Technical Document

For Review

Title

Nuno, PEREIRA, Björn ANDERSSON, Eduardo TOVAR

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: {npereira, bandersson, emt}@dei.isep.ipp.pt

http://www.hurray.isep.ipp.pt

Abstract

We propose a collision-free wireless medium access control (MAC) protocol that implements static-priority scheduling. This MAC protocol allows multiple masters and is fully distributed. It neither relies on synchronized clocks nor on out-of-band signaling nor on the use of base stations; 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. Besides being prioritized and collision-free, our protocol has the ability of supporting multiple broadcast domains and allowing parallel transmissions. We claim that this is a relevant result because a MAC protocol with these features enables schedulability analysis of sporadic message streams in wireless multihop networks.

Static-Priority Scheduling over Wireless Networks with Multiple Broadcast Domains

Nuno Pereira, Björn Andersson, Eduardo Tovar
IPP Hurray Research Group
Polytechnic Institute of Porto, Portugal
{npereira, bandersson, emt}@dei.isep.ipp.pt

Abstract

We propose a collision-free wireless medium access control (MAC) protocol that implements static-priority scheduling. This MAC protocol allows multiple masters and is fully distributed. It neither relies on synchronized clocks nor on out-of-band signaling nor on the use of base stations; 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. Besides being prioritized and collision-free, our protocol has the ability of supporting multiple broadcast domains and allowing parallel transmissions. We claim that this is a relevant result because a MAC protocol with these features enables schedulability analysis of sporadic message streams in wireless multihop networks.

1. Introduction

A fundamental problem in the design of distributed real-time systems is the sharing of a wireless communication channel such that messages' timing requirements are satisfied. Periodic message transmission requests can be scheduled using static table-driven scheduling. Sporadic [1] message requests can be scheduled using polling, but unfortunately such an approach is inefficient when the relative deadline is short as compared to the minimum inter-arrival time between two consecutive requests belonging to the same message stream.

An appealing solution is to assign a static priority to a message and then use a medium access control (MAC) protocol that selects for transmission the message with the highest priority [2]. This approach was originally used in wired networks (the CAN bus) [3] and it has recently been adapted to wireless networks [4, 5]. Experiments showed it to be surprisingly reliable for short-range communications; in particular, the message response-time formulations for the CAN bus could be migrated to the wireless domain and the calculated response times were validated by experiments of an implementation of the protocol using a low-power transceiver [5, 6]. Unfortunately, this wireless version of the dominance MAC protocol was designed for a single wireless broadcast domain (SBD); that is, assuming that every node receives every transmission. Therefore, it did not deal with a well-known phenomenon in wireless networks called *hidden nodes*. Previous work within the wireless networking community offered MAC solutions to the hidden node problem, but they were either not prioritized or they depended on out-of-band signaling. The absence of prioritization inhibits efficient schedulability of sporadic messages. The reliance on out-of-band signaling is a severe technological restriction since most of today's wireless transceivers do not have this capability.

In this paper we therefore propose a MAC protocol for wireless networks where a broadcast from a node does not necessarily reach all nodes in the network. Consequently, the hidden node problem must be dealt with. Our MAC protocol is the first prioritized and collision-free MAC protocol designed to successfully deal with hidden nodes without relying on out-of-band signaling. We consider this research to be significant because it proposes an enabling technology allowing schedulability analysis in wireless multihop networks with multiple broadcast domains; for example to exercise in practice the analysis proposed in [7].

The remainder of this paper is structured as follows. Section 2 gives background on prioritized MAC protocols and outlines the system model used throughout the rest of the paper. Section 3 presents the main idea of the design of our new protocol, whereas in Section 4 formal description of the proposed MAC protocol is provided. Section 5 validates the protocol experimentally and Section 6 discusses related work. Finally, in Section 7 conclusions are drawn.

2. Background and Assumptions

2.1. Dominance Protocols

Dominance/binary countdown protocols [1] are the main inspiration for the MAC protocol proposed throughout this paper. In such protocols, messages are

assigned unique priorities and before nodes transmit they perform a collision resolution phase such that the node trying to transmit the highest priority message succeeds.

During the collision resolution phase, each node sends the message priority bit-by-bit, starting with the most significant one, while simultaneously monitoring the medium. The medium must be devised in such a way that nodes will only detect a "1" value if no other node is transmitting a "0". If any node is transmitting a "0", then every node will detect a "0" bit regardless of what the node itself is sending. For this reason, a "0" is said to be a dominant bit and a "1" is a recessive bit. Therefore, low priority numbers represent high priorities. 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 the collision resolution phase, and this node (the winning node) proceeds with transmitting the data bits.

An adaptation of this approach to a single wireless broadcast domain has been previously proposed [4, 5]. In those works, the MAC protocol evolves trough three main phases: *synchronization*, *tournament* and *receive/transmit*.

Nodes have to recurrently agree on a common reference point in time. This phase is called *synchronization* and happens before every collision resolution phase (named *tournament*). This common reference point in time is achieved with a bounded error, and this error impacts the protocol design by imposing the duration of each priority bit and the silence intervals between them. This is required to guarantee that all nodes can perceive priorities correctly.

The *tournament* phase is similar to the collision resolution phase in dominance/binary countdown protocols; nodes also transmit priorities bit-by-bit and the highest-priority message is granted transmission. There is one important difference though; a node contending with a dominant bit transmits a carrier wave, while a node with a recessive bit transmits nothing, but listens. In this way, a node with a recessive bit can detect whether another node is dominant on that bit, thus emulating a wired-AND behavior.

Figure 1 exemplifies the tournament phase where three nodes (N_1, N_2 and N_3) contend for channel access with 6 priority bits. In the example, N_2 is recessive in bit 3, but hears a dominant bit, and hence it stops transmitting priority bits and proceeds only monitoring the medium. Observe that while N_2 has a dominant bit 4, it has previously lost the arbitration (in bit 3) and thus N_2 does not send its dominant bit 4 or any other subsequent bits.

After the tournament, nodes enter into the *receive/transmit* phase. Here, nodes that have lost the tournament start monitoring the medium to receive data. The node with the winning priority (if priorities are unique, there will be only one winning node) goes on transmitting the data part of the message. Note that in [4, 5] a bit of the

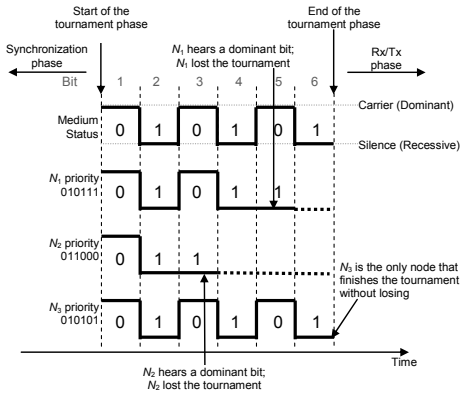


Figure 1. The tournament.

tournament is different from a data bit. During the tournament, each priority bit has a length large enough to encompass the time needed to switch between reception/transmission modes and to detect a carrier. But a node that wins the tournament may transmit the data bits at the full data rate permitted by the radio transceiver.

2.2. The New Challenge

It turns out that the extension of [4, 5] to deal with multiple broadcast domains without relying on synchronized clocks or out-of-band signaling is not trivial. One of the most relevant problems that such extension must deal with is the well known *hidden node* problem [8]. A node is said to be hidden from another node if they are out of each others' range, while both are within range of a third node. Because the two nodes (N_1 and N_3 in Figure 2) cannot detect when the other is transmitting, they may cause undue reception collisions at the third node (N_2 in Figure 2).

The hidden node problem received serious attention in the wireless community because they can cause collisions which lowers system throughput. For real-time traffic, dealing with hidden nodes is crucial, since a collision may cause a deadline miss.

Another problem in networks with multiple broadcast domains (MBD) is the *exposed node* problem. Exposed nodes occur when a node refrains from transmitting because another neighbor node transmits. But the receiving nodes are far apart and do not experience a collision. The existence of exposed nodes may reduce the number of parallel transmissions but it does not violate the correctness of reception. Therefore, the exposed node problem is considered outside of the scope of this paper, albeit being considered as future work.

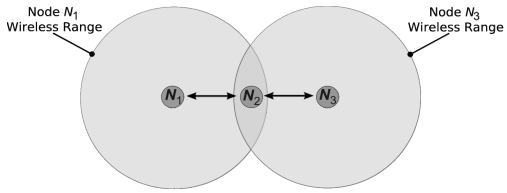


Figure 2. Example illustrating hidden nodes.

2.3. System Model and Assumptions

The computer nodes (or simply nodes) have only one transceiver and cannot send any out-of-band signals. They use the radio transceiver to transmit/receive data messages or pulses of a carrier wave during a certain interval of time. All nodes perform broadcasts; that is, every neighbor is an intended receiver of the transmissions (but a broadcast from a node does not necessarily reach all nodes). The radio transceivers are characterized by three relevant timing parameters: T_{RX}, T_{TX} and T_{CS}. The transceivers take T_{RX} time units to switch from idle mode to reception mode and T_{TX} to switch from idle mode to transmission mode. T_{CS} represents the time nodes take to detect a carrier wave when in receive mode.

Communication links are assumed to be bidirectional and topology does not change while nodes are trying to access the medium. We say that a data message transmission that overlaps at a receiver is a collision and causes that receiver to fail the reception of any ongoing data message transmission. When one or more nodes transmit a carrier pulse at the same time, it is possible for a listening node within range to detect the transmission of a carrier wave.

Nodes execute applications that make requests to transmit data messages. No assumption is made about the origin of messages; two different messages may belong to the same sporadic message stream or they may not. Each message has a unique priority in the range $0..2^{npriobits}-1$, where *npriobits* is the number of bits required to represent the priorities. This priority is denoted as an array of bits `prio`[1..*npriobits*], where the most significant bit is `prio`[1]. Each node has a real-time clock with a granularity denoted as *CLK* that, for every unit of real-time, increases by an amount in the range $[1-\varepsilon, 1+\varepsilon]$, $0<\varepsilon<1$.

We also assume that the propagation delay has an upper bound α and the delay due to executing on a finite-speed processor is represented by L.

In addition, the following definition is used.

Definition. 2-neighbor. We say that a node N_i is a 2-neighbor to node N_j if either (i) N_i is a within the radio range of N_j (N_i is said to be a neighbor of N_j) or (ii) there exists a node N_k such that N_i is a neighbor of N_k and N_k is a neighbor of N_j. As an example, in Figure 2, both nodes N_2 and N_3 are 2-neighbors of N_1.

3. Design

We will now discuss key aspects to be considered in the design of a correct dominance protocol for wireless networks with multiple broadcast domains.

3.1. Synchronization

Prior to the tournament, nodes need perform a synchronization phase where they agree on a common time reference. This synchronization is essential so that nodes exchange priority bits correctly and it must be achieved at least among 2-neighbors nodes.

Consider first the problem of achieving synchronization within a single broadcast domain. As described in [4, 5], this can be performed by letting a node wait for a long period of silence. Our protocol uses a similar approach. A node that requests to transmit monitors the medium to find a period of silence. After this, a node that wishes to transmit starts sending a carrier pulse. This carrier pulse signals that the node will start a tournament and establishes a time reference with other nodes listening, thus it is the *synchronization carrier pulse*.

To make such scheme work across two hops, the synchronization carrier must be retransmitted. That is, a node that detects a synchronization carrier being sent starts itself transmitting a synchronization carrier. A possible solution is to immediately retransmit any carrier wave heard while in the synchronization phase. That is, nodes start synchronization by sending a carrier pulse, and nodes that detect this pulse begin retransmitting the synchronization pulse straightaway.

This is illustrated in Figure 3a) for nodes N_1 and N_2. This solution causes the carrier pulse used to achieve synchronization to be propagated arbitrarily far way. In reality, we only need to achieve synchronization among 2-neighbors, and to accomplish this, the synchronization pulse only needs to be propagated two hops away. However, this demands that nodes are able to differentiate between carriers directly transmitted from a node within radio range or a retransmitted carrier. The only way to do so (without out-of-band signaling) is to either detect a different duration of the carrier used for synchronization, or different patterns (predefined combinations of pulses and silence). Figure 3b) depicts the former solution. In this case, nodes cannot start retransmitting the carrier immediately. They need to wait until they are able to detect that a carrier with the duration of a synchronization carrier is being transmitted and decide to retransmit the synchronization carrier. The latter solution is similar, but a node sends a "synchronization carrier" by sending a combination of carrier pulses and silence. A node monitoring the medium will only detect a "synchronization carrier" if it observes this pattern.

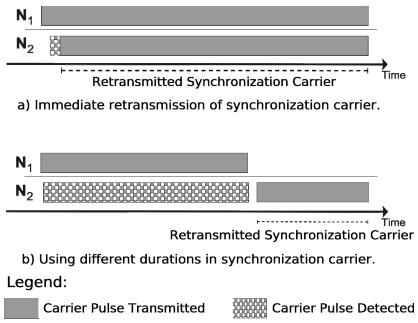


Figure 3. Alternatives do synchronization carrier retransmission.

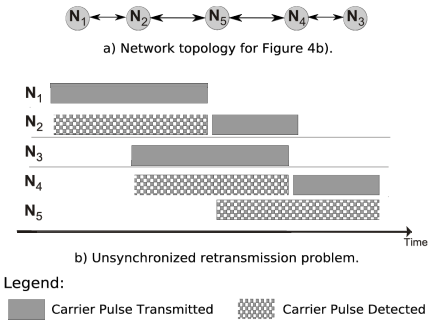


Figure 4. Synchronization carrier retransmission problem.

Unfortunately, this cannot be applied to bound the synchronization amongst 2-neighbours. This happens because two or more nodes that start synchronization at different points in time may hinder a third node from detecting either synchronization pulses. This scenario is depicted in Figure 4. Figure 4a) presents the topology for the illustration in Figure 4b). This scenario involves two nodes, N_1 and N_3 that are more than two hops from each other and start synchronization by sending a carrier pulse. These nodes start the synchronization at different points in time, causing the nodes one hop away (N_2 and N_4) to retransmit the synchronization carrier (notice that this carrier pulse retransmitted has a smaller duration). The problem arises when a node (N_5) two hops way from both N_1 and N_3 observes the retransmission of these two synchronization carrier pulses and thus may mistake them by a synchronization pulse sent from a node one hop away.

A similar scenario can show that using different patterns for the synchronization carrier results in a similar problem.

The solution of immediately retransmitting the synchronization carrier arbitrarily far away may appear to cause very poor performance. However (as we will see in Section 5), this impact is in reality very small. First, although synchronization pulses must be propagated throughout the entire network, it is still possible for many nodes to transmit data messages in parallel (as shown in Section 3.2). Second, the duration of a priority bit is affected by the synchronization error between 2-neighbors but it is independent of the synchronization error between any two nodes in the network more than two hops way from each other, and hence it is independent of the network diameter.

This scheme also guarantees progress as all nodes will either start a tournament themselves (thus sending a synchronization pulse) or detect and retransmit a synchronization pulse.

3.2. Tournament

During the tournament, priority bits are propagated two hops away. This is done by performing the transmission of each bit in two stages. In the first stage – *Transmission Stage*, each node transmits its own priority bit. In the second stage – *Retransmission Stage*, nodes retransmit the priority bit detected at the first stage. If a node transmitted or detected a dominant bit in one of the two priority bit transmission stages, then it knows that the current priority bit was dominant.

Algorithm 1 details this approach. Nodes execute this algorithm for each priority bit. Function `prioBitTxStage()` accepts the value of the priority bit to be transmitted and returns the priority bit value detected (a dominant bit detection when the node had a recessive bit) or transmitted (the node itself transmitted a dominant bit). This function only transmits a dominant bit if the variable `winner` is equal to `TRUE`, because nodes only send their priority bits while they are potential winners.

During the transmission stage (Algorithm 1, line 3), the

Algorithm 1 Bit transmission

```
Requires: Nodes have established a common time
          reference prior to the transmission of
          priority bits
Input
  prio   : array containing the priority bits
  i      : integer with the current priority bit index
  winner: a boolean initialized to TRUE at the beginning of
        the tournament in all nodes with pending messages; used
        also to indicate if nodes are active in the tournament
Variables
  prioBitTx : integer with priority bit in transmission step
  prioBitRTx: integer with priority bit in retransmission
        step
  winner_prio: array containing the priority bits of the
        winner
1:  begin
2:      { Priority bit transmission stage }
3:      prioBitTx ← call prioBitTxStage(prio[i])
4:      { Priority bit retransmission stage }
5:      prioBitRTx ← call prioBitTxStage(prioBitTx)
6:      { Decision }
7:      winner_prio[i]=prio[i];
8:      if prioBitTx = 1 OR prioBitRTx = 1 AND
9:        prio[i] = 1 then
10:         winner_prio[i] ← 1;
11:         winner ← FALSE
12:     endif
13: end
```

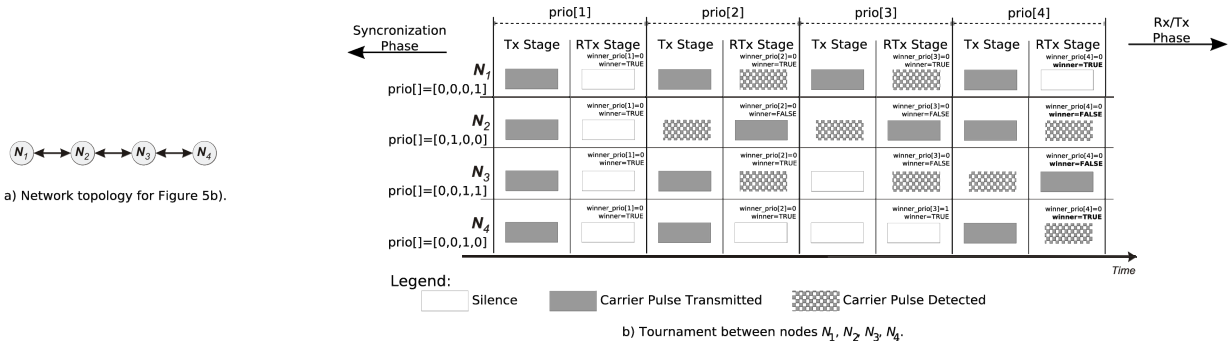


Figure 5. Tournament example.

value of the current priority bit is used. In the retransmission stage, it is uses the value returned by `prioBitTxStage()` in the previous call (Algorithm 1, line 5). At the end of the two stages, nodes decide what was the winning priority observed and if they lost the tournament in this priority bit (Algorithm 1, lines 7 to 12).

Figure 5 shows a tournament between four nodes with *npriobits* = 4. Nodes N_1, N_2, N_3 and N_4 are accessing the medium with priorities 1, 4, 3 and 2, respectively. Nodes are assumed to have achieved synchronization before starting the transmission of priority bits, and the synchronization error is ignored in this example. The topology for this scenario is represented in Figure 5a). Observe that in priority bit 2 (`prio[2]`), N_2 detects a dominant bit during the transmission stage which causes it to send a carrier pulse in the retransmission stage and to withdraw from the tournament (N_2 sets `winner=FALSE`). In bit 3 (`prio[3]`), N_2 detects again a dominant bit during the transmission stage. When N_2 performs the retransmission of this bit, N_3 will detect it and withdraw from the tournament.

Notice that, at the end of the tournament, two nodes N_1 and N_4 have `winner=TRUE` and thus both will transmit. Notice the topology graph in Figure 5a). N_1 and N_4 do not share any common receiver, thus their transmission will never collide. This illustrates an important characteristic of our protocol: it allows multiple winners, and thus parallel transmissions can occur in the system.

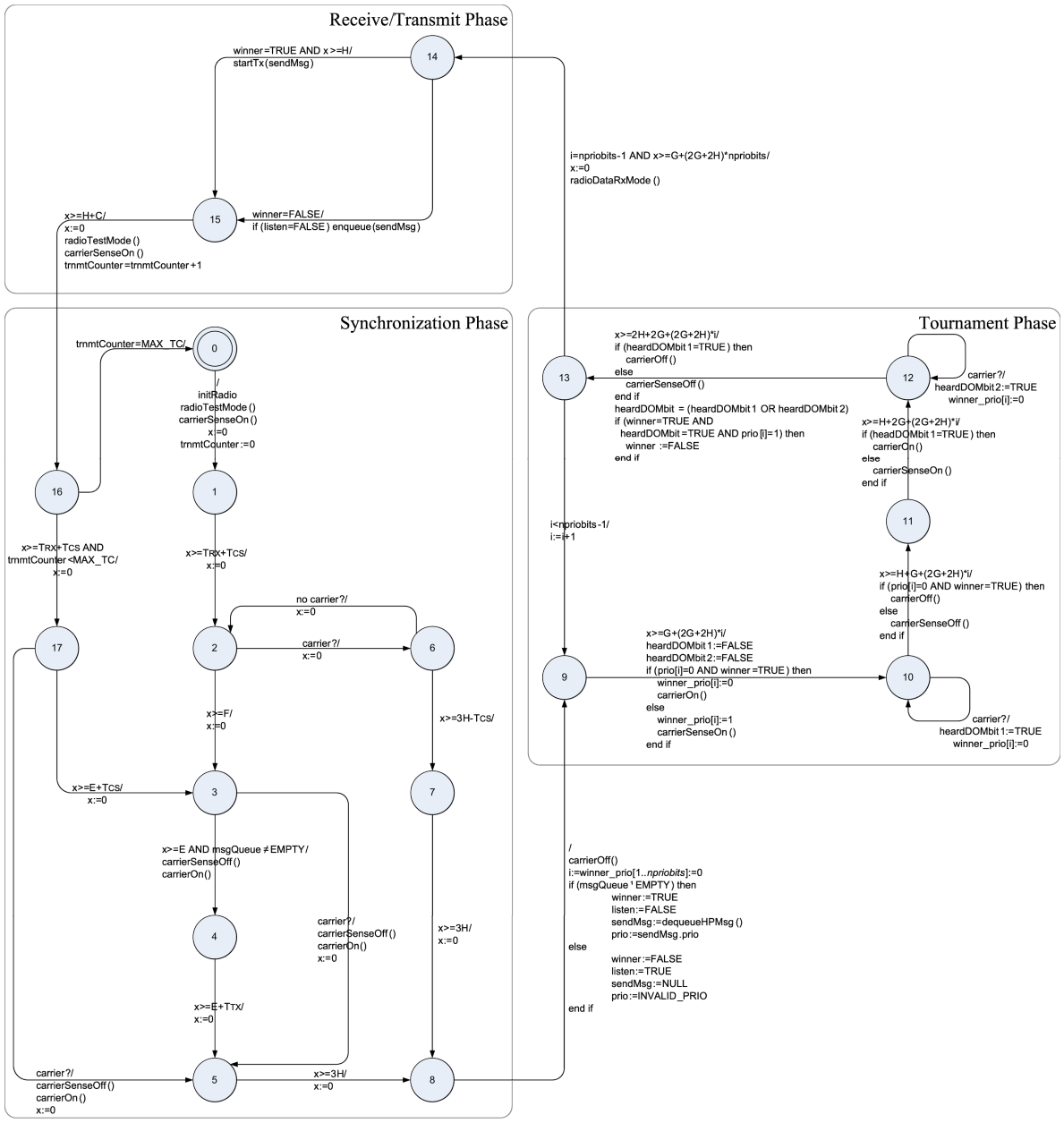
4. Full Protocol

The protocol is illustrated 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). The guards and the updates are separated with "/"; the guards are before "/" and the update is after. Let "=" denote test for equality and ":=" denote assignment to a variable. States are numbered from 0 to 17. State 0 is the initial state. Associated to each node the following variables are considered: a clock `x`; an integer `i` within the range 1..*npriobits*; a boolean variable `winner`, an integer `trnmtCounter` and two arrays of bits `prio` and `winner_prio`.

A node may use nine function calls. The function `initRadio()` is used to perform initialization of the radio chip. `radioTestMode()` sets the radio into a mode where it is able to transmit unmodulated carrier pulses. The function `radioDataRxMode()` prepares the radio to receive a data packet. `startTx()` instructs the radio to transmit the data message passed as argument. The function `carrierOn()` starts transmitting a carrier and continues doing so until function `carrierOff()` is called. Function `carrierSenseOn()` is used to set the radio into receive mode and starts detecting carrier pulses, while `carrierSenseOff()` is called to stop detecting carrier pulses. To get the highest priority message from the local queue of outgoing message requests, a node calls `dequeueHPMsg`(). The symbol "`carrier?`" is used in the timed-automata of Figure 6 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 (*C, E, F, G, H,* T_{CS}, T_{TX} and T_{RX}) are constants. The meaning of these timeouts is briefly given in Figure 6 and their values will be instantiated later in this paper.

To describe more precisely the main concept of the 2-neighbour synchronization solution, we will study the simple sequence of state transitions that nodes can take to synchronize after they boot. After initializing the radio, nodes evolve to State 1. Transition 1→2 ensures that the radio changes to receive mode and monitors the medium for time enough to detect if the medium is idle or not. In State 2, nodes wait for a long duration of silence (denoted by *F*), such that no node disrupts a tournament being performed by other nodes. Then, nodes with pending messages perform transition 3→4 after waiting for *E* time units, so that other nodes have time to reach State 3. Nodes that make the transition 3→4 start sending a carrier pulse

.



Timeouts:

- *C* - Timeout to wait for receive/transmit messages;
- *E* - Timeout to cope with synchronization imperfections (such as clock inaccuracies and transmit/receive switching times).
- *F* - Initial idle period of silence;
- *G* - Gap between the bits in the tournament;
- *H* - Duration of a bit in the tournament;
- T_{CS} - Time For Carrier Sensing
- T_{TX} - Time that the radio takes to switch between idle and transmit mode.
- T_{RX} - Time that the radio takes to switch between idle and receive mode.

Constants:

MAX_TC- Number of tournaments after which transition 16→0 is made;

Figure 6. Protocol state automaton.

in order to synchronize. Other nodes may take one of the two following sequence of state transitions: (i) a node is in State 3 and has pending messages and it does not hear a carrier for *E* time units so it makes the transition 3→4, or (ii) a node in state 3 (either because it is waiting to make transition 3→4, or does not have pending messages) can detect the carrier pulse being sent by other nodes and perform transition 3→5. Nodes making transition 3→5 start transmitting the synchronization carrier pulse and immediately reset their timers, but nodes making transition 3→4 wait for T_{TX} to reset their timers because only at that time the carrier pulse is actually being transmitted. Nodes

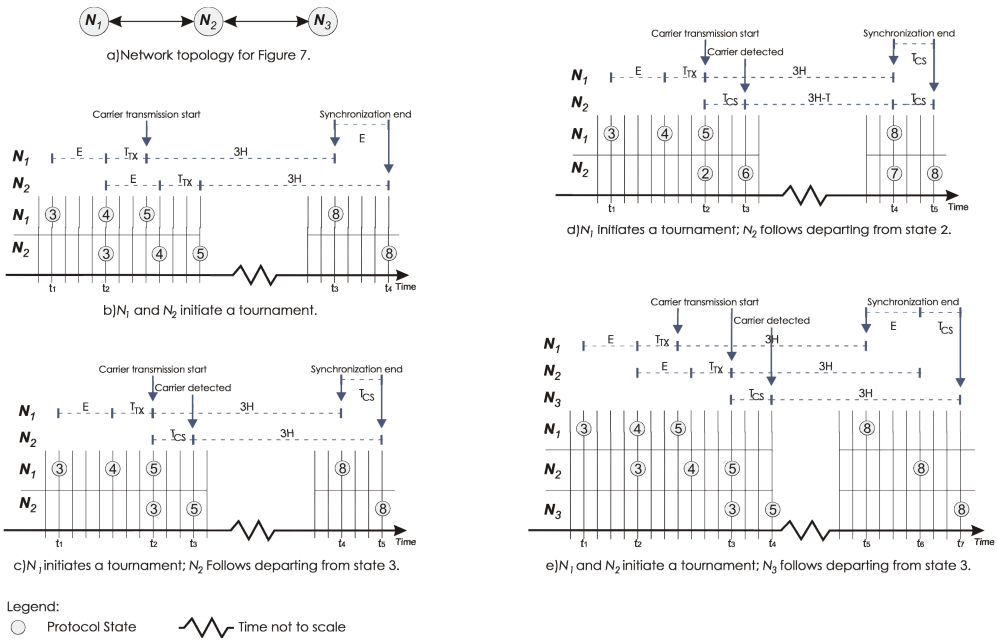


Figure 7. Synchronization scenarios.

then stay in State 5 sending the carrier pulse and make transition 5→8 after $3H$ time units. At this point nodes stop sending the carrier pulse and synchronization ends with nodes resetting their timers.

Nodes can actually take a number of different sequences of state transitions to synchronize. Section 4 discusses all possible sequences nodes can take to synchronize, along the resulting synchronization error.

4.1. Synchronization Error

As observed previously, the synchronization error influences the duration of the priority bits (H) in the tournament and the space between them (G). We will now look into the synchronization error by studying the possible scenarios for nodes to achieve synchronization. Figure 7a) illustrates the first scenario. Consider two neighbor nodes N_1 and N_2, both with pending messages. Node N_1 enters State 3 at time t_1, and stays here for E time units, to ensure that other nodes have time to reach State 3. In a worst case scenario, a node N_2 will enter State 3 exactly at the same time node N_1 leaves State 3, at time t_2. If we choose E such that $E \leq T_{TX} + T_{CS}$, then N_2 will never detect the carrier being sent by node N_1, and thus will move on to State 4 and, after this, nodes will do exactly the same transitions, but with E time units of difference between them. When nodes finally finish synchronization (nodes reach State 8) at times t_3 and t_4, with a difference of E time units. Thus, we say that the synchronization error in this scenario is E.

Figures 7b) and 7c) illustrate the state sequences two neighbor nodes N_1 and N_2 may take when only node N_1 has pending messages. Both figures show that nodes can enter State 8 with a maximum difference of T_{CS} time units.

The synchronization scenarios depict in Figures 7a), 7b) and 7c) analyze the synchronization between two 2-neighbor nodes. However, the synchronization error must be studied between 2-neighbors, and Figure 7d) does this. Consider three nodes N_1, N_2 and N_3. Nodes N_1 and N_2 perform the same sequence of state transitions as in Figure 7a), already described. Node N_3 detects the retransmission of the carrier pulse made by node N_2 at time t_4. Consequently, N_3 reaches State 8 T_{CS} time units after time t_6, when node N_2 reached State 8 and $E+T_{CS}$ after node N_1, that reached state 8 at time t_5. The sequence of state transitions made by node N_3 in this scenario is similar to the one made by N_2 in Figure 7b), and likewise node N_3 could be in State 17 $E+T_{CS}$ time units before time t_4, and take transition 17→5. Observe that N_3 can take a sequence of state transitions similar to node N_2 in Figure 7c), and thus reach state 8 T_{CS} time units after N_2 and $2\times T_{CS}$ after node N_1. Based on this, one can observe that the maximum synchronization error between 2-neighbors δ is:

$$\delta = \max\{ E+T_{CS}, 2\times T_{CS} \} \qquad \textbf{(1)}$$

It is necessary to select time-out parameters to ensure that synchronization before the tournament works and that the synchronous behavior in Figure 6 is achieved. See Appendix A for details on how to do this. This gives us the following properties:

1. Collision-free. There is no pair of nodes (N_1, N_2) such that (i) N_1 is a 2-neighbor of N_2 and (ii) N_1 and N_2 are

both in state 15 and (iii) the variable `winner` in N_1 and N_2 are `TRUE` simultaneously.

2. Progress. Consider a node N_1 that requests to transmit. If for every 2-neighbor node N_i of N_1 such that $prio_{N_i} < prio_{N_1}$, then it holds that at most Q_{HP} after the request to transmit, node N_1 is in State 15 and the variable `winner` is equal to `TRUE`. Q_{HP} is given by:

$$Q_{HP} = T_{TX}+T_{CS}+F+2\times(3H+(npriobits\text{-}1)\times(2G+2H)+G+H)+C+2\alpha+2L \quad \textbf{(2)}$$

3. Prioritization. If a node N_1 requests to transmit and node N_1 is in state 15 and its variable `winner` is equal to *FALSE* then there is a node N_2 such that (i) N_2 is a 2-neighbor of N_1 and (ii) N_2 requests to transmit and (iii) $prio_{N_2} < prio_{N_1}$.

4.2. Dealing With Errors

According to the automaton in Figure 6, the normal behavior of a node after sending/receiving a data message (in state 16) it to proceed to another synchronization without waiting to observe a long period of silence. If nodes only waited for a long period of silence when they boot up, then a single synchronization failure could compromise the network for an arbitrarily long period of time. To avoid this, nodes periodically do transition 16→0 and this will cause them to try observing a long period of silence. This transition is made every a node performs `MAX_TC` tournaments. The value `MAX_TC` can be adjusted to the quality of the radio transceivers in the nodes. In our experiments, we have found that `MAX_TC=100` is an acceptable value.

5. Experimental Evaluation and Discussion

The values chosen for the protocol timeouts are dependent on the implementation platform. However, in order to give an idea of the magnitude of the values for the timeouts C, E, F, G, H, T_{CS}, T_{TX} and T_{RX} in Figure 6, we will work out an example. Assuming a radio range with a maximum range of 30 m, we have $\alpha = 0.1\mu s$. Typical computers have $CLK = 1\mu s$ and $\varepsilon = 10^{-5}$. Assuming that the protocol is implemented on dedicated hardware, $L = 1\mu s$. We choose $T_{CS} = 5\mu s$ because busy tone detection of narrow-band signals have been estimated to need this time [9] and our application of carrier sensing is similar to busy tone detection. We assume $T_{TX}=T_{RX}=1\mu s$; such transceivers have been implemented [10]. Let $npriobits=5$. One choice that satisfies the constraints for this example is: E=10 μs, F=557 μs, G=21μs, H=30μs. Messages are assumed to have at most 54 bytes [10]; at a data rate of 36 Mb/s, C=12 μs. Thus, instantiating (2), we have:

$$Q_{HP} = 1+5+557+2\times(3\times30+(5\text{-}1)\times(2\times21+2\times30)+21+30)+12 = 1676\ \mu s$$

We have implemented the protocol in a discrete event simulator. The simulation implements the automaton in Figure 6. Using again *npriobits*=5, we tested the protocol by running 100 independent simulation runs for each scenario with different probabilities of missing the detection of a carrier pulse. Each node was setup with one message stream having a unique priority between [0,29] and an exponentially distributed inter-arrival time, with an expected value ranging between [0.01, 1] seconds.

For each simulation run, it was generated a random topology with 30 nodes, where each node has in average 3 neighbor nodes. An example of topology generated by the simulator is illustrated in Figure 8a). The numbers close to each node represent the priority given to the message stream of that node. The topology was constructed by randomly placing nodes within a bounded area and maintaining a minimum distance between nodes. Connectivity is then determined if the power level at the receiver is above a defined threshold. The power level at the receiver is calculated as a function of the distance between nodes, using a log-normal shadowing model [11] with parameters $P_t = 0$ dBm, $G_t = G_r = 1$ dBi, $d_0 = 1$ m, $\lambda = 0.125$ m (assuming a 2.4 Ghz operating frequency) , $n = 2.5$ and $\sigma = 5$.

In all simulation runs, nodes perform more than 50 000 tournaments. After each tournament, we detected whether the correctness properties collision-free, progress and prioritization were satisfied for all nodes in the network. Tournaments where any node in the network failed to satisfy one of the properties are named *erroneous tournaments*. The probabilities of observing an erroneous tournament are presented in Figure 9. The fact that no errors were found with a perfect detection of carriers presents evidence that the protocol correctness properties are satisfied. Observe also that the error rate is still under a low threshold when nodes fail to detect carriers.

Let us consider again Figure 8a). This figure also depicts the result of a tournament where all nodes requested to transmit. With this example we can clearly observe the 4 parallel transmissions (the nodes winning a tournament are marked with a solid black circle) allowed by the protocol. Figures 8b) to 8e) illustrate the execution of the tournament for the same scenario of Figure 7a). Figure 7b) shows that, at the beginning of the tournament, all nodes are potential winners, and Figures 8c) to 8e) illustrate the nodes that were potential winners at the end of transmission of bits 1 to 3 of the tournament.

However, Figure 8a) depicts a particularly interesting outcome. Node with priority 26 does not win, but still all its 2-neighbors are not winners either. This happens because node with priority 15 is a 2-neighbor to node 26 and it causes node 26 to lose. But later in the tournament, node 15 lost as well (observe sequence of Figure 8b) to 8d). This phenomenon is well-known [12] and was dubbed *multihop competing*. Figure 8f) presents another possible selection of nodes that increases the parallelism

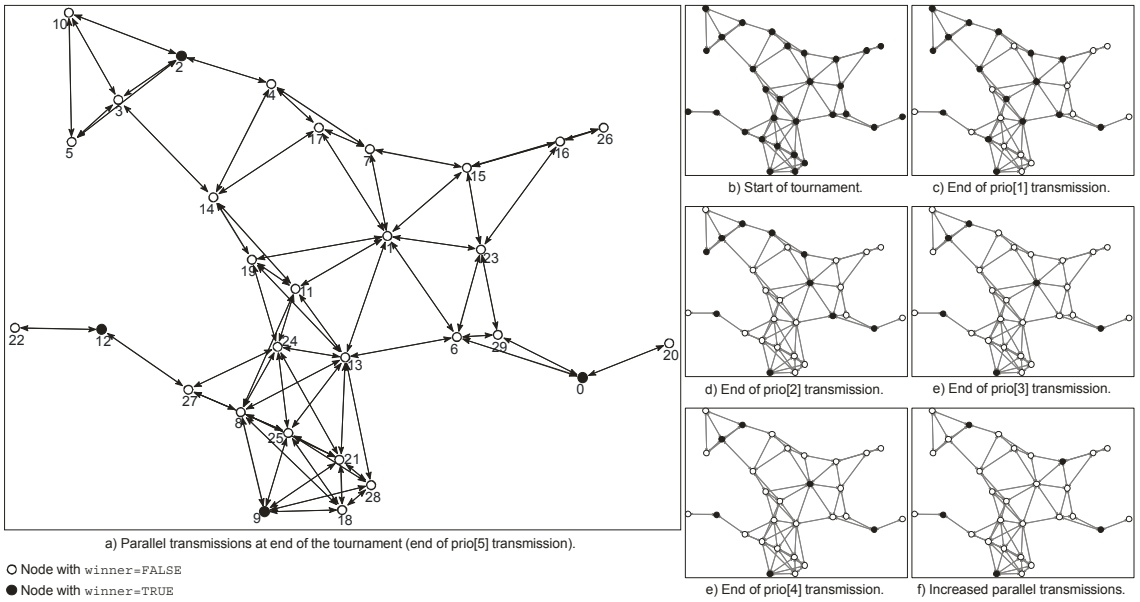


Figure 8. Topology graphs illustrating parallel transmissions and priority bit transmission.

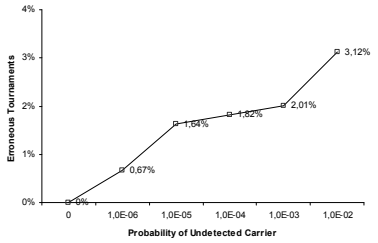


Figure 9. Probability of an erroneous tournament.

and still respects collision-free, progress and prioritization properties. This demonstrates that our protocol does not maximize the number of parallel transmissions but it does allow parallel transmissions. As argued previously in Section 3.1, the limitation in parallel transmissions is not because the synchronization carrier pulse is propagated network-wide; it is because of multihop competing and this is subject of ongoing research.

6. Related Work

The introduction of the wireless LAN standard IEEE 802.11 stimulated development of many [13-18] prioritized Carrier Sense Multiple Access (CSMA) MAC protocols and a few of them [13-15] were adopted for the real-time profile IEEE 802.11e. 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 (for listening to the narrow band signals), requires extra bandwidth (for the narrow band signals) and it supports only two priority levels. We believe that this out-of-band signaling solution [19] can be extended to k priority levels (although the authors do not mention it), but doing so would require $2k$ narrow band signals. Unfortunately, all [13-19] of these MAC protocols can suffer from collisions making it impossible to prove that timing requirements are satisfied.

MAC protocols have also been proposed from the real-time systems community with the goal of meeting deadlines. They are collision-free. Some protocols use tables (sometimes called *TDMA templates*) with explicit start times for message transmissions. Such tables are created at run-time (see [20] or [21]) or at design time [22]. However, all these time-table approaches have the drawback of requiring that sporadic message streams are dealt with using polling, which is inefficient. Another approach, Implicit-EDF [23], is based on the assumption that all nodes know the traffic on other nodes that compete for the medium, and nodes execute the EDF scheduling algorithm. 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.

Two attempts ([4, 5] and [12, 24, 25]) have been made to migrate the dominance protocol to the wireless context. Both of them modulate the priority bits using on-off keying, encoding a dominant bit as the transmission of a carrier and a recessive bit as silence. In this way a node transmitting a recessive bit can detect a dominant bit and this node will withdraw. Our previous work provided prioritization and was collision-free. Unfortunately it was designed to only operate in a single broadcast domain. The

other approach was designed to operate even in networks with multiple broadcast domains but it only offers a partial solution. A sending node transmits a busy tone on a separate channel and this tone has higher transmission power (or the receivers for the tone are more sensitive) so it has double the range as compared to the range of data transmission. This does not work in the case where two source nodes request to transmit to a receiving node and the two source nodes are close to each other but a communication obstacle keeps them hidden from each other. (This scenario is also discussed in Figure 5 in [19]).

7. Conclusions

We have proposed a MAC protocol that is prioritized and collision-free in networks with multiple broadcast domains; that is, it works even in the presence of hidden nodes. It achieves this without base stations and without relying on out-of-band signals. This work offers a solid foundation for schedulability analysis techniques for wireless networks (for example [7]). We assumed that the interference range is equal to the communication range. Our protocol can be extended to maintain all correctness properties even without this assumption but propagating priority bits a number of hops which is equal to twice the interference range.

References

[1] A. Mok, "Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment", in Electrical Engineering and Computer Science. Cambridge, Mass.: Massachusetts Institute of Technology, 1983.

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

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

[4] B. Andersson and E. Tovar, "Static-Priority Scheduling of Sporadic Messages on a Wireless Channel", In proceeding of the 9th International Conference on Principles of Distributed Systems (OPODIS'05), Pisa, Italy, 2005.

[5] N. Pereira, B. Andersson, and E. Tovar, "Implementation of a Dominance Protocol for Wireless Medium Access", In proceeding of the Proc. of 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06), Sydney, Australia, 2006.

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

[7] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On Real-Time Capacity Limits of Multihop Wireless Sensor Networks", In proceeding of the IEEE International Real-Time Systems Symposium, Lisbon, Portugal, 2004.

[8] 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, pp. 1417-1433, 1975.

[9] X. Yang and N. Vaidya, "Priority Scheduling in Wireless Ad Hoc Networks", Wireless networks.

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

[11] T. Rappaport, Wireless Communications, Principles and Practice. Chapter 3, Section 3.9.1 Log-normal Shadowing: Prentice-Hall, Upper Saddle River, 1996.

[12] T. You, C.-H. Yeh, and H. S. Hassanein, "CSMA/IC: A New Class of Collision-free MAC Protocols for Ad Hoc Wireless Networks", In proceeding of the 28th IEEE International Symposium on Computers and Communication (ISCC'03), 2003.

[13] I. A. a. C. Castelluccia, "Differentiation Mechanisms for IEEE 802.11", In proceeding of the Infocom, 2001.

[14] M. Barry, A. T. Campbell, and V. Andras, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks", In proceeding of the Infocom, 2001.

[15] D.-J. Deng and C. Ruay-Shiung, "A Priority Scheme for IEEE 802.11 DCF Access Method", IEICE Transactions 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, 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, pp. 172-187, 1996.

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

[19] X. Yang and N. Vaidya, "Priority Scheduling in Wireless Ad Hoc Networks", Wireless Networks (WINET) vol. 12, pp. 273-286, 2006.

[20] 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 proceeding of the IEEE Real-Time Systems Symposium, Cancun, Mexico, Cancun, Mexico, 2003.

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

[22] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A Time-Synchronized Link Protocol for Energy Constrained Multi-hop Wireless Networks", In proceeding of the 3rd IEEE COMSOC Conference on Sensor, Mesh and Ad Hoc Communication and Networks (SECON'06), Reston, VA, USA, 2006.

[23] M. Caccamo and L. Y. Zhang, "An Implicit Prioritized Access Protocol for Wireless Sensor Networks", In proceeding of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), Austin, Texas, 2002.

[24] T. You, C.-H. Yeh, and H. S. Hassanein, "A New Class of Collision - Prevention MAC Protocols for Ad Hoc Wireless Networks", In proceeding of the IEEE International Conference on Communications, 2003.

[25] T. You, C.-H. Yeh, and H. S. Hassanein, "BROADEN: An efficient collision-free MAC protocol for ad hoc wireless networks", In proceeding of the 28th IEEE International Conference on Local Computer Networks (LCN'03), 2003.

Appendix A: Design Parameters and Correctness

In this section we discuss the correctness of the protocol and demonstrate how assigning values to the constants *C, E*, *F*, *G, H, T_{CS}, T_{TX}* and *T_{RX}* affect the correctness.

The protocol must satisfy the following three relevant properties.

1. Collision-free. There is no pair of nodes (*X*, *Y*) such that (i) *X* is a 2-neighbor of *Y* and (ii) *X* and *Y* are both in state 15 and (iii) the variable `winner` in *X* and *Y* are `TRUE` simultaneously.
2. Progress. Consider a node *X* that requests to transmit. If for every 2-neighbor node *Y* of *X* it holds that prio(*Y*) > prio(*X*) then node *X* must have the variable `winner` equal to `TRUE`.
3. Prioritization. If a node *X* requests to transmit and node *X* is in state 15 and its variable `winner` is equal to *FALSE* then there is a node *Y* such that (i) *Y* is a 2-neighbor of *X* and (ii) *Y* requests to transmit and (iii) *Y* has higher priority than *X*.

These properties hold if the following protocol constraints (C1 – C5) are respected.

C1) *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 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 and inaccuracy of synchronization, this overlap becomes smaller and smaller with the iterations within the tournament. Hence the last iteration (the worst scenario) of the tournament is considered and the following constraint can be derived:

$$\begin{aligned}&[3H + H + G + (2H + 2G)\times(npriobits - 1)]\times[1 - \varepsilon] - \\ &[3H + G + (2H + 2G)\times(npriobits - 1)]\times[1 + \varepsilon] - \\ &2CLK - L - 2\alpha - \delta > T_{CS} + 2T_{RX}\end{aligned} \quad \textbf{(3)}$$

Equation (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 (T_{CS}).

C2) If a node N_i has perceived a time of silence long enough (F time units) to make the transition 2→3 but other nodes perceive the duration of silence to be less than F time units so far due to different time-of-flights and clock-imperfections, then node N_i needs to wait until all nodes have perceived this long time of silence.

Implications:

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

$$\begin{aligned}&2CLK + L + 2\alpha + (F + T_{RX} + T_{CS})\times 2\varepsilon + \\ &T_{CS} < E\end{aligned} \quad \textbf{(4)}$$

C3) With similar reasoning as for C2, a node which has won the tournament must wait H time units before transmission (this waiting occurs in 14→15) to be sure that all losing nodes have reached State 15.

Implications:

H must satisfy the following constraint:

$$\begin{aligned}&[3H + H + G + (2H + 2G)\times(npriobits - 1)]\times 2\varepsilon + \\ &2CLK + L + 2\alpha + \delta < H\end{aligned} \quad \textbf{(5)}$$

C4) 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 2 to State 3 (the initial idle time period) then all nodes will do it at most *E* time units later. Therefore, the following protocol constraint must be satisfied:

$$\begin{aligned}&\left[\begin{matrix}3H + H + G + (2H + 2G)\times(npriobits - 1) + \\ G + H + C + T_{RX} + T_{CS} + E + T_{CS}\end{matrix}\right]\times[1 + \varepsilon] - \\ &[3H]\times[1 - \varepsilon] + 2CLK + L + 2\alpha + T_{CS} < F\end{aligned} \quad \textbf{(6)}$$

C5) The time interval between two successive dominant bits must be long enough to assure that no node interprets the first dominant bit to be transmitted in the time interval for the second dominant bit.

Implications:

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

$$\begin{aligned}&[3H + H + G + (2H + 2G)\times(npriobits - 2)]\times[1 - \varepsilon] - \\ &[3H + H + (2H + 2G)\times(npriobits - 1)]\times[1 + \varepsilon] \\ &- 2CLK - L - 2\alpha - \delta > 0\end{aligned} \quad \textbf{(7)}$$

C6) *Transition 6→7 cannot occur when a node is transmitting a message (a message transmission is detected as a carrier, if nodes are performing carrier detection)*:

$$3H >= C + T_{TX} + T_{CS} \quad \textbf{(8)}$$

C7) *Transition 15→16 takes, at least, the time to transmit/receive the longest message in the network.*

$$\forall i \in \{i..n\}\, C \geq \max\{C_i\} \quad \textbf{(9)}$$

The values of *C, E*, *F*, *G* and *H* must be selected such as they satisfy (3)-(9). The selection of T_{CS}, T_{RX} and T_{TX} is imposed by the implementation platform chosen.