Technical Report

Collision-Free Prioritized Medium Access Control in Wireless Networks with Hidden Nodes

Björn Andersson

Nuno Pereira

Eduardo Tovar

TR- 060201

Version: 1.0

Date: 6 February 2006

Collision-Free Prioritized Medium Access Control in Wireless Networks with Hidden Nodes

Björn ANDERSSON, Nuno PEREIRA, 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: {bandersson, npereira, emt}@dei.isep.ipp.pt

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

Abstract

We propose a collision-free medium access control (MAC) protocol, which implements static-priority scheduling and works in the presence of hidden nodes. 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 sense the channel while transmitting to the channel. Our protocol is collision-free even in the presence of hidden nodes and it achieves this without synchronized clocks or out-of-band busy tones. In addition, the protocol is designed to ensure that many non-interfering nodes can transmit in parallel and it functions for both broadcast and unicast transmissions.

Keywords: MAC protocol, collision-free, multihop networks, real-time communication

1 Introduction

A fundamental problem in the design of distributed real-time systems is the sharing of a wireless communication channel such that timing requirements are satisfied. This sharing it achieved using a so-called MAC protocol, which solves the distributed mutual exclusion problem among the set of computer nodes that request to transmit. However, solving the mutual exclusion problem is not enough, it is also necessary to take the timing requirements of the message transmission requests into account during run-time. Periodic message transmission requests can be scheduled using static table-driven scheduling. Sporadic [1] and aperiodic [2] messages requests can be scheduled by 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. An intuitive alternative approach is to assign a static priority to a message, and use a medium access control (MAC) protocol that selects for transmission the message with the highest priority. This approach has been used in wired networks [3] and have recently [4-7] been proposed for wireless networks. But unfortunately they do not work in networks with multiple hops because the protocols suffer from several well-known phenomena (which will be described in detail later): *hidden nodes* and *exposed nodes*.

In this paper we develop such a MAC protocol without those drawbacks. Our protocol assumes that message streams are assigned unique priorities (for example: from an algorithm such as Deadline Monotonic [8], from the identifier of the node and message stream id or a combination). Hence, it is possible to design a dominance protocol [9] (similar to the one used in the CAN bus [10]) where nodes contend bit-by-bit. However, the wireless channel requires special care because (i) a node cannot sense and transmit simultaneously, (ii) the presence of hidden nodes can cause collisions and (iii) prioritization can inhibit the parallel transmission of non-interfering nodes if prioritization is implemented poorly. Hence, the dominance protocol requires a redesign – something that we do.

The remainder of this paper is structured as follows. Section 2 discusses related work and their ability to support sporadic messages on a wireless channel. Section 3 presents the system model with our assumptions and terminology. Section 4 highlights key design issues about the protocol; due to space limitations, a precise statement of the protocol is given in Appendix A and Appendix B. Finally, conclusions are drawn in Section 5.

2 Related work

There has been a significant amount of research on MAC protocols aiming at goals such as fairness or high throughput. Here, we will only focus on works relevant to the problem of scheduling sporadic messages with deadlines and on fully distributed algorithms. We will first (in Section 2.1.) review related work in prioritized MAC protocols and then (in Section 2.2.) review related work dealing with the special requirements of MAC protocols in multihop networks.

2.1 Prioritized medium access control

The introduction of the wireless LAN standard IEEE 802.11 stimulated the development of many [11-16] prioritized Carrier Sense Multiple Access (CSMA) MAC protocols and a few of them [11-13] were adopted for the real-time profile IEEE 802.11e. Another technique [17], 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 [17] 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 [11-17] 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. These tables are created at run-time in a distributed fashion

[18] or by a leader [19]. It is also conceivable to use a TDMA template designed before run-time [20] 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 is inefficient. Another approach, Implicit-EDF [21], 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.

The dominance protocol [9] performs a tournament among the messages that request to transmit, and the winner will transmit. It uses global priorities, can schedule sporadic message streams and it is collision-free. 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. Two attempts ([7] and [4-6]) 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 [7] provided prioritization and was collision-free. Unfortunately it was designed to operate in non-multihop networks. The other approach [4-6] was designed to operate in multihop networks but it has several shortcomings. First, it claims to solve the hidden node problem (the hidden node problem will be explained in Section 2.2), but actually 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 [17]). Second, in a network with *n* nodes it can happen that only one node transmits although it would be possible for *n*/3 nodes to transmit in parallel (the authors of [4-6] actually mention this in Figure 3 and Figure 4 in [4], but they do not offer any solution).

2.2 Medium access control protocols for multihop networks

In multihop networks, a source node cannot reach a destination node with a single broadcast and this requires that other nodes relay the message. These networks bring several new challenges to MAC protocols.

Exposed nodes. A node A requests to transmit to node C and another node B, which is close to A, requests to transmit to node D. A and B can transmit simultaneously because their destinations (C and D) are far apart. Although the transmissions will collide at A and B, they will not collide at C and D. The transmission of a sender is exposed to another sender although they do not affect each other. A is said to be an exposed node to B; and similarly, B is said to be an exposed node to A. The existence of exposed nodes reduces the performance but it does not violate the correctness of reception. For this reason, the exposed node problem is not considered to be severe. Some attempts to solve it exists though (MACA in [22] gives a partial solution; the dual busy tone solution [23] is a fully functioning solution, and [24] attempts to find out the sender/receiver roles of different nodes to achieve even more parallelism).

Hidden nodes. A node A requests to transmit to node C and another node B, which is far away from A, requests to transmit to node C as well. A and B cannot sense each other´s transmissions and hence their transmissions may collide at the receiver C. A is said to be a hidden node to B; and similarly, B is said to be a hidden node to A. The existence of hidden nodes jeopardizes the correct reception of a message and hence it has received serious attention from the research community.

The notion of hidden nodes was introduced early and a solution, called the *busy tone solution*, was proposed and analyzed [25]. It that work, it was assumed that nodes communicate with a single base station, and the base station can transmit to all nodes. Whenever a node transmits, the base station hears a carrier wave and then the base station transmits a tone on a narrow band to all nodes. This prevents the hidden node problem. In cases where there is no base station, a similar scheme, called *receiver initiated busy-tone multiple access*, was proposed [26]. Here, an ordinary node (which is not a base station) listens for a carrier wave and when it hears a carrier wave, it transmits a busy tone on a narrow band channel when it is receiving data (or has heard one request to send), hence informing other senders that they should not transmit. Common to these two solutions ([25] and [26]) is that they rely on a separate channel. The MACA protocol [22] was designed to remove that limitation. In MACA, a node that requests to send a data packet, sends a `Request-To-Send` (RTS) packet on the channel (there is only one) and then waits for the receiving node to transmit a `Clear-To-Send` (CTS) packet to the sender's id. Only then the node can transmit; all other nodes are not permitted to transmit. Certain parameters were left unspecified in MACA so based on that, a more well-defined protocol was proposed MACAW [27]. A common theme of these protocols is that collisions may occur, even from non-hidden nodes.

To rectify this situation, a MAC protocol without collisions was designed and given the name FAMA-NCS [28]. It is based on (i) a RTS-CTS dialogue but also on carrier sensing before transmission of RTS and CTS and (ii) carefully selected lengths of these messages and intervals of silence. This ensures that collisions between an RTS and a DATA packet cannot occur. Schemes with only RTS-CTS (and with no carrier sensing) are explored as well. In order to ensure that a DATA packet does not collide with an RTS, it is necessary that the receiving node sends a number of CTS which is large enough to be sure that all neighbor nodes of the receiving node have received collision-free CTS. This implies that if carrier sensing is not used then a sender needs to wait for a long time before it can transmit a data packet. Hence, it was concluded that carrier sensing is necessary to implement collision-free MAC protocols efficiently [28]. (A related paper [29] deals only with fully connected networks; that is, there are no hidden nodes). The CTS packet has the role of the receiver informing the sender that it is OK to transmit. Another way to achieve the same effect is to let the receiver be idle for a time duration if it is OK to transmit; if the receiver (or other nodes) detected a collision between RTS packets then the receiver (or other nodes) transmits a jamming signal. If the Tx-Rx turnaround time is larger than the time-of-flight (which is common), and there is only one sender and one receiver, and there are no other nodes in the network, then this solution does not work.

Another solution to achieve collision-free transmission of DATA packets in the presence of hidden nodes is to use the usual RTS-CTS dialog with no carrier sensing (just like MACA) but also use two busy tones on separate channels. A tone (called BTt) is transmitted by the sender when it transmits a DATA packet and another tone (called BTr) is transmitted when the receiver receives a DATA packet. The author proved that this solution is collision-free [30]; it offers excellent throughput [23] and it also has the advantage of solving the problem of exposed nodes. But, unfortunately it has the disadvantage of requiring extra bandwidth (for the two busy tones) and specialized hardware (to detect those busy tones).

The FAMA protocol [28] ensures that DATA packets are collision-free but it is still possible for RTS packets to collide if two RTS packets are transmitted from two different nodes and the difference between their transmission times is less than the time of flight. In very high loads, this can cause the channel to only transmit colliding RTS packets and hence the throughput of DATA packets drops to zero. To rectify this, a protocol with collision resolution was proposed [31, 32]. It works as follows. When a node transmits a RTS and it does not hear a CTS it concludes that the RTS collided. Then, only half of all nodes are permitted to transmit the next RTS; the other half of nodes must wait. These groups are selected based on the ids of the nodes. By repeating this procedure (called *tree-splitting*) eventually only one node transmits the RTS.

Priority inversion. Priority inversion means that a lower-priority message is transmitted before a higher-priority message. This problem is well-known in uniprocessor scheduling (where tasks are prioritized rather than messages) [33, 34]. In real-time communication, the same problem can occur due to the non-preemptive nature of communication, so it is an issue in multihop networks too. In multihop networks, yet another, but similar problem (called *pseudo priority inversion*) can occur too [35]. To understand this, consider three sending nodes S1, S2 and S3, and two receiving nodes R1 and R2. S1 requests to send to R1; S2 and S3 request to send to R2. S2 is within the range of R1. S1 and S2 compete for R1. S2 and S3 compete for R2. S1 has higher priority than S2, which in turn has higher priority than S3. S1 will transmit because it has the highest priority. S2 cannot transmit because it lost the competition against S1. S3 will transmit although it competes with S2 which has higher priority; this is because S2 lost the competition against S1. It was also shown that this effect can cause chains where a message is dependent on another message arbitrarily far away.

We conclude in this section that there is only one [4-6] prioritized collision-free MAC protocol which works in the presence of hidden nodes. As already mentioned, this solution is only a partial solution to the hidden node problem and it can cause a large loss in the degree of parallelism. We need a MAC protocol without those drawbacks.

3 System model

We consider computer nodes; which we often call simply nodes. On a node, applications make requests to transmit and applications need to receive a subset of the messages that a node hears. The protocol does not know about the origin of messages; two different messages may belong to the same sporadic message stream with a minimum inter-arrival time or they may not.

Priorities. Messages 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. Let prio[0..npriobits-1] be an array of bits representing the priority of a message. The most significant bit is prio[0].

Propagation. We assume that the connectivity is described by a graph, such that when a node broadcasts, all of its neighbors hear it. Hence, there are hidden nodes. Radio propagation may be different in different directions. It may even happen that in one direction a node that is far away can be reached, but a node that is closer and in the same direction cannot be reached. We assume symmetric links though, and we assume that connectivity does not change with time. We say that a node j is a 2-neighbor to node i if either (i) node j is a neighbor to node i or (ii) node j is a neighbor to node k and node k is a neighbor to node i. The time-of-flight between nodes i and j is unknown but it is non-negative and there is an upper bound α on the time-of-flights. If two or more nodes transmit data and there exists a node which receives two or more of these data transmissions, then we say that this node has suffered from a collision. Our protocol ensures that there is no such node that suffers from a collision. However, there may be an area which is reached by more than one transmitter but there is no receiving node in that area, so it is not defined as a collision. It may also happen that a node receives non-modulated carrier waves that are not part of the data packet from two or more different nodes. These non-modulated carrier waves are a part of the MAC protocol and the protocol is designed for this. Since they are not data packets, this is not considered to be a collision either. We assume that whatever signals a node transmits, all of its neighbor nodes receive exactly one copy of the signal; that is, there is no multipath distortion and there is no noise.

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 $[1-\varepsilon, 1+\varepsilon]$, $0< \varepsilon <1$. We let *CLK* denotes the granularity of the clock. We assume that the clock does never “wrap-around”.

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 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. We let E,F,G,H and R denote time-out parameters.

Our MAC protocol treats unicast and broadcast in the same way and it has no special support for acknowledgement. We do not assume any particular modulation technique or coding scheme for the data bits, but we assume that the transmission of data bits takes at most F-G time units. We assume that nodes can transmit a carrier wave, and that all nodes are able to detect that carrier wave if they do not transmit themselves. A node needs *TFCS* (Time-For-Carrier-Sensing) time units to detect that a carrier wave was transmitted. A node can sense other transmissions only if the node does not transmit. It needs $turnaround_{RxTx}$ time units to switch from transmission to reception or vice-versa. All nodes have the same $turnaround_{RxTx}$ time.

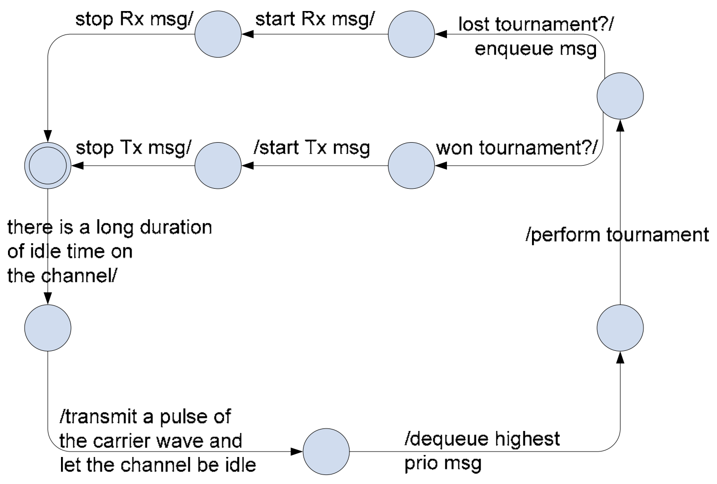

stop Rx msg/
start Rx msg/
lost tournament?/ enqueue msg
stop Tx msg/
/start Tx msg
won tournament?/
there is a long duration of idle time on the channel/
/perform tournament
/transmit a pulse of the carrier wave and let the channel be idle
/dequeue highest prio msg

Fig. 1. Overview of the protocol.

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 ◉), 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 discusses informally how to design a dominance protocol on a wireless channel and Section 4.2 extends it to deal with hidden nodes. In Section 4.3 we will see that the protocol may reduce the amount of parallelism significantly. For this reason, we will extend the protocol to ensure that the amount of parallelism is high; we will prove this. Due to space limitations, we have put some material in Appendices. Appendix A puts all these ideas together and presents formally a protocol which is prioritized and collision-free in the presence of hidden nodes, and which has a high degree of parallelism. It depends on certain time-out parameters, E,F,G,H and R. In Appendix B we select values of these time-out parameters.

4.1 Dominance on a wireless channel

Dominance protocols operate such that nodes transmit their priority bit-by-bit starting with their most significant bit. If a node transmits a recessive bit and hears a dominant bit then the node withdraws. Finally, all nodes but one has withdrawn; the node which has not withdrawn is called a *winner* and it transmits the message. In wired networks, dominance is straightforward to implement because a node can transmit and receive simultaneously and hence if a node has a recessive bit it can still sense a dominant bit. In wireless networks, this sensing simultaneous with transmitting requires more thought. We can observe that a node does not need the full ability of sensing simultaneously with transmission. A node only needs the ability to sense simultaneously with transmission when a node transmits a recessive bit. This can be achieved by encoding a recessive bit as no transmission at all, and by encoding a dominant bit as a non-modulated carrier. This makes it possible to design a dominance protocol on a wireless channel (this approach was used in our previous work [7] and also in [4-6]). Before the tournament starts, it is necessary that nodes agree on a common reference point in time. In [7] this was achieved by listening for a long period of silence. Then, if a pulse is heard the timer is reset; if it is not heard then it waits for a short while and if it still had not heard a pulse then it transmits a pulse and resets its timer. The solution in [4-6] however, is based on that all nodes perform periodic beaconing with randomized decision on whether to transmit the beaconing. This gives a low overhead but unfortunately, it can happen that no node transmits the beaconing signal.

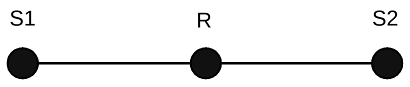


Fig. 2. Prioritization in the presence of hidden nodes.

4.2 Dealing with hidden nodes

Consider Figure 2. Consider the case that S1 has higher priority than S2 and that both of them request to broadcast. S1 does not hear S2, and similarly, S2 does not hear S1. There may be many nodes but for now let us only consider one node R which can hear both S1 and S2. Hence, S1 and S2 compete for R but, unfortunately, S1 and S2 do not hear each other. Hence, it is necessary that R conveys S1's priority to S2 and, similarly, it is necessary that R conveys S2's priority to S1.

The researchers of [4-6] proposed that the hidden nodes can be dealt with by sending the carrier on a separate channel and the transmitter on this separate channel has more transmission power (or the receivers of this channel are more sensitive). This works in free-space but it does not work when the nodes are hidden from each other because an obstructive physical object is located between them.

Let us now design a correct solution. We have to ensure that carrier sensing can reach two hops and that transmission of data still only reaches one hop. A natural way of doing this is that a sender transmits its priority bit and then waits for a short duration. During this short duration, a node will transmit a carrier if it hears a carrier. In this way, a dominant bit will be propagated to the other sender. It is important, however, that priorities are only propagated two hops. (If priorities were propagated three or more hops then a dominant node *i* could win over another node *j* which is so far way that if node *i* and node *j* transmitted simultaneously then there is no receiver that receives both broadcasts. This would lower the amount of parallelism and hence performance, so this is undesirable.) If any node which received a dominant bit would retransmit a dominant bit then a dominant bit would propagate arbitrarily far. On the other hand, if the priority was propagated less than two hops; that is one hop, then the hidden node problem can occur and hence cause collisions. Hence, it is important that the priority is propagated exactly two hops from the sender.

This requires three modifications to the dominance protocol [7]. First, before the tournament starts it is necessary that nodes synchronize to agree on the time when the tournament should start. When priority bits are transmitted it is necessary that nodes agree on a common reference point in time. In CAN this is performed by letting a node wait for a long period of silence and our protocol does the same (as indicated in Figure 1). If the node hears that the carrier makes a transition from idle to busy then it resets its timer. Otherwise the node waits for a short while extra and if it still has not heard a carrier then it transmits a carrier and resets its timer. This synchronization works only in single-hop networks. In our multihop network, the protocol must retransmit the pulse of carrier. To ensure that a pulse of the carrier wave is propagated exactly two hops, we let senders transmit a pulse of the carrier wave with one duration and when any node (sender or receiver) hears it for such a long duration (or knows about it) they retransmit the pulse but with another (shorter) duration.

The second modification to the protocol is that priorities need to be retransmitted. When a bit in the tournament is transmitted it will wait for a short while before the next bit is transmitted, any nodes which hear the previous transmission, retransmit it just afterwards. These two pulses have the same duration; there is no need for pulses of different duration because nodes have already synchronized.

Third, when a node has won the tournament and transmits the data bits, all nodes one hop away from the sender will hear about the transmission. However, nodes two hops away will experience silence. If the transmission of databits would be long then other nodes may experience a long period of silence and start the tournament. If that would happen then a node may go through the tournament and it may transmit pulses of carrier waves which may collide with the ongoing transmission. However, recall (from Section 3) that we assume that each data packet has a maximum transmission time (given by the parameters *F-G*). This assures that no node will interfere with the winning node when it transmits its data bits.

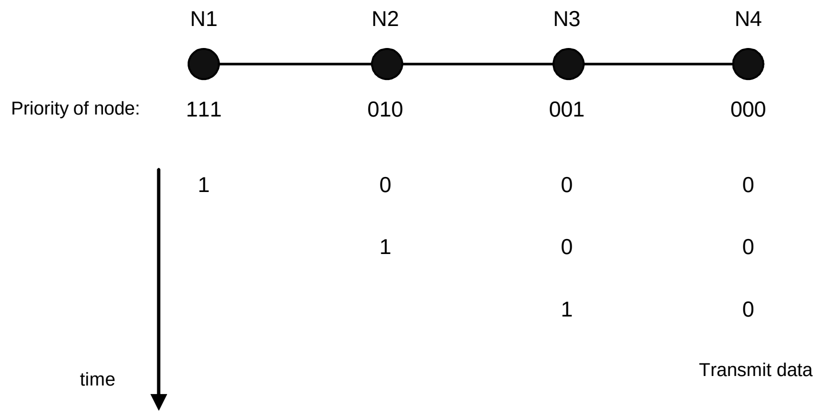


Fig. 3. Some prioritization protocols can cause a very low amount of parallelism.

4.3 Dealing with "multihop competing"

We want a MAC protocol that maximizes the number of nodes that can transmit simultaneously subject to the constraints that priority scheduling is achieved and that no collisions occur. We will now define these concepts.

Consider a set of nodes *N*. There is a subset *T* of these nodes that request to transmit. A MAC protocol is defined as follows:

Definition. MAC protocol. A MAC protocol selects at every moment the set *S* of nodes which are transmitting a data packet. *S* is a subset of *T*. This set *S* must satisfy the mutual exclusion property.

Definition. Mutual Exclusion Property. The set *S* satisfies the mutual exclusion property if there is no node *i* in *N* such that *i* has two or more neighbors which are in *S*.

This guarantees that every node will hear at most one broadcast. Note that at certain times, the number of elements in *S* may be small. It may even happen that *S* is empty at certain times. This can happen when nodes do not have any outgoing messages. It can also happen that nodes have outgoing messages but at this moment the tournament is performed.

Since we are studying prioritization we will now define it. A MAC protocol is prioritized if *S* satisfies the prioritization property at every time. We will say that a node *i* has priority *prio(i)* when we mean that the highest priority message at node *i* which request to transmit has priority *prio(i)*. We assume that if the number *prio(i)* is low then the priority is high.

Definition. Prioritization Property. The set *S* satisfies the prioritization property if both conditions below are satisfied:

1. For every pair of nodes *i* and *j* such that *i* and *j* are neighbors and both node *i* and node *j* have outgoing messages (that is $i \in T$ and $j \in T$) and $prio(i) < prio(j)$ it holds that exactly one of the following conditions are true: (i) $i \in S$ and $j \notin S$; (ii) ($j \in S$ and $i \notin S) \Rightarrow \exists k$:(($k \in S$ and node *k* is a neighbor of node *i* and $prio(k) < prio(i)$).
2. Consider a node *i* and every neighbor node *j* which is a neighbor to *i*. If $prio(j) > prio(i)$ then node *i* must be in *S*.

Case (i) in Condition 1 represents normal prioritization, and case (ii) in Condition 1 represents the case of pseudo priority inversion. Condition 2 states that if a node requests to transmit and it has higher priority than all its neighbors, then this node must transmit.

It is easy to design a MAC protocol that satisfies the mutual exclusion property and the prioritization property; a protocol which only transmits if Condition 2 is true does that. Unfortunately, such a protocol suffers from a problem called "multihop competing" [4], a situation where |*S*| is unnecessarily small.

Consider Figure 3. The nodes N1, N2, N3 and N4 request to transmit and we consider the case that there are no nodes that only receive. The topology and priority is given in Figure 3. In the first round of the tournament, N1 transmits a recessive bit "1" and the others transmit a dominant bit "0". In particular N2 transmits a dominant bit and hence N1 loses. In the next bit, N2 loses and in the final, third bit, N3 loses. We can see that N4 wins the tournament and it will transmit data. This is the behavior of one MAC protocol that satisfies the prioritization property and mutual exclusion property. Unfortunately, it does not maximize |*S*| because it would have been possible for N1 and N4 to transmit. Based on the behavior in Figure 3, we will now discuss how to design a MAC protocol that maximizes |*S*|.

In the example in Figure 3, the MAC protocol has to (according to Condition 2 in the definition of prioritization) select N4 as one of the nodes to transmit. Due to the prioritization property, N3 cannot be selected as a member in *S*. Due to the prioritization property either N1 or N2 could be a member in *S*, but the mutual exclusion property requires that N2 is not a member in *S*. Hence N1 and N4 can be members in *S*, so it is possible to transmit two messages in parallel. A MAC protocol however does not know the entire topology and all the priorities of the messages; it only knows about the priorities of its neighbors. N1, N2 and N3 have similar relationships to their neighbors; they all have a neighbor with higher priority. It is problematic if a MAC protocol makes the same decision for all nodes because if N1, N2 and N3 transmit then the mutual exclusion property is violated, and if none of them transmit then |*S*| = 1, which is undesirable as well. Hence, N1, N2 and N3 must make different decisions although their environments appear to be the same. We must find one of the nodes N1, N2 or N3 which is different from the other. A more careful look reveals however, that N1 and N2 experience different things. N1 loses in the first round (called round 0) and N2 loses in second round (called round 1). By analyzing the behavior in Figure 3 we can see that the source of the low degree of parallelism is that a node *i* can transmit a dominant bit and hence cause another node *j* to lose the tournament but then later node *i* also loses. Hence, node *j* lost unnecessarily. If a node loses late in the tournament then it has many opportunities to cause other nodes to lose. It seems to be a good idea to change the protocol so that first a node knows in which round it will lose and then the tournament is performed in reverse order so that the nodes that lost late will now lose early. Figure 4 describes such an algorithm.

Let us apply the algorithm in Figure 4 on the example in Figure 3. Lines 1 to 10 in the tournament initialize variables. Lines 11 to 14 just run the normal tournament. In line 13, we call a subroutine (described to the right in Figure 4) which takes care of the contention of one single bit. After line 14, node 1 has `lostindex` = 0; node 2 has `lostindex` = 1, node 3 has `lostindex` = 2 and node 4 has `lostindex` =INFINITY (it did not lose). In lines 15 to 19, we observe that a node which did not request to transmit is not a potential winner of the tournament. A node which lost in the last round in the first part of the tournament (lines 11 to 14) cannot be a potential winner either. However, all other nodes are potential winners. That is node 1, node 2 and node 4 are potential winners (they have `winner=TRUE)`. Node 3 lost in the last bit so it is not a potential winner (it has `winner=FALSE`). In lines 20 to 23 we run the tournament in the reverse order. Observe that we do not consider the bit `npriobits`-1 in this part of the tournament; we start with the bit `npriobits`-2.

```
1  begin tournament
2    lostindex:=INFINITY
3    if node request to transmit
4      then
5        winner:=TRUE
6        priobits represents the
7          priobits of message
8    else
9        winner:=FALSE
10   endif

11   for index:=0 to npriobits-1 do
12     active:=TRUE
13     bit_contention(index)
14   endfor

15   if lostindex/=npriobits-1 AND node
            requested to transmit then
16     winner:=TRUE
17   else
18     winner:=FALSE
19   endif

20   for index:=npriobits-2 downto 0 do
21     active:=index<=lostIndex
22     bit_contention(index)
23   endfor

24   if winner then
25     action:=TRANSMIT
26   else
27     action:=IDLE
28   endif
29  end tournament
```

```
1  procedure bit_contention(index: integer)
2  begin
3    exec:=0
4    wait until all nodes have exec=0
5    if priobits[index]=DOMINANT AND
       winner=TRUE AND active=TRUE then
6       transmit DOMINANT
7       heardDOMbit:=FALSE
8    else
9      if transmission of DOMINANT
      bit is detected then
10       heardDOMbit:=TRUE
11     else heardDOMbit:=FALSE endif
12   endif

13   exec:=1
14   wait until all nodes have exec=1
15   if heardDOMbit then
16       transmit DOMINANT
17   else
18       if transmission of DOMINANT
           bit is detected then
19         heardDOMbit:=TRUE
20       else heardDOMbit:=FALSE endif
21   endif

22   exec:=2
23   wait until all nodes have exec=2
24   if priobits[index]=RECESSIVE AND
25     winner=TRUE AND active=TRUE AND
26     heardDOMbit=TRUE then
27       winner:=FALSE
28       lostIndex:=index
29   endif
30  end procedure
```

Fig. 4. A tournament that solves the "multihop competition problem".

Let us now consider the iteration when `index`=1. Line 22 will be executed by node 2, node 3 and node 4. However, only node 2 and node 4 have both `winner`=`TRUE` and `active`=`TRUE`. Hence, they are the only ones that are allowed to transmit a dominant bit (see Line 5 in the procedure `bit_contention`). Node 2 transmits "1" and node 4 transmits "0". Node 3 has `active=FALSE` so it does not transmit dominant bit but it propagates the priority bits of node 2 and node 4 (see Lines 13-21 in the procedure `bit_contention`). Hence, node 2 loses because it hears a dominant bit. Let us now consider the iteration when `index`=0. Line 22 will be executed by all nodes again but only node 1 and node 4 have both `winner=TRUE` and `active=TRUE`. Node 1 will transmit "1" and node 4 will transmit "0". At line 24, we have two nodes, node 1 and node 4, with `winner=TRUE`, so they will transmit. In this example, this is the maximum degree of parallelism. As we can see by Theorem 1 below, this is not a coincidence.

Theorem 1. If all nodes request to transmit at the same time then there is no other prioritized MAC protocol that obtains a larger |*S*| than the algorithm in Figure 4.

Proof: We will prove Theorem 1 using mathematical induction.

Base case. Theorem 1 is true for *npriobits=1*.

Proof. This is clearly true because with *npriobits=1* the second part of the tournament does not exist.

Induction step. For every $x \geq 2$ it holds that:

Theorem 1 is true for *npriobits=x-1*⇒Theorem 1 is true for *npriobits=x*.

Proof of the induction step. We can prove this using contradiction. Let us assume that the induction step was incorrect. Then the left hand side of the implication must be true but there must exist a graph and a priority assignment such that the expression to the right of the implication in the induction step is false. We will now reason about this. There must exist a connectivity graph *G* and a priority assignment to nodes *P* (where the priorities assigned have *x* bits) such that $|S(G,P,A)|<|S(G,P,OPT)|$. *S* is the set of nodes that are selected and it depends on the graph, priority assignment to nodes and on the algorithm. *A* is our algorithm. *OPT* is an algorithm that maximizes |*S*|. We let *N(G)* denote the nodes in *G* and *E(G)* denote the edges in *G*. Let *notS* denote the set $|N(G)|\backslash S$. We can reason about the nodes that necessarily must be in *notS* . Let us consider a node *i* with `lostindex`=*x*-1 at line 15. (Recall that *npriobits=x*.) If this was the case then there was another 2-neighbor *j* such that every node *k* which is a 2-neighbor with *j* has *prio(j)* < *prio(k)*. Based on the definition of the property of prioritization, node *j* must be in *S* and consequently (due to the prioritization property), node *i* must be in *notS*. We have now obtained the conclusion that:

All nodes with `lostindex` = *x*-1 *at line 15 must be a member of notS.*

We can observe that those nodes with `lostindex` = *x*-1 at line 15 must be a member of *notS* for every prioritized MAC protocol. We will now create a new graph *G´* such that *N(G´)=N(G)\{nodes with lostindex=x-1}* and *E(G´)=E(G)\{edges from or to a node with lostindex=x-1}*. Hence *G´* is a graph where we have subtracted the nodes that necessarily lose. Based on this, we obtain

$$|S(G',P,A)| = |S(G,P,A)| \text{ and } |S(G',P,OPT)| = |S(G,P,OPT)|$$

and consequently:

$$|S(G',P,A)| < |S(G',P,OPT)|$$

The priority assignment *P* is the same as before but now *P* may assign priorities to nodes which do not exist. This is no problem. We will now create a new priority assignment *P´*. The priorities in *P´* have only *npriobits*-*1* bits instead of the *npriobits* which *P* had. Consider a node A in *G´*. Let us assume that *P* assigned node A the priority *Q*. We define *P´* as follows. *P´* assigns node A the priority *Q* but without the least significant bit of the priority. This least significant bit does not change the number of nodes that were selected anyway, so we obtain:

$$|S(G',P',A)| = |S(G',P,A)| \text{ and } |S(G',P',OPT)| = |S(G',P,OPT)|$$

Applying this to the inequality $|S(G',P,A)| < |S(G',P,OPT)|$ that we obtained earlier in this proof, yields:

$$|S(G',P',A)| < |S(G',P',OPT)|$$

But this is a contradiction to the expression to the left of the implication in the claim of the induction step. Hence, the induction step is true.

Using the base case and the induction step proves the Theorem 1. □

Based on Theorem 1, we can conclude that it is possible to achieve a prioritized MAC protocol that functions in the presence of hidden nodes and achieves a high degree of parallelism.

5 Conclusions

We have presented a MAC protocol for sporadic messages on a wireless channel in the presence of hidden nodes. The protocol is collision-free, supports a large number of priority levels and does not require synchronized clocks. In order to solve the hidden node problem, the protocol used two new ideas: (i) synchronization across node that are not neighbors and (ii) propagation of priority bits. In order to maximize the number of parallel transmissions, the protocol used the new idea of a reverse tournament.

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] T. Abdelzaher and C. Lu, "Schedulability Analysis and Utilization Bounds for Highly Scalable Real-Time Services," presented at IEEE Real-Time Technology and Applications Symposium, TaiPei, Taiwan, 2001.

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

[4] T. You, C.-H. Yeh, and H. S. Hassanein, "CSMA/IC: A New Class of Collision-free MAC Protocols for Ad Hoc Wireless Networks," presented at 8th IEEE International Symposium on Computers and Communication, 2003.

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

[6] T. You, C.-H. Yeh, and H. S. Hassanein, "BROADEN: An efficient collision-free MAC protocol for ad hoc wireless networks," presented at IEEE International Conference on Local Computer Networks, 2003.

[7] B. Andersson and E. Tovar, "Static-Priority Scheduling of Sporadic Messages on a Wireless Channel," presented at International Conference on Principles of Distributed Systems (OPODIS´05), Pisa, Italy, 2005.

[8] J. Leung and J. Whitehead, "On the Complexity of Fixed-priority Scheduling of Periodic Real-Time Tasks," *Performance Evaluation, Elsevier Science*, vol. 22, pp. 237-250, 1982.

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

[10] Bosch, "CAN Specification, ver. 2.0, Robert Bosch GmbH, Stuttgart," 1991.

[11] I. Aad and C. Castelluccia, "Differentiation Mechanisms for IEEE 802.11," presented at Infocom, 2001.

[12] M. Barry, A. T. Campbell, and V. Andras, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks," presented at Infocom, 2001.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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," presented at IEEE Real-Time Systems Symposium, Cancun, Mexico, 2003.

[19] H. Li, P. Shenoy, and K. Ramamrithan, "Scheduling Communication in Real-Time Sensor Applications," presented at IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, Canada, 2004.

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

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

[22] P. Karn, "MACA - A New Channel Access Method for Packet Radio," presented at ARRL/CRRL Amateur Radio 9th Computer Networking Conference, 1990.

[23] Z. J. Haas and J. Deng, "Dual busy tone multiple access (DBTMA)--A multiple access control scheme for ad hoc networks," *IEEE Transactions on Communications*, vol. 50, pp. 975 - 985, 2002.

[24] A. Acharya, A. Misra, and S. Bansal, "MACA-P : A MAC for Concurrent Transmissions in Multi-Hop Wireless Networks," presented at IEEE International Conference on Pervasive Computing and Communication, Los Alamitos, CA, 2003.

[25] 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.

[26] C.-S. Wu and V. O. K. Li, "Receiver-initiated busy-tone multiple access in packet radio networks," presented at ACM workshop on Frontiers in computer communications technology, Stowe, Vermont, United States, 1987.

[27] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LAN's," presented at Applications, Technologies, Architectures,and Protocols for Computer Communication, London, United Kingdom, 1994.

[28] C. L. Fullmer and J. J. Garcia-Luna-Aceves, "Solutions to hidden terminal problems in wireless networks," presented at ACM SIGCOMM '97, Cannes, France, 1997.

[29] C. L. Fullmer and J. J. Garcia-Luna-Aceves, "Floor acquisition multiple access for packet-radio networks," presented at Proc. ACM SIGCOMM 95, Cambridge, MA, 1995.

[30] Z. J. Haas, J. Deng, and S. Tabrizi, "Collision-free Medium Access Control Scheme for Ad Hoc Networks," presented at Proc. of IEEE Military Communications Conference (MILCOM '99), Atlantic City, NJ, USA, 1999.

[31] R. Garcés and J. J. Garcia-Luna-Aceves, "Floor acquisition multiple access with collision resolution," presented at International Conference on Mobile Computing and Networking, Rye, New York, United States, 1996.

[32] R. Garcés and J. J. Garcia-Luna-Aceves, "Collision avoidance and resolution multiple access with transmission queues," *Wireless Networks*, vol. 5, pp. 95 - 109, 1999.

[33] R. Rajkumar, *Synchronization in real-time systems : a priority inheritance approach*. Boston: Kluwer Academic Publishers, 1991.

[34] B. W. Lampson and D. D. Redell, "Experience with processes and monitors in Mesa," *Communications of the ACM*, vol. 23, pp. 105 - 117, 1980.

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

[36] *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*.

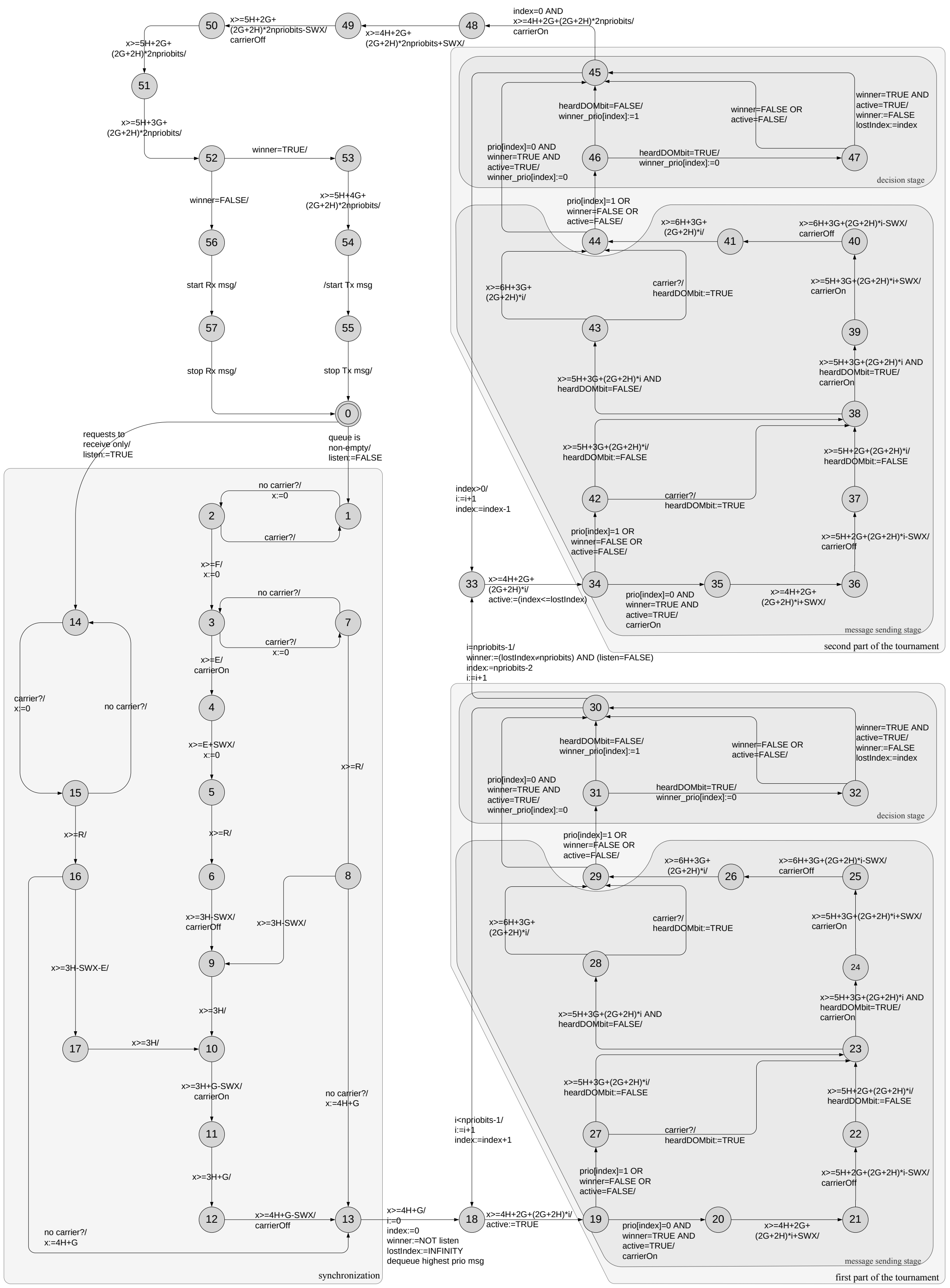
[37] "Broadband Radio Access Networks(BRAN);HIPERACCESS; PHY protocol specification," ETSI TS 101 999 v1.1.1 (2002-04) ed.

Appendix A: Precise statement of the proposed protocol

Based on the discussions in Section 4.1-4.3, we will now formally present our protocol. Figure 5 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 56. State 0 is the initial state. Each node has the following variables: a clock `X`, an integer `i` within the range 0… 2*npriobits*-1, an integer `index` within the range 0…*npriobits*-1, an integer `prio` occupying *npriobits* bits, an integer `winner_prio` occupying *npriobits* bits and two boolean variables `active` and `winner`. We let `winner_prio[i]` denote the bit `i` in the variable `winner_prio`. Analogous 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, SWX* and *R* are constants used for time-outs, whose values we will choose later.

The states 0-17 in Figure 5 establish a common reference point in time between all nodes that requests to transmit. The states 14, 15, 16 and 17 establish the same references point for nodes that only request to receive; this reference point is the same as the reference point of the nodes that request to transmit. The transition 3→4 is designed to make the protocol robust to clock inaccuracies. The nodes go through the tournament in states 18-47. In the beginning, all nodes the request to transmit have the potential to win (and hence their variable winner is TRUE) but after the tournament, only one node is a winner and this node will enter the state 53 and transmit. The states 18-32 perform the first part of the tournament that is intended to find out in which turn a node lost. During the tournament, nodes contend bit-by-bit, starting with the most significant bit. The states 32-47 perform the second part of the tournament. The first part of the tournament and the second part of the tournament are very similar; they differ mainly in that in the second part (i) the contention between priority bits starts at the least significant bit and goes to the most significant bit and (ii) the conditions for when a node is active differ.(the variable `active` takes care of this.) In the first part or the tournament, a node is always active; in the second part of the tournament, a node is only active is `index` ≤ `lostindex`. If a node is not active then (i) it cannot transmit a dominant bit and (ii) if it is recessive and heard a dominant bit then it cannot lose.



50
49
48
x>=5H+2G+(2G+2H)*2npriobits-SWX/ carrierOff
x>=4H+2G+(2G+2H)*2npriobits+SWX/
index=0 AND x>=4H+2G+(2G+2H)*2npriobits/ carrierOn
x>=5H+2G+(2G+2H)*2npriobits/
51
x>=5H+3G+(2G+2H)*2npriobits/
52
winner=TRUE/
53
winner=FALSE/
x>=5H+4G+(2G+2H)*2npriobits/
56
54
start Rx msg/
/start Tx msg
57
55
stop Rx msg/
stop Tx msg/
0
requests to receive only/ listen:=TRUE
queue is non-empty/ listen:=FALSE
no carrier?/ x:=0
carrier?/
2
1
x>=F/ x:=0
no carrier?/
carrier?/ x:=0
3
7
14
x>=E/ carrierOn
4
carrier?/ x:=0
no carrier?/
x>=E+SWX/ x:=0
x>=R/
15
5
x>=R/
x>=R/
16
6
8
x>=3H-SWX/ carrierOff
x>=3H-SWX/
x>=3H-SWX-E/
9
x>=3H/
17
x>=3H/
10
x>=3H+G-SWX/ carrierOn
no carrier?/ x:=4H+G
11
x>=3H+G/
12
x>=4H+G-SWX/ carrierOff
13
no carrier?/ x:=4H+G
synchronization
x>=4H+G/ i:=0 index:=0 winner:=NOT listen lostIndex:=INFINITY dequeue highest prio msg
18
x>=4H+2G+(2G+2H)*i/ active:=TRUE
19
prio[index]=0 AND winner=TRUE AND active=TRUE/ carrierOn
20
x>=4H+2G+(2G+2H)*i+SWX/
21
x>=5H+2G+(2G+2H)*i-SWX/ carrierOff
22
x>=5H+2G+(2G+2H)*i/ heardDOMbit:=FALSE
23
x>=5H+3G+(2G+2H)*i AND heardDOMbit=TRUE/ carrierOn
24
x>=5H+3G+(2G+2H)*i+SWX/ carrierOn
25
x>=6H+3G+(2G+2H)*i-SWX/ carrierOff
26
x>=6H+3G+(2G+2H)*i/
29
prio[index]=1 OR winner=FALSE OR active=FALSE/
27
carrier?/ heardDOMbit:=TRUE
x>=5H+3G+(2G+2H)*i/ heardDOMbit:=FALSE
x>=5H+3G+(2G+2H)*i AND heardDOMbit=FALSE/
28
x>=6H+3G+(2G+2H)*i/
carrier?/ heardDOMbit:=TRUE
prio[index]=1 OR winner=FALSE OR active=FALSE/
message sending stage
30
31
32
heardDOMbit=FALSE/ winner_prio[index]:=1
winner=FALSE OR active=FALSE/
winner=TRUE AND active=TRUE/ winner:=FALSE lostIndex:=index
prio[index]=0 AND winner=TRUE AND active=TRUE/ winner_prio[index]:=0
heardDOMbit=TRUE/ winner_prio[index]:=0
decision stage
first part of the tournament
i<npriobits-1/ i:=i+1 index:=index+1
i=npriobits-1/ winner:=(lostIndex=npriobits) AND (listen=FALSE) index:=npriobits-2 i:=i+1
33
x>=4H+2G+(2G+2H)*i/ active:=(index<=lostIndex)
34
prio[index]=0 AND winner=TRUE AND active=TRUE/ carrierOn
35
x>=4H+2G+(2G+2H)*i+SWX/
36
x>=5H+2G+(2G+2H)*i-SWX/ carrierOff
37
x>=5H+2G+(2G+2H)*i/ heardDOMbit:=FALSE
38
x>=5H+3G+(2G+2H)*i AND heardDOMbit=TRUE/ carrierOn
39
x>=5H+3G+(2G+2H)*i+SWX/ carrierOn
40
x>=6H+3G+(2G+2H)*i-SWX/ carrierOff
41
x>=6H+3G+(2G+2H)*i/
44
prio[index]=1 OR winner=FALSE OR active=FALSE/
42
carrier?/ heardDOMbit:=TRUE
x>=5H+3G+(2G+2H)*i/ heardDOMbit:=FALSE
x>=5H+3G+(2G+2H)*i AND heardDOMbit=FALSE/
43
x>=6H+3G+(2G+2H)*i/
carrier?/ heardDOMbit:=TRUE
prio[index]=1 OR winner=FALSE OR active=FALSE/
message sending stage
index>0/ i:=i+1 index:=index-1
45
46
47
heardDOMbit=FALSE/ winner_prio[index]:=1
winner=FALSE OR active=FALSE/
winner=TRUE AND active=TRUE/ winner:=FALSE lostIndex:=index
prio[index]=0 AND winner=TRUE AND active=TRUE/ winner_prio[index]:=0
heardDOMbit=TRUE/ winner_prio[index]:=0
decision stage
second part of the tournament

Fig. 5. Details of the protocol.

Fig. 6. An illustration of the carrier wave transmitted assuming *npriobits* = 2 and the priority of the requested message is 2. This illustration is simplified; it ignores the many smaller time intervals, like *E*, *SWX* and *R*. For a full understanding of the protocol, see the automata in Figure 5.

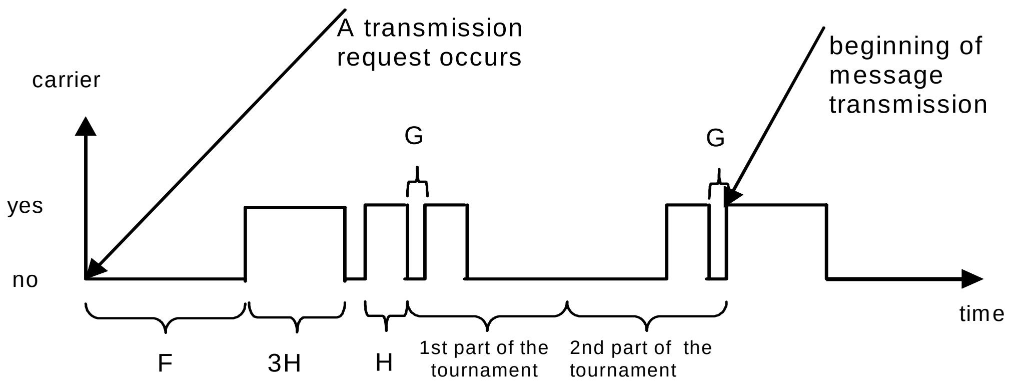

A transmission request occurs
beginning of message transmission
carrier
yes
no
G
G
time
F
3H
H
1st part of the tournament
2nd part of the tournament

Fig. 7. An illustration of the carrier wave transmitted assuming *npriobits* = 2 and the priority of the requested message is 2. This illustration is simplified; it ignores the many smaller time intervals, like *E*, *SWX* and *R*. For a full understanding of the protocol, see the automata in Figure 6.

We will now describe the two parts of the tournament. Because both parts of the tournament are similar, we only describe the first part. 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. The contention of a single bit is taken care of in state 18-32. First, a dominant bit it transmitted if it should do so and a node that hears a dominant bit, retransmits it. This happens in the states 18-29, and the only purpose of these states is that a node which had a recessive priority bit but heard a dominant bit should assign `headDOMbit:=TRUE`. After that, state 29-32 makes decisions based on the value of `heardDOMbit`.

In order to understand the time-out parameters *E*, *F*, *G*, *H* and *R* consider Figure 6. *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. *R* is intended to improve the reliability of the synchronization before the tournament.

Based on Figure 6 we compute the transmission time of a message taking the overhead of the protocol into account as:

$$C_i^{'} = C_i + 5H + 4G + (2G + 2H) \times (2npriobits) + 2L \tag{1}$$

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

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

The time-outs have to be selected carefully in order to assure the correctness of the protocol. One such choice (see Appendix B) is: E =8 µs, F = 1970µs, G = 65µs, H = 108µs, SWX = 20µs and R = 0.

Appendix B

We will now discuss the correctness of the protocol and discuss how assigning values to the constants *E*, *F*, *G*, *H*, *SWX* and *R* affects the correctness. In order to satisfy mutual exclusion and prioritization we need to assure that certain events do not occur at the wrong time. We need to assure that:

1. **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 second part of the tournament, and we consider the case when a bit is retransmitted. We require:

$$\begin{gathered}[6H+3G+(2H+2G)\times(2npriobits-1)]\times[1-\varepsilon]- \\ [5H+3G+(2H+2G)\times(2npriobits-1)]\times[1+\varepsilon] \\ -2CLK-L-2\alpha-2\times(E+SWX)>TFCS+2SWX\end{gathered} \tag{3}$$

The motivation of Equation (3) is that the inaccuracy of the synchronization after the initial period of silence is 2 × (*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.

2. **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:

$$2CLK+L+2\alpha+F\times2\varepsilon<E \tag{4}$$

3. **A node which has lost the tournament must be in receiving mode in state 53 before it receives the data bits from the transmission of the winning node (which is in state 51).** This is taken care of by the delay between state 50 to state 51. 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 satisfy the following inequality:

$$\begin{gathered}[6H+3G+(2H+2G)\times(2npriobits-1)]\times2\varepsilon+ \\ 2CLK+L+2\alpha+2\times(E+SWX)<G\end{gathered} \tag{5}$$

4. **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:

$$\begin{gathered}[6H+3G+(2H+2G)\times(2npriobits-1)]\times[1+\varepsilon]- \\ [4H+G]\times[1-\varepsilon] \\ +2CLK+L+2\alpha+2\times(E+SWX)<F\end{gathered} \tag{6}$$

5. **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 second tournament. In this iteration we consider the first bit and when it is retransmitted. We require:

$$\begin{gathered}[5H+3G+(2H+2G)\times(2npriobits-1)]\times[1-\varepsilon]- \\ [5H+2G+(2H+2G)\times(2npriobits-1)]\times[1+\varepsilon] \\ -2CLK-L-2\alpha-2\times(E+SWX)>0\end{gathered} \tag{7}$$

6. **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:

$$SWX > turnaround_{RxTx} \tag{8}$$

7. **The time that a node listens to a carrier (R) in the synchronization before it decides whether a pulse was long (*3H*) or short (*H*) must be shorter than the short pulse.** We must also take into account the uncertainty in syncronization. This requires:

$$R < H - SWX \tag{9}$$

The values of *E*, *F*, *G*, *H, SWX* and *R* must be selected to satisfy Inequalities (3)-(9). 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 = 1μs. Typical computers have *CLK* = 1μs 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* = 2μs. We choose *TFCS* = 5μs because busy tone detection of narrow-band signals have been estimated to need this time [17] and our application of carrier sensing is similar to busy tone detection. We choose $turnaround_{RxTx}$ = 19μs based on the requirement in IEEE 802.11 standard (see page 180 in [36]). We choose *npriobits* = 20 to obtain approximately one million priority levels. One choice that satisfies the constraints for this example is: *E* =8 μs, *F* = 1970μs, *G* = 65μs, *H* = 108μs, *SWX* = 20μs and *R* = 0. Hence, the overhead per message (calculated based on (2)) will be 28229μs.

We make two remarks. First, by just looking at the number 28229μs, it may seem quite high. However, our scheme should be compared with other schemes that are designed to deal with hidden nodes. We should keep in mind that RTS-CTS schemes have an overhead with the two packets and they have an additional overhead due to the collisions. These collisions have two effects (i) at least two messages needs to be retransmitted and this takes time and (ii) the time-out may make the channel idle for a quite long time. Second, the overhead our protocol is heavily dependent on $turnaround_{RxTx}$ This is an implementation parameter; there is nothing fundamental about it. We observe that the HIPERLAN [37] standard required 2μs; something that gives us an overhead of approximately 4000μs. With further improvements in Tx-Rx turn time in radio technology, the overhead of our protocol becomes less whereas other protocols are unaffected.