Static-Priority Scheduling of Sporadic Messages on a Wireless Channel

Björn Andersson and Eduardo Tovar

Department of Computer Engineering, School of Engineering,
Polytechnic Institute of Porto (ISEP-IPP),
Rua Dr. António Bernardino de Almeida 431,
4200-072 Porto, Portugal
{bandersson,emt}@dei.isep.ipp.pt

Abstract. Consider the problem of scheduling sporadic messages with deadlines on a wireless channel. We propose a collision-free medium access control (MAC) protocol which implements static-priority scheduling and present a schedulability analysis technique for the protocol. The MAC protocol allows multiple masters 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.

1 Introduction

The sporadic model [11] has proven to be very useful in the design of real-time systems. In this model, 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 model is supported in processor scheduling [4] (where a message stream is called a task) and in wired communication channels [17]. Wireless communication is of increasing interest in the design of distributed real-time systems, and many scheduling algorithms and analysis techniques for wireless communication are available for periodic messages. But for sporadic messages such results are less well developed. 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 this paper we solve the problem of sporadic scheduling on a wireless channel. We adapt the dominance protocols [12] (used in the CAN bus [5]) to a wireless channel and perform a schedulability analysis. The main idea of our dominance protocol is that a message stream is assigned a static priority and when message streams contend for the channel, they perform a tournament such that the highest-priority message is granted access to the channel. This tournament is performed bit-by-bit, starting with the most significant bit. A bit is assigned a time interval. If a node contends with a dominant bit then a carrier

wave is transmitted in this time interval; if the node contends with a recessive bit, it transmits nothing but listens. This makes it possible for a node with a recessive bit to detect that another node has transmitted a dominant bit, and hence the node with the recessive bit withdraws. In order for this scheme to work, nodes must agree on which time interval to use. This requires a convention, something that is easy to state and which we do. It also requires that nodes have a common reference point in time. We provide this as well.

The remainder of this paper is structured as follows. Section 2 discusses related work and their ability to solve the problem of sporadic messages on a wireless channel. Section 3 presents the system model with our assumptions and terminology. Section 4 presents the protocol and discusses the rationale behind its design. Section 5 presents the schedulability analysis. Finally, Section 6 offers conclusions and future work.

2 Related work

The introduction of the wireless LAN standard IEEE 802.11 [1] stimulated the development of many prioritized Carrier Sense Multiple Access (CSMA) protocols. Some of these protocols [2, 3, 7] changed parameters in the IEEE 802.11 standard to be a function of deadlines, either choosing (i) inter-frame spacing (the amount of time that a station waits before transmitting) or (ii) the back-off times after a collision has occurred. These techniques are useful to meet deadlines because they can implement algorithms such as deadline monotonic [9]. But they 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" [13, 15, 14]. They do not only change some parameters in the IEEE 802.11 but they also require other signals to be transmitted. 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 finished transmitting its jamming signal, the node listens to find out whether other nodes transmit a jamming signal. If so, the node did not have the highest priority so it waits until the channel is idle again. The protocols based on "black-burst" were originally used to ensure that all real-time traffic is given a higher priority than non real-time traffic and dynamically change priorities of real-time traffic to achieve round-robin scheduling [15, 14]. These schemes treat all real-time messages in the same way and hence they are inappropriate for our purpose. The black-burst scheme in [13] implements static-priority scheduling though and is more interesting. However, all these black-burst schemes [13, 15, 14] have the drawback that (i) collisions can occur if the channel is idle and two nodes request to transmit simultaneously and (ii) the maximum length of the black-burst is proportionate to the number of priority levels, so only a small number of priority levels can be supported. Another technique [19], not based on IEEE 802.11, is to implement prioritiza-

tion 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 (for listening to the narrow band signals), 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 approach [2, 3, 7] of choosing back-off times as a function of priorities was adopted, and the polling scheme in IEEE 802.11 was refined with traffic classes.

MAC protocols have also been proposed from the real-time 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 [16] or by a leader [10]. It is also conceivable to use a TDMA template designed before run-time [8] 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 [6], 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. If the message selected by the EDF scheduling algorithm is in the node's queue of outgoing messages, then the node transmits this message otherwise it does not transmit. 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. MAC protocols based on token bus can be used in wireless channels and some of their analyses can be extended to sporadic messages [18]. Unfortunately, they only prioritize messages on a node; global prioritization is not achieved. As a result, deadlines can be missed although the utilization of the messages is low, and there exists a schedule of the messages that meets deadlines.

The dominance protocol [12] (used in for example the CAN bus [5]) uses global priorities and it can schedule sporadic message streams. Unfortunately, it requires that a node has the ability to receive an incoming bit from the channel while transmitting to the channel. Such a behavior is impossible on a wireless channel due to the large difference in transmitted energy and the received energy. 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.

3 System model

Consider n message streams $\tau_1, \tau_2, \tau_3, \ldots, \tau_n$ and m computer nodes $N_1, \ldots, N_m$. A message stream is assigned to exactly one node.

Workload. Message stream τ_i makes a sequence of requests to transmit a message. 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 T_i. Every message from τ_i requires to transmit for C_i contiguous time units. The maximum time from a request of a message from τ_i to the completion of the transmission of that message is called the *response time* of τ_i, and it is denoted R_i.

Success and failure. If there exists an overlap between a pair of transmission of data bits then both transmissions have failed. If a message finishes transmission later than D_i time units after it requested to be transmitted then the transmission has failed as well. The goal of our protocol is to schedule all messages in all message streams to finish their transmission before their deadlines. Then we say that the protocol has succeeded and we will (in Section 5) derive equations to compute whether a set of message streams succeeds using our protocol.

Priorities. Message streams are assigned unique priorities; these priorities are non-negative integers. Messages with low numbers have high priority. As a result, we will say that if a bit is "0" then it is dominant and if a bit is "1" then it is recessive. Let *npriobits* denote the number of bits required to represent the priorities. We use lower order first; that is, bit "0" is considered to be the most-significant bit in an integer. We do not assume any particular priority-assignment scheme.

Propagation. The time-of-flight between two arbitrary nodes i and j is unknown but it is non-negative and there is an upper bound α on the time-of-flights. We assume that when a node transmits and there is no collision, then all nodes receive exactly one copy of the message; that is, there is no noise, no hidden terminals and the transmitted signal takes only one path to the receiving node(s).

Nodes. We assume that nodes are equipped with real-time clocks. They are not synchronized; that is, their values may be different. For every unit of real-time, the clock increases by an amount. This amount is unknown but it is in the range $[1\text{-}\epsilon, 1+\epsilon]$, $0<\epsilon<1$. We let *CLK* denote the granularity of the clock. We assume that the clock does never "wrap-around".

A message may have one intended node as a receiver (unicast) or all nodes (broadcast); our protocol can deal with both types of traffic. We assume, however, that when a node receives a message it does not send an acknowledgement. A node can sense other transmissions only if the node does not transmit. We do not assume any particular modulation technique or coding scheme for the data bits but we assume that when data bits are transmitted, there is no interval of continuous idle time that exceeds F time units. (F is a design parameter that will be discussed later). We assume that nodes can transmit a carrier wave and all nodes are able to detect that carrier wave if they do not transmit themselves. A node needs *TFCS* time units to detect that a carrier wave was transmitted.

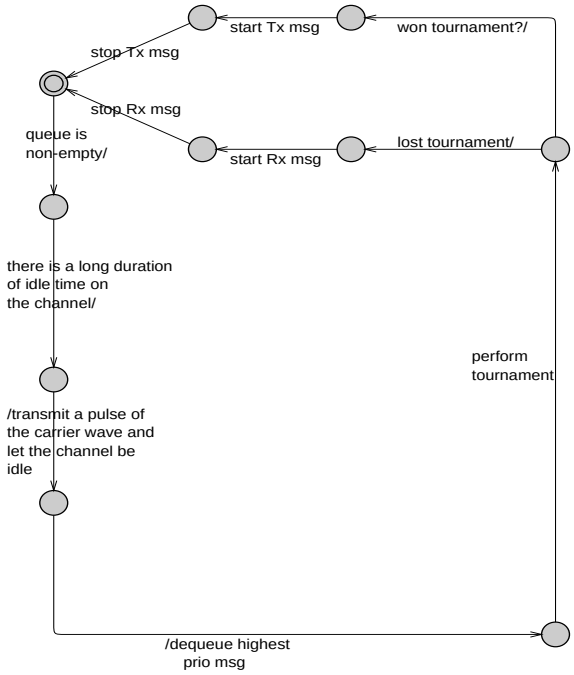

Fig. 1. Overview of the protocol.

The transceiver of a node needs at most $turnaround_{RxTx}$ time units to switch from reception to transmission or vice-versa.

We will describe the protocol using a 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, we let "/" separate the guards and the updates; the guards are before "/" and the update is after. We let "=" denote test for equality and let ":=" denote assignment to a variable.
We assume that when a time-out 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.

4 The protocol

Figure 1 gives an informal overview of the protocol. Between the protocol and applications on the node, there is a queue storing messages that requested to be transmitted. In the starting state (marked as a circle with a circle inside),

the protocol waits until the queue is non-empty. Then the protocol waits for a long idle time and then it transmits a pulse of the carrier wave. The beginning of the pulse represents a common reference point in time for all nodes. A node dequeues the highest priority message and then the nodes perform a tournament. If a node wins the tournament then it transmits the message. If a node loses the tournament then it continues to listen on the channel to figure out which priority was the winner and then it receives the message. If the node which lost had an application that requested this message then it is delivered to that application, if not then the message is discarded.

The remainder of this section is structured as follows. Section 4.1 gives a detailed view of the protocol. Section 4.2 explains the rationale for the design of the protocol and why it is robust against imperfections in clocks.

4.1 Details of the protocol

Figure 2 shows details of the protocol. The figure illustrates how the protocol is designed; the actual behavior 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 18. State 0 is the initial state. Each node has the following variables: 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. We let winner_prio[i] denote the bit i in the variable winner_prio. Analogously for prio[i]. We assume that when the protocol dequeues the highest-priority message then the variable prio is assigned the priority of that message. There are two functions carrierOn and carrierOff that can be called by a node. The function carrierOn requests to start the transmission of a carrier wave. It may take up to $turnaround_{RxTx}$ until the carrier actually starts to be transmitted but then it continues doing so. If carrierOff is called then it is requested that the carrier stops being transmitted but it may take up to $turnaround_{RxTx}$ until it stops. The symbol "carrier?" means: sense for a carrier and if there is a carrier then "carrier?" is true. E,F,G,H and SWX are constants used for time-outs, whose value we will choose later.

The states 1-5 in Figure 2 establish a common reference point in time between all nodes that requests to transmit. The transition 3→4 is designed to make the protocol robust to clock inaccuracies. The states 6-11 perform the tournament. During the tournament, nodes contend bit-by-bit, starting with the most significant bit.

If a node loses the contention of a bit then it loses the entire tournament but it continues to listen to find out which priority wins the tournament. If a node does not lose the contention during this bit, it continues with the contention for the next bit. Finally there is only one winner of the tournament because priorities are unique. This winning node makes the transition to state 14 and then transmits the message and then makes the transition to the initial state 0.

If the protocol contends with a dominant bit ("0") then it transmits a pulse of the carrier wave by taking the path $7 \rightarrow 8 \rightarrow 9 \rightarrow 11$. If the protocol contends with a recessive bit ("1") then it may take either the path $7 \rightarrow 10 \rightarrow 11$ or

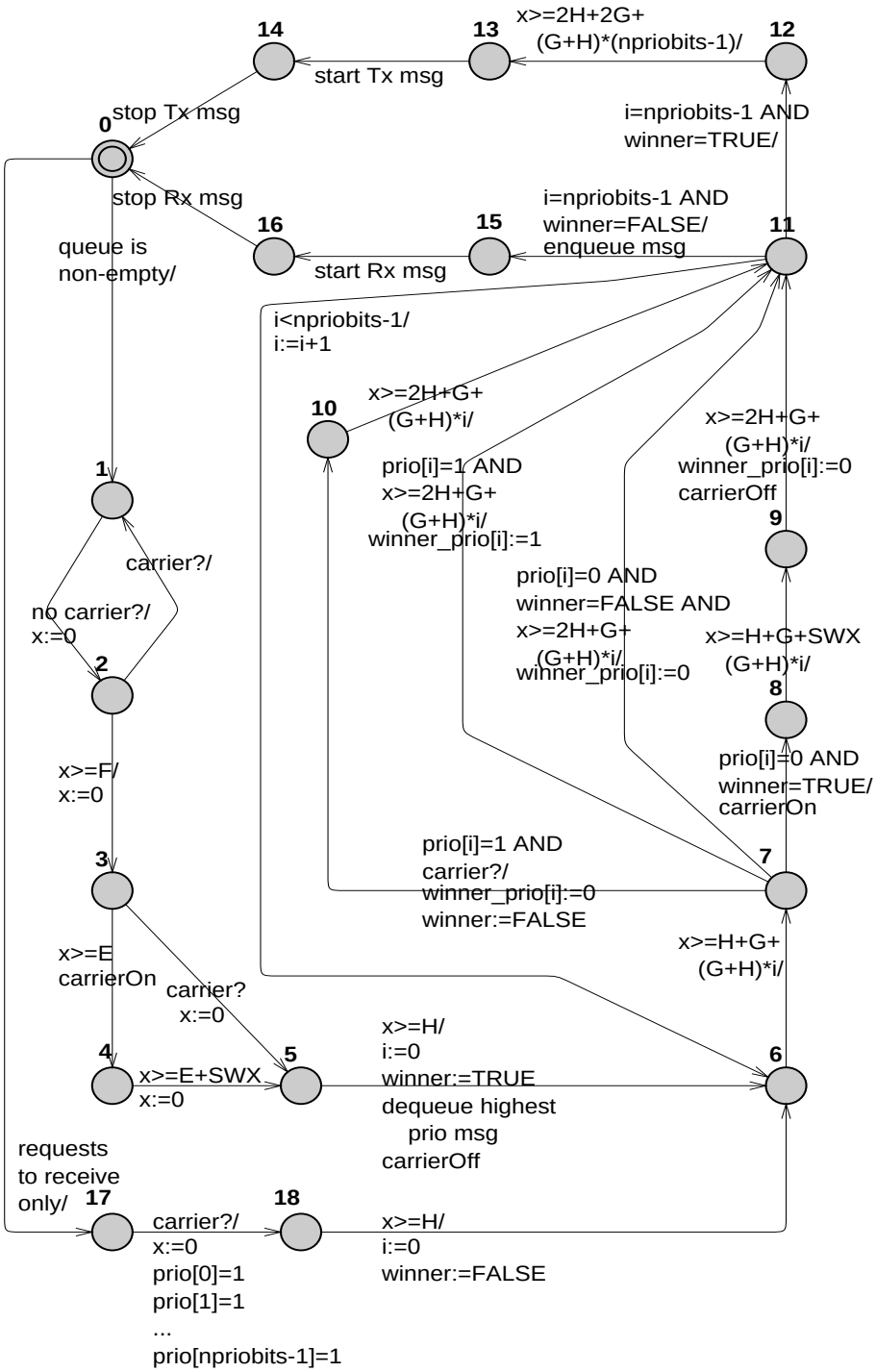

Fig. 2. Details about the protocol. This figure illustrates the design; the behavior is slightly different due to clock imperfection, time-of-flight of the carrier-signal and delays in the transitions.

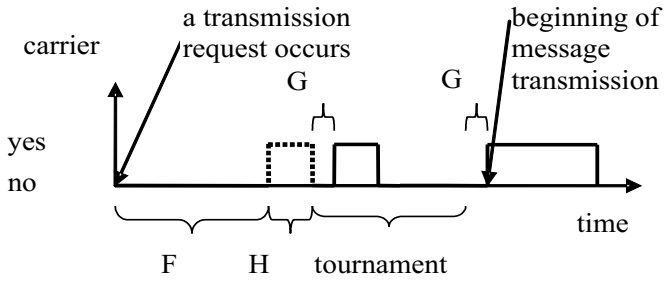

Fig. 3. An example of the carrier wave transmitted assuming npriobits=2 and the priority of the requested message is *1*. (The priority is encoded in a binary way as 01 which is signaled as: first a dominant bit and then a recessive bit). A solid box indicates that the node transmitted a carrier wave whereas a dotted box indicate that the node heard a carrier wave. The node that requested to transmit this message heard that another node transmitted a carrier wave so it did not need to transmit the carrier wave.

the path $7 \rightarrow 11$; it depends on whether the node heard a carrier wave (which signals that another node transmits a dominant bit). If a node contended with a recessive bit ("1") but heard a carrier wave then this node has lost.
Consider a node which has lost the tournament. It continues in the tournament and if such a node has a recessive bit then it acts in the same way as if it had not lost. The reason for this is that a recessive bit just listens; it does not transmit a carrier wave. However, if a node has a dominant bit and it has lost, then the protocol acts differently from the case when it had won; no carrier wave is transmitted. A node which only requests to receive, acts like a node losing the tournament from the start (see $0 \rightarrow 17 \rightarrow 18 \rightarrow 6$).

In order to understand the time-out parameters E,F,G,H and SWX, consider Figure 3. F denotes the initial idle time period. H represents the duration of a pulse of the carrier wave. G denotes a "guarding" time interval to separate pulses of carrier waves. 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. E makes the protocol robust to inaccuracies of when the nodes measure F. SWX is used for the protocol to wait to be sure that a request to transmit a carrier really has resulted in a carrier being transmitted.

Consider the automata in Figure 2 again. Traverse the path of the transitions of the winning node and observe the last time-out (the transition $12 \rightarrow 13$). Based on this, we can compute the transmission time of a message taking the overhead of the protocol into account as:

$$C_i' = C_i + 2H + 2G + (G + H) \times (npriobits - 1) + 2L \qquad (1)$$

By taking into account also the initial idle time (in state 2) we obtain:

$$C_i'' = F + E + SWX + C_i' \tag{2}$$

4.2 Rationale of the design and correctness

We will now discuss the correctness of the protocol and discuss how assigning values to the constants E,F,G,H and SWX affects the correctness. The protocol must satisfy:

- **Mutual Exclusion.** At most one node is in state 14.
- **Progress.** There are two types of progress (i) 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 a message of the backlogged nodes should be transmitted after the finished transmission.
- **Prioritization.** Of all nodes which were backlogged, the one that will transmit a message is the one that dequeues (at the transition 5→6) the message with the highest priority.

In order to assure that these properties hold we need to assure that certain events do not occur at the wrong time. We need to assure that:

- **When a node transmits a dominant bit, it is received by all other nodes.**
  Consider an iteration of the tournament. It must have been sufficient overlap between the time interval 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, this overlap becomes smaller and smaller for each iteration of the tournament. Hence we consider the last iteration of the tournament. We require:

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

  The motivation of Equation 3 is that the inaccuracy of the synchronization after the initial period of silence is $(E + SWX)$. Consequently, two different nodes can have different opinion on when a bit should be transmitted. If the time windows of the two nodes overlap by $TFCS + 2SWX$ then we can be sure that the node that attempts to detect the dominant bit will hear at least TFCS time units of the carrier. The reason for requiring $2SWX$ extra time units is that it may take SWX for the sender to enter Tx mode and when it has transmitted the carrier for TFCS time units, when it switches the carrier off, this may take effect immediately.

– **If one node i has perceived a time of silence long enough (F time units) to make the transition from state 2 to state 3 but other nodes perceive that the duration of silence to be less than F time units so far due to different time-of-flight and clock-imperfections, then node i needs to wait until all nodes have perceived this long time of silence.** The protocol should stay in state 3 for E time units to ensure this. We require:

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

– **A node which has lost the tournament must be in receiving mode in state 15 before it receives the data bits from the transmission of the winning node (which is in state 13).**
This is taken care of by the delay between state 12 to state 13. We know that the node which lost was in receiving mode because in the last bit in the tournament, it was in receiving mode (if the losing node would have transmitted in the last bit in the tournament then it must have lost in the last bit and transmitted a dominant bit, something which is impossible). For this reason, it is only required that the delay between state 12 to state 13 is large enough to ensure that the losing node reaches state 15 before the winning node reaches state 13. To do so we require that the following inequality is satisfied:

$$\begin{aligned}[2H + 2G + (H + G) \times (npriobits - 1)] \times [1 - \epsilon] - \\ [2H + G + (H + G) \times (npriobits - 1)] \times [1 + \epsilon] - \\ -(E + SWX) > 0\end{aligned} \tag{5}$$

– **During the tournament, the maximum time interval of idle time should be less than F, the initial idle period.**
This assures that if one node makes the transition from state 2 to state 3 (the initial idle time period) then all nodes will do it at most $E+SWX$ time units later. We require:

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

– **The time interval between two successive dominant bits must be long enough to assure that no node interprets the first dominant bits to be transmitted in the time interval for the second dominant bit.** The worst case occurs when these two bits are the last ones in the tournament. We require:

$$[2H + 2G + (H + G) \times (npriobits - 2)] \times [1 - \epsilon] -$$

$$[2H + G + (H + G) \times (npriobits - 2)] \times [1 + \epsilon] \\ -2CLK - L - 2\alpha - (E + SWX) > 0 \quad (7)$$

– **The time to wait from when a carrier is requested to be transmitted until it is known that a carrier is transmitted must be greater than the time required by the hardware.** Naturally, this requires:

$$SWX > turnaround_{RxTx} \quad (8)$$

The values of E,F,G,H and SWX must be selected to satisfy the inequalities 3-8. In order to get an idea of the magnitude of these values, we will work out an example.

Consider a typical distributed real-time system (a car, a factory or a ship) with a diameter of at most *300m*. This gives:α=*1*μs. Typical computers have CLK=*1*μs and ϵ=10^{-5} (assuming a low resolution timer and a poor quality crystal). We assume that the protocol is implemented on dedicated hardware and use L=*2*μs. We choose $TFCS$=*5*μs because busy tone detection of narrow-band signals have been estimated to need this time [19] and our application of carrier sensing is similar to busy tone detection. We choose $turnaround_{RxTx}$=*19*μs based on the requirement in IEEE 802.11 standard (see page 180 in [1]). We choose *npriobits=20*. One choice that satisfies the constraints for this example is: E=*8*μs, F=*2349*μs, G =*35*μs, H=*79*μs and SWX=*20*μs. Hence the overhead per message (calculated based on Equation 2) is 4775μs.

5 Schedulability analysis

Consider the mutual exclusion, the two progress properties and the prioritization property in Section 4.2. By combining them and a previous result on schedulability analysis of the CAN bus [17] we obtain that the response time can be calculated as a sum of the waiting time w_i and C_i''.

$$R_i = w_i + C_i'' \quad (9)$$

where C_i'' is defined as in Equation 2. The waiting time is obtained as:

$$w_i = B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i}{T_j} \right\rceil \times C_j'' \quad (10)$$

where *hp(i)* is the set of all message streams with a higher priority than τ_i. B_i can be computed as follows:

$$B_i = \max \left\{ C_j' : j \in lp(i) \right\} \quad (11)$$

where *lp(i)* is the set of all message streams with a lower priority than τ_i. Note that the schedulability analysis considers the initial idle time between states 1-4 to be a part of the "message" when we compute interference. This initial idle period should not be included when computing the blocking in Equation 11.

6 Conclusions and Future work

We have presented a MAC protocol for sporadic messages. The protocol is collision-free, does not require synchronized clocks and supports a large number of priority levels. We consider for future work (i) implementation of the protocol in Berkeley motes, (ii) automated formal verification of mutual exclusion, progress and prioritization, (iii) extending the protocol to deal with hidden nodes, (iv) analyzing the resilience of the protocol to noise in the carrier sensing and (v) techniques for achieving a low overhead and a large number of priorities on computer platforms with a large turnaround time from transmission to reception.

Acknowledgements

We are grateful to the reviewers for suggested improvements of the paper. This work was partially funded by Fundação para Ciência e Tecnologia (FCT).

References

1. IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
2. Aad, I.,Castelluccia, C.: Differentiation mechanisms for IEEE 802.11. In *Infocom*, pages 209–218, 2001.
3. Barry, M., Campbell, A.T., Andras, V.: Distributed control algorithms for service differentiation in wireless packet networks. In *Infocom*, 2001.
4. Baruah, S. K. , Mok, A. K., Rosier, A. K.: Preemptively scheduling hard-real-time sporadic tasks on one processor. In *IEEE Real-Time Systems Symposium*, pages 182–190, 1990.
5. Bosch. CAN specification, ver. 2.0, Robert Bosch GmbH, Stuttgart. Technical report, 1991.
6. Caccamo, M.,Zhang, L. Y.: An implicit prioritized access protocol for wireless sensor networks. In *23rd IEEE Real-Time Systems Symposium (RTSS'02)*, pages 39–48, Austin, Texas, 2002.
7. Deng, D.-J.,Ruay-Shiung, C.: A priority scheme for IEEE 802.11 DCF access method. *IEICE Transactions on Communication*, E82-B:96–102, 1999.
8. Kopetz, H., Grunsteidl, G.: TTP - a protocol for fault-tolerant real-time systems. *IEEE Computer*, 27(1):14–24, 1994.
9. Leung, J.,Whitehead, J.: On the complexity of fixed-priority scheduling of periodic real-time tasks. *Performance Evaluation, Elsevier Science*, 22(4):237–250, 1982.
10. Li, H., Shenoy, P., Ramamrithan, K.: Scheduling communication in real-time sensor applications. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, Toronto, Canada, 2004.
11. Mok, A.: *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment.* PhD thesis, Massachusetts Institute of Technology, 1983.

12. Mok, A.K., and Ward, S.: Distributed broadcast channel access. *Computer Networks*, 3:327–335, 1979.
13. Sheu, J.-P., Liu, C.-H., Wu, S.-L., Tseng, Y.-C.: A priority MAC protocol to support real-time traffic in ad hoc networks. *Wireless networks*, 10(1):61–69, 2004.
14. Sobrinho, J. L., Krishnakumar, A. : Quality-of-service in ad hoc carrier sense multiple access networks. *IEEE J. Selec. Areas Commun.*, 17(8):1353–1368, 1999.
15. Sobrinho, J. L., Krishnakumar, A. S: Real-time traffic over the IEEE 802.11 medium access control layer. *Bell Labs Technical Journal*, 1(2):172–187, 1996.
16. Thomas, W. C., Moussa, A. B., Rajeev, B., David, B.S. Contention-free periodic message scheduler medium access control in wireless sensor / actuator networks. In *IEEE Real-Time Systems Symposium*, pages 298–307, Cancun, Mexico, 2003.
17. Tindell, K., Hansson, H., Wellings, A.: Analysing real-time communications: controller area network (CAN). In *15th Real-Time Systems Symposium (RTSS'94)*, pages 259–263, 1994.
18. Tovar, E., Vasques, V.: Non pre-emptive scheduling of messages on SMTV token-passing ne tworks. In *12th Euromicro Conference on Real Time Systems (ECRTS00)*, pages 209–218, 2000.
19. Yang, X., Vaidya, N.: Priority scheduling in wireless ad hoc networks. *Wireless networks*.