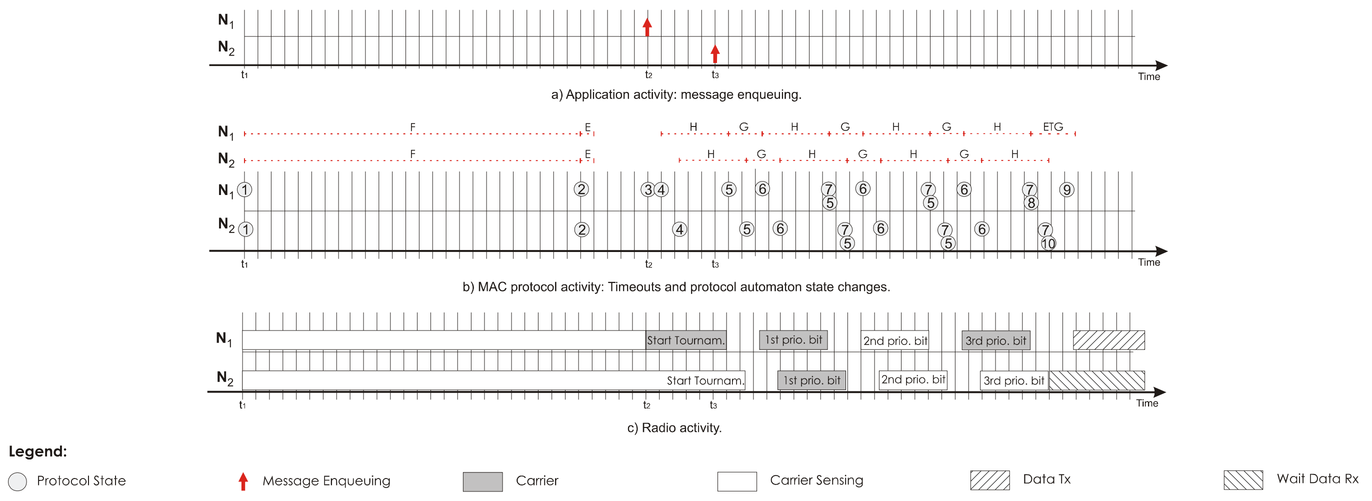

Fig. 3. Application (a), MAC protocol (b) and Radio (c) activity example in nodes N_1 and N_2. In this example it is assumed that *npriobits* = 3; the priority of the message queued at N_1 is '010' and the message queued at N_2 has a priority '011'. Before time t_1, the medium was busy; at this time both nodes start detecting an idle medium, so they enter State 1 (a). N_1 queues a message request at time t_2 and N_2 queues a message request at time t_3.

B. *Rationale of the design and correctness*

In this section we discuss the correctness of the protocol and demonstrate how assigning values to the constants *E*, *F*, *G*, *H*, *ETG, TFCS* and *SWX* affect the correctness. The protocol must satisfy the following relevant properties.

- Mutual Exclusion. At any given time, at most one computer node can be in State 9.
- Progress. There are two types of progress: (i) if a computer node is backlogged then State 0 is reached after at most C_i'' time units from any state; and (ii) if a message finishes transmission and there exists a backlogged node then one message of the backlogged nodes should be transmitted next.
- Prioritization. Of all nodes which were backlogged, the one that will transmit a message is the one that dequeues (at transition 4→5) the message with the highest priority.

These properties hold if the constraints corresponding to inequalities (3)-(7) below are satisfied.

When a node transmits a dominant bit in iteration `i` in the tournament, it is received by all other nodes and it is perceived to be received in iteration `i`.

Implications: Consider an iteration of the tournament. It must have been sufficient overlap between the time where one node transmits the carrier to inform that it has a dominant bit and the time interval where a node with a recessive bit listens for nodes with a dominant bit. Due to clock drift and inaccuracy of synchronisation, this overlap becomes smaller and smaller with the iterations within the tournament. Hence, the last iteration (the worst-case scenario) of the tournament is considered. Therefore, we derive the following constraint:

$$\begin{aligned}&[H+G+(H+G)\times(npriobits-1)]\times[1-\varepsilon]-\\&[G+(H+G)\times(npriobits-1)]\times[1+\varepsilon]-\\&2CLK-L-2\alpha-(SWX+E)>TFCS\end{aligned}\tag{3}$$

Inequality (3) guarantees that even in the presence of worst-case clock inaccuracies, all nodes will hear a dominant bit for at least the time necessary to detect a carrier (*TFCS*).

If a node N_i has perceived silence long enough (*F* time units) to make transition from State 1 to State 2 but other nodes perceive the duration of silence to be less than *F*, due to different time-of-flights and clock-imperfections, then node N_i needs to wait until all nodes detected this long time of silence.

Implications: The protocol must stay in State 2 for *E* time units to ensure this, and the following constraint is derived:

$$2CLK+L+2\alpha+F\times 2\varepsilon+SWX<E \tag{4}$$

With similar reasoning as for (4), a node which has won the tournament must wait *ETG* time units before transmission (this occurs in 8→9) to ensure that all losing nodes reached State 10.

Implications: *ETG* must satisfy the following constraint:

$$\begin{aligned}&2CLK+L+2\alpha+(H+G+(G+H)\times(npriobits-1))\times 2\varepsilon+\\&(SWX+E)<ETG\end{aligned}\tag{5}$$

During the tournament, the maximum time interval of idle time should be less than *F*, the initial idle period.

Implications: This assures that if one node makes the transition from State 1 to State 2 (the initial idle time period) then all nodes will do it at most *E* time units later. Therefore, we have the following constraint:

$$\begin{aligned}&[H+G+(H+G)\times(npriobits-1)+ETG]\times[1-\varepsilon]-\\&[H+G]\times[1+\varepsilon]+2CLK+L+2\alpha<F\end{aligned}\tag{6}$$

Finally, the time interval between two successive dominant bits must assure that bits are interpreted correctly.

Implications: The worst-case scenario occurs when these two bits are the last ones in the tournament. Therefore, the following constraint must also be satisfied:

$$\begin{aligned}&[H+2G+(H+G)\times(npriobits-2)]\times\\&[1-\varepsilon]-[H+G+(H+G)\times(npriobits-2)]\times[1+\varepsilon]-\\&2CLK-L-2\alpha-(SWX+E)>0\end{aligned}\tag{7}$$

The values of *E*, *F*, *G*, *H*, *ETG*, *TFCS* and *SWX* must be selected such that they satisfy constraints (3)-(7). The selection of *TFCS* and *SWX* is imposed by the platform chosen. Section IV instantiates these timeouts for a concrete platform.

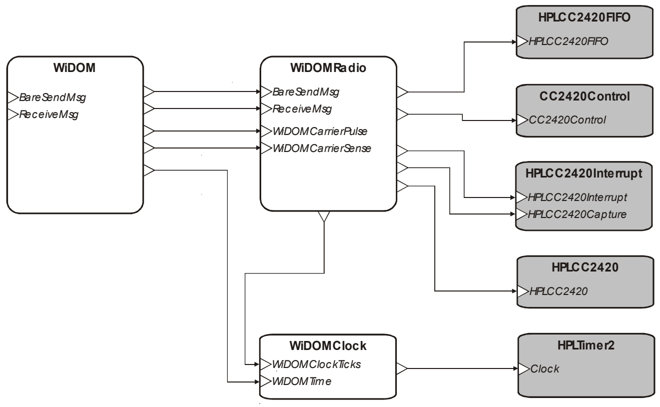

Fig. 4. TinyOS component assembly. Rectangles are implementation modules of components. For the sake of simplicity, only the most relevant modules and interfaces are depicted, and configurations wiring components are omitted. Shaded rectangles are TinyOS components reused in the implementation. Triangles pointing into a rectangle are provided interfaces. Triangles pointing out represent used interfaces. The names of the provided interfaces are in italics.

IV. Implementation

There are a number of difficulties in implementing a wireless dominance protocol. There exist priority levels for which the protocol needs to switch between transmit and receive modes for every priority bit. Many transceivers are not designed for frequent switching, and hence every switching takes a non-negligible amount of time. It is also well known that wireless channels typically have significantly higher noise levels than wired channels, and that detection of pulses of short duration is difficult [6]. Therefore, the demonstration of an actual implementation of dominance protocols for wireless medium is significant. This implementation is named WiDom.

A. The Platform

WiDom was developed for an embedded computer platform known as MicaZ [7]. It is a sensor network platform offering a low power microcontroller, 128 Kbytes of program flash memory and an IEEE 802.15.4 compliant radio transceiver CC2420 [8] capable of 250 kbps data rate. The MicaZ platform is supported by TinyOS, an open-source operating system designed for wireless sensor networks. This platform turned out to be an attractive alternative for the implementation because of the following relevant characteristics: (i) it allows replacing the existing MAC protocol in TinyOS easily; (ii) the available timers are sufficiently precise; (iii) the radio can be put into a specific test mode, where it is possible to transmit a non-modulated carrier for an arbitrary duration; (iv) the radio has built-in RSSI (Receive Signal Strength Indicator)/energy detection functionality and Clear Channel Assessment (CCA) is available through a digital output pin; (v) the spread spectrum modulation used to transmit data messages makes them resistant to noise. Due to (v), the main factor that affects message transmission reliability is collisions.

Dominance protocols in wired media require that a node can simultaneously transmit while it detects the transmissions from other nodes. Unfortunately, this is not possible in most radio transceivers, including the CC2420, because the transmitted energy is much higher than the received energy. For this reason, the CC2420 can only be either in transmission mode or in reception mode, and it can take up to 192 μs to switch between these two modes.

The CC2420 radio can be set into a transmitter test mode to either transmit a modulated carrier or a non-modulated carrier wave. The RSSI obtained with a non-modulated carrier is 9dBm stronger than the one obtained when transmitting a modulated carrier [8]. Hence, the non-modulated carrier is used for transmitting the carrier waves during the tournament.

It is also necessary to detect when other nodes transmit a carrier wave. For this, the CC2420 support for CCA is used. The CCA functionality of the CC2420 radio computes the average RSSI over the last 128 μs. This average is compared to a configurable threshold and then CC2420 sets the CCA output pin accordingly. This pin is sampled by our software communication stack to detect if other nodes are sending carrier pulses. Every time the radio is set into receive mode, it takes at least 128 μs to make the first valid CCA operation.

The proposed protocol is heavily dependent on timers. The MicaZ's ATmega128 microcontroller provides two 8-bit timer/counter and two 16-bit timers. The 8-bit Timer/Counter2 provides timing for the protocol implementation since this is the timer used in the CC2420 TinyOS communication stack, which we are partially replacing.

B. Protocol Implementation Software Description

The main TinyOS software components of the implementation are presented in Fig. 4, which also provides a simplified overview of the implementation component assembly. In Fig. 4, components `CC2420Control`, `HPLCC2420FIFO`, `HPLCC2420Interrupt` and `HPLCC2420`, are part of the Hardware Presentation Layer (HPL) for the CC2420, and are regular TinyOS components reused by the implementation. These components provide basic functionalities for handling the radio.

The component `WiDomClock` uses `HPLTimer2` (also an HPL component component from TinyOS) to drive the timing of the protocol. `WiDomClock` configures the timer prescaler to deliver the timer interrupts every 34.722 μs, which is used to drive the timing maintained by this component. For this reason, when timeouts are selected, these will be multiples of 34.722 μs.

The `WiDomRadio` component is responsible for providing all radio functionalities needed by the MAC protocol. It handles the interactions with the HPLs for sending/receiving packets, and it also provides the `WiDom` component with a simple send/receive interface (interfaces `BareSendMsg` and `ReceiveMsg` are commonly used TinyOS interfaces for these purposes).

Finally, the `WiDom` component implements the WiDom protocol following closely the protocol's specification as depicted in Fig. 2.

V. Response-time Calculations

Let us now introduce the messages' response-time calculations for the WiDom protocol. This analysis is based on a recently developed analysis for the CAN bus [3], which in turn builds upon existing analysis for non-preemptive static-priority scheduling [9]. But subtleties about the synchronization in the protocol require our schedulability analysis to deal with aspects that are not dealt with in [3].

Firstly, the timeout parameters of the protocol (introduced in Fig. 2) must be selected according to the constraints given in Section III. We choose *npriobits* = 10. The platform described in Section IV gives us: $CLK = 34.722$ µs, $L = 5$ µs, $\alpha = 1$µs and $\varepsilon = 10^{-5}$. A choice of parameters that satisfies (3)-(7) is then: $E = 312$ µs; $F = 24409$ µs; $G = 729$ µs; $ETG = 555$ µs; $H = 1562$ µs. The timeout parameters are based on the assumption that a carrier pulse must have a duration (timeout previously denoted as *TFCS*) of 486 µs so that the other nodes may detect it, and $SWX = 347$ µs (this is the time to switch between transmission/reception mode – 192 µs, added to the time until the first CCA operation; rounded up to a multiple of 34.722 µs). The value of *TFCS* was experimentally obtained.

Recall the sporadic model [10] with a system of n message streams: $\tau_1, \tau_2, \ldots, \tau_n$. Each message stream τ_i is characterized by T_i, D_i and C_i. Let us now compute C_i, C_i' and C_i''. Assume that the data length of messages is 64 bytes (one byte for the length of data in a packet is included). Adding 3 bytes used for preamble and 1 byte for the start frame delimiter, the time to transmit a message is then given by (bit rate of 250 Kbits/s for data transmission):

$$\forall i \in \{1..n\} : C_i = (64+3+1) \times 8 \times \frac{1}{250000} = 2176 \text{ µs} \tag{8}$$

Applying Equation (1), yields:

$$\forall i \in \{1..n\} : C_i' = 28011 \, \mu s \tag{9}$$

and from (2) it will result that:

$$\forall i \in \{1..n\} : C_i'' = 52420 \, \mu s \tag{10}$$

We will now address the formulation of the response time calculations. Assume that the release jitter is zero. The granularity of the time is $Q_{bit} = 4/250000 = 16$ µs, because the radio uses Direct-Sequence Spread-Spectrum such that every 4 bits is modulated as 32 chips and the data rate is 250 Kbits/s (this is equivalent to 2 Mchip/s). Using these assumptions, the response time is (similar to [9]) as follows:

$$R_i = \max_{q=0..Q} \left\{ w_{i,q} + C_i'' - q \times T_i \right\} \tag{11}$$

where $q = \lfloor L_i/T_i \rfloor - 1$. L_i is the length of the longest level-i busy period in non-preemptive context, which is given by the smallest positive integer L_i satisfying:

$$L_i = \max_{j \in lp(i)} \left\{ C_j' - Q_{bit} \right\} + \sum_{j \in hp(i) \cup i} \left\lceil \frac{L_i}{T_j} \right\rceil \times C_j'' \tag{12}$$

where $hp(i)$ is the set of all message streams with a higher priority than τ_i, and, similarly, $lp(i)$ is the set of all message streams with a lower priority than τ_i. The waiting time $w_{i,q}$ is then:

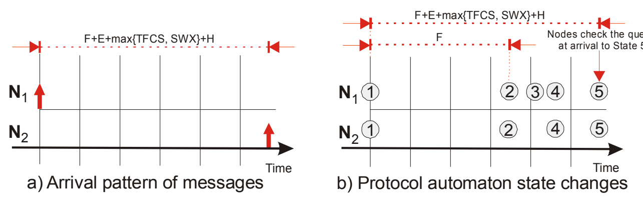
F+E+max{TFCS, SWX}+H
N1
N2
Time
a) Arrival pattern of messages
F+E+max{TFCS, SWX}+H
F
Nodes check the queue at arrival to State 5
N1
N2
1
2
3
4
5
Time
b) Protocol automaton state changes

Fig. 5. Worst-case arrival pattern and protocol automata state changes. (b) shows two computer nodes N_1 and N_2 and how their states change as time progresses. Let us assume that there is a message stream τ_1 on N_1 and a message stream τ_2 on N_2 and that τ_1 has higher priority than τ_2. (a) depicts the arrival pattern that maximizes the response time of τ_2. Observe that nodes check the queue of message transmission requests when they arrive to State 5.

$$w_{i,q} = q \times C_i'' + \sum_{j \in hp(i)} \left(\left(\left\lfloor \frac{1 + w_{i,q} + F + E + \max\{TFCS, SWX\} + H + Q_{bit}}{T_j} \right\rfloor \right) \times C_j'' \right) + \max_{k \in lp(i)} \left\{ C_k' - Q_{bit} \right\} \tag{13}$$

Note that the analysis considers the initial idle time between States 1-5 (Fig. 2) to be part of the "message" when computing the interference. This initial idle period should not be included when computing the blocking. Thus the last term on the right hand side of (13) uses C_k' instead of C_k''. Observe that (13) differs from the analysis used in the CAN bus. With WiDom, it is necessary to add the time before the next message is dequeued after the previous message has been transmitted: $F + E + \max\{TFCS, SWX\} + H$. Fig. 5 provides further intuition.

Let us apply the response time analysis to calculate the values according to (8)-(13) in the following example (the response times will be tested empirically in Section VI).

Example 1. Consider $m = 10$ computer nodes with one message stream on each node. Message streams are given periods as shown in Table 1 (all values are in µs).

Deadline monotonic is used to assign priorities, and $D_i = T_i$. It can be seen that the difference between the greatest T_i and the smallest T_i is very large; this is intentional because it is exactly for such workloads that prioritization is crucial in order to meet deadlines. Applying (8)-(13) results in the response times as shown in Table 1. Observe (Table 1) that the scheduling theory predicts that all deadlines will be met because we obtain $\forall i : R_i \leq T_i$.

VI. Experimental Evaluation

The implementation reported in the previous section enabled conducting a performance evaluation of the protocol using real-world platforms. For this experimental evaluation, we advance the following hypotheses:

1. the implementation of the protocol offers collision-free medium access for data messages;
2. the implementation of the protocol offers prioritized medium access;
3. the response-time analysis equations in (8)-(13) can be used to analyze the response-times of the implementation of the protocol.

TABLE I
MESSAGE STREAMS FOR EXAMPLE 1

i	1	2	3	4	5
T_i (μs)	256 000	512 000	1024 000	2 048 000	4 096 000
C_i (μs)	2 176	2 176	2 176	2 176	2 176
$C_i^{'}$ (μs)	28 011	28 011	28 011	28 011	28 011
$C_i^{''}$ (μs)	52 420	52 420	52 420	52 420	52 420
R_i (μs)	80 415	132 835	185 255	237 675	342 515
i	6	7	8	9	10
T_i (μs)	8 192 000	16 384 000	32 768 000	32 768 000	32 768 000
C_i (μs)	2 176	2 176	2 176	2 176	2 176
$C_i^{'}$ (μs)	28 011	28 011	28 011	28 011	28 011
$C_i^{''}$ (μs)	52 420	52 420	52 420	52 420	52 420
R_i (μs)	394 935	447 355	499 775	709 455	733 880

TABLE II
RESPONSE TIMES OBTAINED EXPERIMENTALLY FOR SETTINGS OF EXAMPLE 1.

	i	1	2	3	4	5
Periodic	Min r_i *(μs)*	25 752	25 787	25 926	27 002	26 620
	Avg r_i *(μs)*	34 363	35 891	35 822	46 551	48 322
	Max r_i *(μs)*	118 738	131 932	131 793	182 105	184 987
	Dead. miss prob.	0.007%	0.000%	0.000%	0.000%	0.000%
Sporadic	Min r_i *(μs)*	25 752	25 752	25 891	26 065	27 627
	Avg r_i *(μs)*	33 079	36 829	38 460	43 252	45 509
	Max r_i *(μs)*	97 210	168 807	184 814	218 980	240 334
	Dead. miss prob.	0.014%	0.009%	0.000%	0.000%	0.000%
	i	6	7	8	9	10
Periodic	Min r_i *(μs)*	25 891	27 627	27 627	27 627	27 627
	Avg r_i *(μs)*	87 523	48 738	47 662	57 662	47 627
	Max r_i *(μs)*	187 071	181 029	184 328	261 063	179 536
	Dead. miss prob.	0.000%	0.000%	0.000%	0.000%	0.000%
Sporadic	Min r_i *(μs)*	27 627	27 627	27 627	27 627	27 627
	Avg r_i *(μs)*	47 349	47 349	48 565	44 954	44 190
	Max r_i *(μs)*	240 056	240 056	224 258	214 848	235 786
	Dead. miss prob.	0.000%	0.000%	0.000%	0.000%	0.000%

§Hypothesis 1 and Hypothesis 2. In order to test Hypothesis 1 and Hypothesis 2, four experiments were conducted. Let d denote the maximum distance between any two nodes. A set of m nodes were positioned in a circle such that for every node the distance to its neighbours with the minimum distance is maximized (this placement was selected only for the convenience of the experiment's description). The experiment runs as follows. A special node (which is not included in the m nodes) transmits a carrier wave and all other nodes boot. All nodes request to transmit a message and they enter State 1 (refer to Fig. 2). These nodes stay in State 1 until the special node stops transmitting the carrier. The experiment was performed for $m = 2$ and $m = 10$. For the case of $m = 2$, all nodes make a new request to transmit a message using `random`(0, 255) ms (this means generate a uniformly distributed random number with a minimum value of 0 and maximum of 255) after the previous message request. For the case $m = 10$, all nodes make a new request to transmit a message `random`(0,1023) ms time units after the previous request. The diameter was also varied; one experiment had $d = 1$ m and another $d = 4$ m. Nodes are given IDs from 1 to 10 and their priority is equal to the ID. All messages have 64 bytes.

Every node has a sequence counter, initialized to 1. The sequence counter is transmitted in every message and then the sequence counter on the node is incremented. Whenever a node received a message, it compares the received sequence counter to the sequence number previously received from the same node. The difference between the received sequence counter and the previously stored, allowed checking if transmissions were collision-free. Since a collision causes a lost message, this gives us an upper bound on the number of collisions.

Prioritization was also tested. This was done as follows. When a node sends a message it sends its priority in the data packet. All nodes receive this packet (if they did not receive, it would be considered as a collision, see Hypothesis 1) and if the priority of the winner was less than the priority of this node then it is considered as a prioritization error.

The results of the experiments are presented in Fig. 6. More than 99.998% of all messages were collision-free and prioritized. This corroborates Hypotheses 1 and 2.

§Hypothesis 3. In order to test Hypothesis 3, two experiments using parameters from Table 1 were conducted. The only difference between experiments was that message streams were strictly periodic in one experiment while, in the other, message streams were sporadic so a node with a message stream τ_i made a new transmission request $T_i +$ `random`$(0{,}5 \times T_i)$ ms after the previous request. Every node had one message stream, thus $n = m$.

The experimental setup was as follows. In each node, messages were requested to be sent periodically or sporadically, depending on the experiment. Nodes count the time since a message is requested to send until it is actually transmitted. This time (Q_i, the queuing time of the message) is sent in the data payload of the packet. A special node (not included in the m nodes) receives the messages, gets the queuing time in the data payload and calculates the response time. From the reasoning in Section V, the response time is:

$$R_i = Q_i + C_i \tag{14}$$

The experiments were run for the message streams in Example 1, until 100 000 messages were transmitted. The results obtained with this experiment are shown in Table 2 (all values are given in μs). The measured response time is denoted by r_i. Only a small number of the maximum response times are above the calculated upper bounds. The cause of these was investigated experimentally, in our embedded computer platforms. It was found that nodes perceived noise when waiting for F time units of silence (transition 1→2 in Fig. 2).

Several modifications to the protocol can be introduced to make the protocol more resilient to noise. The transmit power during the tournament could be increased and the sensitivity of receivers decreased in order to make them less susceptible to noise, or enforce that a receiver must have detected a carrier pulse of a given duration in order to perceive it as a detected carrier, such that it is ensured that short spikes of noise are not mistaken as being a carrier. These modifications to the protocol imply the study of tradeoffs regarding the resilience of the protocol and the target operating environmental conditions. This however, is out of the scope of this paper. Nonetheless, we can see that WiDom is designed to perform well in the presence of noise: a node waiting for F units of idle channel may need to restart its waiting due to noise and noise can cause a node to perceive a dominant bit when there is only recessive bits but these scenarios do not cause a collision. In particular, we note that if a node experiences strong noise for several seconds, then the protocol will simply lose its tournament and not start any new ones during the duration of noise and then the noise is over, WiDom operates normally, attempting to send the messages that was never transmitted during the duration of strong noise.

Experiments conducted in a semi-industrial environment [11] led us to conclude that the main source of noise to the protocol were other wireless devices operating in the same frequency (for example IEEE 802.11 devices), and not the industrial machinery itself (e.g. DC motors and variable-frequency drives). We also note that, to further improve the robustness of wireless communication, proper planning of frequencies and node positioning within the factory-floor is necessary. Such approach is common practice today, and tools for conducting such studies are available. Nonetheless, the deadline miss probability provides evidence that the protocol performs effectively and that this experiment corroborates Hypothesis 3.

VII. Related Work

The introduction of the wireless LAN standard IEEE 802.11 [12] stimulated the development of many prioritized Carrier Sense Multiple Access (CSMA) protocols. Some of these protocols [13-15] changed parameters in the IEEE 802.11 standard to be a function of deadlines. These techniques have two drawbacks: (i) they only approximate priority scheduling; it may happen that a high-priority message has to wait for one or many lower-priority messages; and (ii) collisions can occur, hence causing deadline misses. Other prioritization protocols based on IEEE 802.11 use *black-bursts* [16-18]. Black-bursts work as follows. If the channel is idle then a node transmits a message immediately. Otherwise the node waits until the channel becomes idle and transmits a black-burst (a jamming signal) for a time duration which is proportional to the priority. When a node finishes transmitting its jamming signal, the node listens to find out whether other nodes transmit a jamming signal or not. If so, the node did not have the highest priority and so it waits until the channel is idle again. All these black-burst schemes [16-18] have the drawback that the maximum length of the black-burst is proportional to the number of priority levels. Therefore, only a small number of priority levels can be supported. Another technique [19], not based on IEEE 802.11, is to implement prioritization using two separate narrow band busy-tones to communicate that a node is backlogged with a high-priority message. This technique has the drawback of requiring specialized hardware, requires extra bandwidth (for the narrow band signals) and it supports only two priority levels.

The IEEE 802.11 standard also defined another MAC protocol where a base station polls a node, and gives it the right to transmit in a time interval. Naturally such an approach is inefficient to schedule sporadic messages. Recently, the IEEE 802.11e profile was introduced with the intention of offering better support for Quality-of-Service. The previous proposed approaches [13-15] of choosing back-off times as a function of priorities were adopted, and the polling scheme in IEEE 802.11 was refined with traffic classes.

In [20], a MAC protocol based on a binary countdown was proposed. However, the binary countdown arbitration was employed such that collisions can cause deadline misses.

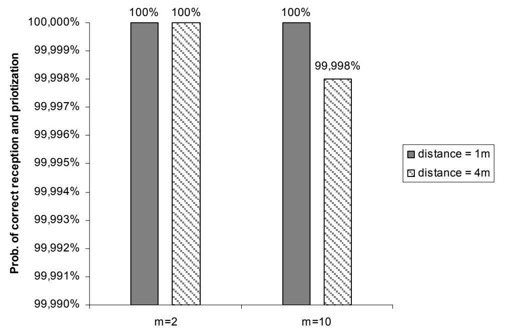

Fig. 6. Prioritization and collision free test results. Each experiment was performed until a total of 100 000 messages were sent.

MAC protocols have also been proposed from the real-time systems community with the goal of meeting deadlines. Some protocols use tables (sometimes called Time-Division Multiple-Access (TDMA) templates) with explicit start times of message transmission. These tables are created at run-time in a distributed fashion [21] or by a leader [22]. It is also conceivable to use a TDMA template designed before run-time [23] and use it to schedule wireless traffic. However, all these time-table approaches have the drawback of requiring that sporadic message streams are dealt with using polling, which, as previously stated, is inefficient. Another approach, Implicit EDF [24], is based on the assumption that all nodes know the traffic on the other nodes that compete for the medium, and all these nodes execute the EDF scheduling algorithm. Unfortunately, this algorithm is based on the assumption that a node knows the *arrival time* of messages on other nodes, and this implies that polling must be used to deal with sporadic message streams. The conclusion of this section is that several prioritization protocols and real-time scheduling algorithms exist, but they do not efficiently solve the problem of sporadic scheduling in wireless networks.

VIII. Conclusion

Millions of CAN controllers are deployed and it is being used pervasively in automotive electronics and factory automation due to its flexible MAC protocol, large number of priorities and its ability to schedule sporadic message streams with real-time requirements. No equivalent to this MAC protocol was available for the wireless domain. In this paper we have presented such a MAC protocol for wireless channels. The protocol is collision-free, does not require synchronized clocks and supports a large number of priority levels. The proposed wireless dominance protocol can be implemented and it allows devising a set of analysis appropriate for the pre-run-time computation of the response times. It is intended for short-range communication and for this purpose our protocol is reliable, even in industrial environments.

We stress that the overhead introduced by the protocol is, to a large extent, due to the transition time between transmission and reception. This is a technological limitation that can be minored with better hardware, as witnessed by the fact that the HIPERLAN standard [25] requires a switching time of only

2μs. If such transceivers were available, the overhead could be reduced by two orders of magnitude.

A protocol providing an upper bound on the queuing times of messages is naturally useful for supporting scheduling of real-time traffic. For unicast communication, acknowledgements and retransmissions can be introduced without any modification to the protocol. The protocol, parameters and the response-time analysis presented in Section V are easily extendable to consider the time for a receiver to send an acknowledgement. Similarly, a technique like sending several replicas of the same message after the tournament can be introduced to improve reliability. It is even possible to have a stochastic approach to model faults similarly to previous results on CAN [26]. Moreover, bus-guardian-like solutions for containment of *babbling-idiots* in wired networks [27] can be adapted to our protocol.

Acknowledgment

The authors would like to show their appreciation for the useful comments provided by the anonymous reviewers and editors, which we believe helped us very much in improving the technical quality of this manuscript.

References

[1] Bosch, "CAN Specification, ver. 2.0, Robert Bosch GmbH, Stuttgart", 1991, online at: http://www.semiconductors.bosch.de/pdf/can2spec.pdf.

[2] K. Tindell, H. Hansson, and A. Wellings, "Analysing real-time communications: controller area network (CAN)", In proc. of the 15th Real-Time Systems Symposium (RTSS'94), pp. 259-263, 1994.

[3] R. J. Bril, J. J. Lukkien, R. I. Davis, and A. Burns, "Message response time analysis for ideal controller area network (CAN) refuted", In proc. of the 5th Int. Work. on Real-Time Net. (RTN'06), pp. 5-10, Dresden, Germany, 2006.

[4] A. K. Mok and S. Ward, "Distributed Broadcast Channel Access", Computer Networks, vol. 3, issue 5, pp. 327-335, 1979.

[5] B. D. Bui, R. Pellizzoni, M. Caccamo, C. F. Cheah, and A. Tzakis, "Soft Real-Time Chains for Multi-Hop Wireless Ad-Hoc Networks", In proc. of the 13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'07) pp. 69-80, Bellevue, WA, USA, 2007.

[6] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution", IEEE Transactions on Communication, vol. 23, issue 12, pp. 1417-1433, 1975.

[7] Crossbow, "MICAz - Wireless Measurement System Product Datasheet", 2005, online at: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

[8] Chipcon, "CC2420 Datasheet", online at: http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf.

[9] L. George, N. Rivierre, and M. Spuri, "Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling", INRIA, Technical Report RR-2966, September 1996, online at: http://www.inria.fr/rrrt/rr-2966.html.

[10] A. Mok, "Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment", in PhD thesis, EECS. Cambridge, Mass.: MIT, 1983.

[11] IPP-HURRAY, "RFieldbus Manufacturing Field Trial website", 2002, online at: http://www.hurray.isep.ipp.pt/activities/rfpilot/.

[12] ISO/IEC, "IEEE 802.11, IEEE Standards for Inf. Tech. -- Telecom. and Infor. Exchange between Systems -- Local and Metro. Area Net. -- Specific Req. -- Part 11: WLAN MAC and PHY Spec." 1999.

[13] I. Aad and C. Castelluccia, "Differentiation Mechanisms for IEEE 802.11", In proc. of the 20th Joint Conf. of the IEEE Comp. and Comm. Soc. (INFOCOM'01), pp. 209-218, 2001.

[14] M. Barry, A. T. Campbell, and A. Veres, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks", In proc. of the 20th Annual Joint Conf. of the IEEE Comp. and Comm. Soc. (INFOCOM'01), pp. 582-590, Anchorage, AK, USA, 2001.

[15] D.-J. Deng and C. Ruay-Shiung, "A Priority Scheme for IEEE 802.11 DCF Access Method", IEICE Trans. on Communication, vol. E82-B, pp. 96-102, 1999.

[16] J.-P. Sheu, C.-H. Liu, S.-L. Wu, and Y.-C. Tseng, "A priority MAC protocol to support real-time traffic in ad hoc networks", Wireless networks, vol. 10, issue 1, pp. 61-69, 2004.

[17] J. L. Sobrinho and A. S. Krishnakumar, "Real-time traffic over the IEEE 802.11 medium access control layer", Bell Labs Technical Journal, vol. 1, issue 2, pp. 172-187, 1996.

[18] J. L. Sobrinho and A. S. Krishnakumar, "Quality-of-Service in ad hoc carrier sense multiple access networks." IEEE Journal on Selected Areas of Comm., vol. 17, issue 8, pp. 1353-1368, 1999.

[19] X. Yang and N. H. Vaidya, "Priority Scheduling in Wireless Ad Hoc Networks", In proc. of the 3rd ACM Int. Symp. on Mobile ad hoc net. & comp. (MobiHoc'02), pp. 71-79, Lausanne, Switzerland, 2002.

[20] H. Wu, A. Utgikar, and N.-F. Tzeng, "SYN-MAC: A Distributed Medium Access Control Protocol for Synchronized Wireless Networks", Mobile Networks and Applications (MONET), vol. 10, issue 5, pp. 627-637, 2005.

[21] W. C. Thomas, A. B. Moussa, B. Rajeev, and B. S. David "Contention-Free Periodic Message Scheduler Medium Access Control in Wireless Sensor / Actuator Networks", In proc. of the IEEE Real-Time Systems Symposium, pp. 298-307, Cancun, Mexico, 2003.

[22] H. Li, P. Shenoy, and K. Ramamrithan, "Scheduling Communication in Real-Time Sensor Applications", In proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, Canada, 2004.

[23] H. Kopetz and G. Grunsteidl, "TTP-a protocol for fault-tolerant real-time systems", IEEE Computer, vol. 27, issue 1, pp. 14-24, 1994.

[24] M. Caccamo and L. Y. Zhang, "An Implicit Prioritized Access Protocol for Wireless Sensor Networks", In proc. of the 23rd IEEE Real-Time Syst. Symp. (RTSS'02), pp. 39-48, Austin, Texas, 2002.

[25] ETSI, " TS 101 475 V1.3.1:" Broadband Radio Access Networks (BRAN);HIPERLAN Type 2; Physical (PHY) layer.

[26] I. Broster, A. Burns, and G. Rodríguez-Navas, "Probabilistic Analysis of CAN with Faults", In proc. of the 23rd IEEE International Real-Time Systems Symposium (RTSS'02), pp. 269-278, Austin, TX. (USA), 2002.

[27] I. Broster and A. Burns, "An Analysable Bus-Guardian for Event-Triggered Communication", In proc. of the 24th Real-Time Systems Symp. (RTSS'03), pp. 410-419, Cancun, Mexico, 2003.

Nuno Pereira is a researcher at CISTER/IPP-Hurray. He received a licentiate degree in Computer Engineering from the School of Engineering of the Polytechnic Institute of Porto and an MSc Degree from the University of Minho in 2002 and 2005 respectively. Currently he is working towards his PhD. His research interests are in the areas of distributed and embedded systems.

Björn Andersson received his M.Sc. degree at Chalmers University of Technology in Sweden in 1999 andreceived the SNART best master of science thesis award that year. He extended (together with others) static-priority scheduling from uniprocessors to multiprocessors and earned his Ph.D. degree at Chalmers University of Technology. He is currently a visiting scientist at CISTER/IPP-Hurray, exploring real-time communication, real-time scheduling on multiprocessors and data aggregation in cyber-physical computer systems.

Eduardo Tovar received the Licentiate, MSc and PhD degrees in electrical and computer engineering from the University of Porto, Porto, Portugal, in 1990, 1995 and 1999, respectively. Currently he his Professor of Industrial Computer Engineering in the Computer Engineering Department at the Polytechnic Institute of Porto (ISEP-IPP), where he is also engaged in research on real-time distributed systems and factory communications. He heads the CISTER/IPP-Hurray Research Unit (UI 608), a top rated ("Excellent") unit of the FCT Portuguese network of research units. He authored or co-authored more than 80 scientific and technical papers in the area of real-time computing systems and industrial computer engineering.