Technical Report

Delay-Bounded Medium Access for Unidirectional Wireless Links

Björn Andersson
Nuno Pereira
Eduardo Tovar

TR-060701
Version: 1.0
Date: July 2006

Delay-Bounded Medium Access for Unidirectional Wireless Links

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

Wireless sensor networks (WSNs) perform collaborative processing and communication of sensor readings. The sensor readings are valid only during a certain time interval and hence it is necessary that a message reaches its destination node before a pre-specified deadline. Collaborative processing mandates that the network be connected whenever radio conditions permits so. This may require that a link from sensor node A to sensor node B is used although there is no link from B to A. Such links are called unidirectional.

We study medium access in wireless sensor networks where links may be unidirectional and messages have timing requirements. Three results are presented. First, we present a medium access control (MAC) protocol which replicates a message with carefully selected pauses between replicas, and in this way it guarantees that for every message at least one replica of that message is transmitted without collision. The protocol ensures this with no knowledge of the network topology and it requires neither synchronized clocks nor carrier sensing capabilities. Second, we propose schedulability analysis techniques for the protocol. Third, we implement the protocol and show experimentally that it reduces the number of lost messages (and deadline misses) significantly as compared to schemes with pauses of random duration between replicas. We believe these results are significant because (i) this protocol is the only one that supports schedulability analysis and is designed for unidirectional links and (ii) of all MAC protocols in the literature that support schedulability analysis, our protocol is the one that makes the least assumptions.

Delay-Bounded Medium Access for Unidirectional Wireless Links

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

Abstract

Wireless sensor networks (WSNs) perform collaborative processing and communication of sensor readings. The sensor readings are valid only during a certain time interval and hence it is necessary that a message reaches its destination node before a pre-specified deadline. Collaborative processing mandates that the network be connected whenever radio conditions permits so. This may require that a link from sensor node A to sensor node B is used although there is no link from B to A. Such links are called unidirectional.

We study medium access in wireless sensor networks where links may be unidirectional and messages have timing requirements. Three results are presented. First, we present a medium access control (MAC) protocol which replicates a message with carefully selected pauses between replicas, and in this way it guarantees that for every message at least one replica of that message is transmitted without collision. The protocol ensures this with no knowledge of the network topology and it requires neither synchronized clocks nor carrier sensing capabilities. Second, we propose schedulability analysis techniques for the protocol. Third, we implement the protocol and show experimentally that it reduces the number of lost messages (and deadline misses) significantly as compared to schemes with pauses of random duration between replicas. We believe these results are significant because (i) this protocol is the only one that supports schedulability analysis and is designed for unidirectional links and (ii) of all MAC protocols in the literature that support schedulability analysis, our protocol is the one that makes the least assumptions.

1. Introduction

A significant number of wireless sensor network applications involve periodic transmission of sensor data and notification of important sporadic events, and these messages must be guaranteed to reach their destination before a pre-specified deadline. Sensor nodes self-organize and perform sensing and aggregate data collaboratively. This requires that connectivity is achieved whenever radio conditions allow so. In particular, it is important to avoid network partitioning whenever possible since this prevents

the collaboration among all nodes, such as one node notifying other nodes that an important event has occurred.

These wireless sensor networks are often intended to be deployed in a very ad-hoc fashion, such as thrown out from helicopters. The application designer knows the number of nodes deployed but does not know the topology at design time. In addition, radio irregularities, which have been found to be a common and non-negligible phenomenon in wireless networks [1-8], makes the design of the communication protocols challenging. The received signal strength from a transmitter is direction dependent and this makes links *asymmetric*; that is, the received signal strength at node B when node A broadcasts is different from the received signal strength at node A when node B broadcasts. When the asymmetry becomes large enough, a link becomes *unidirectional*; that is, if node A broadcasts a message then node B receives it, but if B broadcasts then A will not receive it. Ignoring unidirectional links can cause a route from source to destination to be longer than necessary. But more important, ignoring unidirectional links reduces connectivity; it can cause network partitioning and hence render the collaboration between sensor nodes impossible. Therefore, it is of paramount importance that the communication protocols can still be effective when unidirectional links exist.

Unidirectional links bring significant challenges to wireless communications. A network node that sends cannot receive any direct feedback from the receiver. Hence, normal implementations of acknowledgement schemes and request-to-send/clear-to-send (RTS/CTS) dialogs used in medium access do not work. Unfortunately, the medium access control (MAC) layer is still poorly developed for unidirectional links; the only existing solution [9] today for unidirectional links is based on time division multiple access (TDMA) schemes. This approach in [9] is collision-free, but it requires synchronized clocks and it does not take deadlines into account in its decisions. Static table-driven scheduling could probably be used to achieve medium access for unidirectional links (although we are not aware of any publication on it). But it has the drawback of requiring synchronized clocks and it is also well-known to

be inefficient for those sporadic message streams where the deadline is short compared to the minimum inter-arrival time of messages within a message stream.

In this paper we study medium access of wireless links which may be unidirectional and where messages have timing requirements. We show informally that under certain assumptions, designing a collision-free MAC protocol is impossible. For this reason, we design a MAC protocol that uses message replication; every message that an application requests to transmit is replicated by the MAC protocol with carefully selected pauses between the transmissions of replicas. This guarantees that for every message, *at least one of its replicas is transmitted without collision*. We analyze whether timing requirements can be met for the protocol. We also present and evaluate an implementation. The protocol depends neither on carrier sensing nor on synchronized clocks nor on topology information; it only requires that the number of sensor nodes is known.

We believe these results are significant because: (i) this protocol is the only MAC protocol designed for unidirectional links that supports schedulability analysis; and (ii) of all MAC protocols in the literature that support schedulability analysis, our protocol is the one that makes the least assumptions.

The remainder of this paper is organized as follows. Section 2 presents the system model as well as the main idea behind the protocol. Section 3 presents a schedulability analysis for the protocol whereas Section 4 evaluates it experimentally. The evaluation is performed over a real implementation on MicaZ motes [10] and performance comparison is made against other alternative MAC protocols. Additionally, in Section 4, a comparison through simulation is also performed. Section 5 discusses various practical issues of our protocol. It also reviews previous work and discusses unidirectional links in its larger context. Finally, Section 6 offers conclusions and future work.

2. Preliminaries and the Main idea

2.1. Network and Message Model

The topology is described using a graph with nodes and links. A node represents a sensor node. A link is directed. Consider a node N_i that broadcasts a message or any signal (for example an unmodulated

carrier wave). Then node N_j will receive it if and only if there is a link from node N_i to node N_j. A node can only transmit by performing a broadcast and it is impossible for a node N_i to broadcast such that only a proper subset of its neighbor nodes receives it. No assumption on the connectivity of each node is made. It is allowed that a node has only outgoing links or only ingoing links or no links at all. Unless otherwise stated, the topology is assumed to be unknown to the MAC protocol.

The traffic is characterized by the *sporadic model* [11] which can model both sporadic message requests and strictly period message requests. Each node has exactly one message stream. Node N_i is assigned the message stream τ_i. This message stream makes an infinite sequence of requests, and for each request, the message stream requests to transmit a message. The exact time of a request is unknown before run-time and the MAC protocol only knows about the time of the request when it occurs. But for every message stream τ_i there is at least T_i time units between those requests and the MAC protocol knows all T_i. For every such request, the MAC protocol must finish the transmission of $ncollisionfree(\tau_i)$ replicas of a message from stream τ_i without collisions at most D_i time units after the request. If this is the case, then we say that deadlines are met; otherwise a deadline is missed. Naturally, we assume $0 \leq D_i$ and $0 \leq T_i$. We also assume that $D_i \leq T_i$ and hence there is at most one message request at a time on a node (as long as all deadlines are met).

Let *mtotal* denote the number of nodes and let *m* denote the number of nodes that can transmit. Nodes are indexed from 1 to *mtotal*. Let $tof_{i,j}$ denote the time of flight between nodes N_i and N_j. We assume that $tof_{i,j}$ is unknown but it is bounded such that $\forall i,j \in \{1..m\}: 0 < tof_{i,j} \leq tof$. Hence, *tof* is an upper bound on the time of flight. We assume that *tof* is finite but we make no assumptions on its actual value. However, we assume the following: (i) nodes can "boot" at different times and when they boot, they do not have synchronized clocks; (ii) when a node is transmitting it cannot receive anything; and (iii) the MAC protocol can be represented as a set of timed automata, with potentially different automata on different nodes.

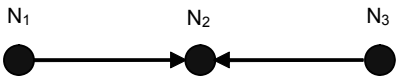


Fig. 1. A topology which illustrates the impossibility of collision-free medium access in the presence of unidirectional links. N_1 can transmit to N_2 but N_2 cannot transmit to N_1. Analogously for N_2 and N_3. When N_1 and N_3 transmit there will be a collision on node N_2.

2.2. Impossibility

Let us now show that, under these assumptions, it is impossible to design a collision-free MAC protocol when there are unidirectional links. Consider Figure 1. It illustrates a simple exemplifying topology. For such topology and links characteristics, it is necessary that N_1 does not transmit simultaneously with N_3, in order to guarantee that collisions will not occur. This requires that N_1 can get some information about the other nodes on whether there is an ongoing transmission on the other link. But N_1 cannot hear anything so the transmission from N_1 may overlap with the transmission from N_3, and then N_2 will not receive any of them. Hence, it is impossible to design a MAC protocol that is guaranteed to be collision-free in the presence of unidirectional links. Even if a node knows the topology but it does not know the time when other nodes will transmit then a collision can occur, and hence the above mentioned impossibility also extends to the case where the topology is known to the MAC protocol.

Given the impossibility of collision-free medium access in the presence of unidirectional links we will now design a solution: transmit each message many times such that at least one of the transmissions is collision-free.

2.3. The main idea

For each message request of stream τ_i, the MAC protocol transmits the message several times. Each one of them is called a *replica*. Of those replicas from message stream τ_i, let $\tau_{i,1}$ denote the one that is transmitted first. Analogously, let $\tau_{i,2}$ denote the one that is transmitted second, and so on. The number of replicas transmitted for each message of τ_i is $nreplicas(\tau_i)$, and the time between the start of the

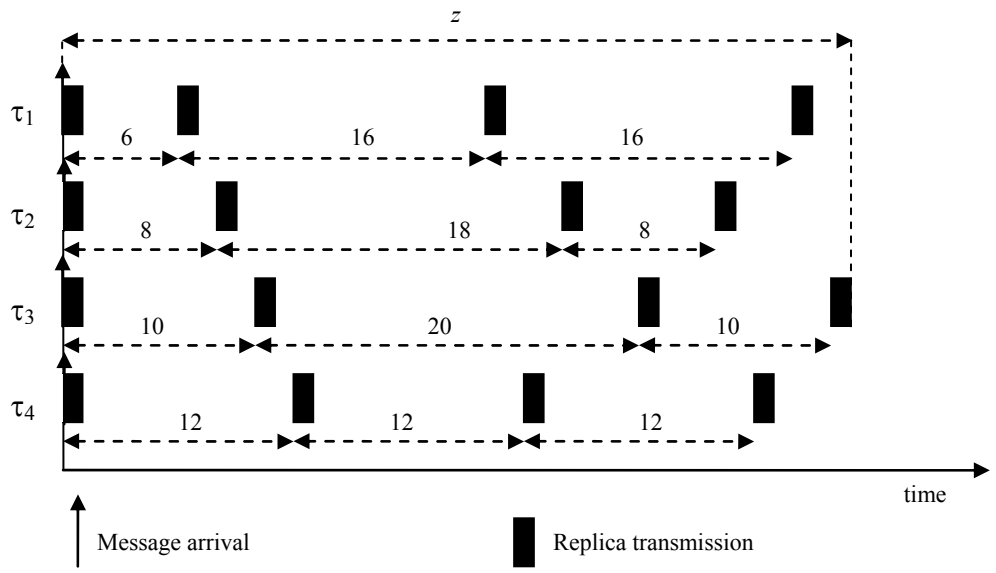


Fig. 2. Transmission of replicas with a possible assignment of Δ:s for messages τ_1, τ_2, τ_3, τ_4 requested to transmit simultaneously at time 0. As it can be seen, at least one replica is collision-free. It turns out that for every possible combination of times of requests of τ_1, τ_2, τ_3, τ_4 this is true as well.

transmission of $\tau_{i,j}$ and the time between the start of the transmission of $\tau_{i,(j+1)}$ is denoted as $\Delta_{i,j}$. To give the intuition of the replication scheme, the remainder of this section assumes that each message stream performs a single transmission request (later, in Section 3, this assumption will be removed).

Figure 2 illustrates these concepts for the case when all message streams request to transmit simultaneously. We let $\tau_{i,1}$ be transmitted immediately when τ_i is requested to be transmitted. We assume that every replica requires the same time, 1 time unit, for transmission. For convenience, we assume in this section (Section 2) that $tof = 0$ and this is known to the MAC protocol. In Section 5, we will discuss a simple technique to extend the theory for $tof > 0$. Our goal is to ensure that of those $nreplicas(\tau_i)$ replicas, the number of collision-free replicas of τ_i is at least $ncollisionfree(\tau_i)$. This should hold for all nodes that can transmit.

We will now reason about how to select $nreplicas(\tau_i)$ and then select $\Delta_{i,j}$. It is necessary to select $nreplicas(\tau_i) \geq m$ because otherwise there is a topology for which it is possible that all replicas of τ_i collide. To see this, consider m nodes where one central node N_k has ingoing links from all other nodes; one of these other nodes is node N_i. There is also a link from N_k to N_i. Let us now consider the case

where N_i broadcasts its replicas. Let N_l denote any other node than N_k and N_i. The first message transmission of τ_l can happen at any time, and so it can collide with one of the replicas from τ_i. Analogously, the first replica of another message τ_l can collide with another replica of τ_i. In addition, the first replica from τ_k can occur any time too, so this first replica can be transmitted when τ_i sends a replica to N_k. Then N_k will not hear the replica from τ_i. Hence, if τ_i transmits $nreplicas(\tau_i) < m$ replicas, it can happen that none of them are received at node N_k. Therefore, $nreplicas(\tau_i)$ must be selected such that:

$$nreplicas(\tau_i) \geq m - 1 + ncollisionfree(\tau_i).$$

Later in this section, we will select $\Delta_{i,j}$ such that at most one replica from a message of τ_i can collide with a replica from a message of τ_j. With such an assignment of $\Delta_{i,j}$, the assignment of $nreplicas(\tau_i)$ is as follows:

$$\forall i \in \{1,..,m\}: nreplicas(\tau_i) = m - 1 + ncollisionfree(\tau_i) \quad (1)$$

and this causes at least one replica from each message to be transmitted before its deadline.

Having selected $nreplicas(\tau_i) = m - 1 + ncollisionfree(\tau_i)$, the issue of selecting $\Delta_{i,j}$ will now be considered. Clearly, since a node i transmits $nreplicas(\tau_i)$ replicas, it is necessary to specify $nreplicas(\tau_i) - 1$ values of $\Delta_{i,j}$ for node i. Consider the time span starting from when an application requests to transmit on a node until the last replica has finished its transmission on that node. The maximum duration of this time span over all nodes is z (illustrated in Figure 2). An intuitive objective is to minimize z, since it corresponds to the maximum response-time of a message. This can be formulated as a mixed linear/quadratic optimization problem. Therefore, the objective is to minimize z subject to

$$\begin{aligned} \forall i \in \{1,..,m\}: \sum_{j=1}^{nreplicas(\tau_i)-1} \Delta_{i,j} + 1 \leq z \\ \forall i \in \{1,..,m\}, \forall j \in \{1,..,nreplicas(\tau_i) - 1\}: 0 \leq \Delta_{i,j} \end{aligned} \quad (2)$$

and (1), and subject to an additional third constraint that will be described now. Let u and v denote the indices of two nodes. Hence u and v belong to the set $\{1..m\}$. Let j_u and j_v denote the indices of the first replica of the sequence of replicas transmitted in nodes N_u and N_v, respectively. Hence j_u belongs to

$\{1..nreplicas(\tau_u)-1\}$ and j_v belongs to $\{1..nreplicas(\tau_v)-1\}$. Let l_u and l_v denote the lengths of these subsequences in terms of the number of replicas. l_u should be selected such that $j_u + (l_u - 1) \leq nreplicas(\tau_u) - 1$. Analogous for l_v. Hence l_u belongs to $\{1.. nreplicas(\tau_u) - j_u\}$ and l_v belongs to $\{1.. nreplicas(\tau_v) - j_v\}$. We say that a combination of u, v, j_u, j_v, l_u, l_v is valid if: (i) these 6 variables are within their ranges; and (ii) $u \neq v \wedge (j_u \neq j_v \vee l_u \neq l_v)$. For every valid combination of u, v, j_u, j_v, l_u, l_v, the optimization problem must respect the following constraint:

$$\left(\left(\sum_{j=j_u}^{j_u+l_u-1}\Delta_{u,j}\right)-\left(\sum_{j=j_v}^{j_v+l_v-1}\Delta_{v,j}\right)\right)^2 \geq 2^2 \qquad (3)$$

Intuitively, (3) states that there is no sum of consecutive Δ*:s* on node u which is equal to a consecutive sum of Δ*:s* on node v. In addition, the difference is larger than 2; this implies that it is enough to be sure that there is no collision. (To understand why the difference must be 2, consider the following system: $m = 2$, $nreplicas(\tau_1) = 2$ and $nreplicas(\tau_2) = 2$ and $\Delta_{1,1} = 2$ and $\Delta_{2,1} = 3.98$, and τ_1 arrives at time 0.99 and τ_2 arrives at time 0. Then the first replicas of τ_1 and τ_2 will collide, and the second replicas of τ_1 and τ_2 will collide as well. One can see that the sum of Δ*:s* must differ by the duration of two.).

Therefore, (3) states that at most one replica from node u can collide with a replica from node v. Hence, of those $nreplicas(\tau_u)$ replicas sent from node u, at most $m - 1$ of them can collide. Naturally, this permits stating Theorem 1 below.

Theorem 1. If the differences between transmission start times of replicas are selected according (1)-(3), then it holds that: (i) for every node i, at least $ncollisionfree(\tau_i)$ replicas do not collide; and (ii) the time from when an application requests to transmit on node i until the last replica is transmitted on node i is at most z.

Proof: Follows from the discussion above. □

We will now illustrate the use of (1)-(3) in Example 1.

Example 1. Consider $m = 4$ and $\forall i \in \{1..m\}$: $ncollisionfree(\tau_i) = 1$ to be solved using (1)-(3). The solution that is obtained is as follows:

$$\begin{array}{lll} \Delta_{1,1} = 6 & \Delta_{1,2} = 16 & \Delta_{1,3} = 16 \\ \Delta_{2,1} = 8 & \Delta_{2,2} = 18 & \Delta_{2,3} = 8 \\ \Delta_{3,1} = 10 & \Delta_{3,2} = 20 & \Delta_{3,3} = 10 \\ \Delta_{4,1} = 12 & \Delta_{4,2} = 12 & \Delta_{4,3} = 12 \end{array}$$

This is illustrated in Figure 2. □

It is easily perceived that the number of inequalities in (3) grows as $O(m^6)$. Hence, it is only possible to solve small problems with this approach. (There were 232 constraints for $m = 4$ and 3411 constraints for $m = 6$. We used a modeling tool (AMPL [12]) and a back-end solver (LOQO [13]), and with these tools it was only possible to solve (1)-83) for $m \leq 6$.) Many interesting systems are larger though. For those systems the optimization problem phrased in (1)-(3) simply cannot be solved because the number of inequalities in (3) is too large. For this reason, later on in this paper, another technique for selecting Δ*:s* will be proposed and discussed.

3. Schedulability analysis

Since wireless sensor networks interact closely with their surrounding physical environment, it is often necessary to guarantee that a collision-free message reaches its destination before a pre-specified deadline. This requires that the waiting time of a message of a single hop can be bounded and analyzed at design time. Such an analysis was performed in [14], but for an abstract MAC protocol. This section discusses how the technique with message replication described earlier can be used to guarantee that sporadic message streams meet their deadlines.

From Section 2 it results that the maximum time it takes from when a message requests to send until the MAC protocol has transmitted a collision-free replica is z, if a message stream only makes a single request. Based on this, it would be tempting to think that if $\forall i \in \{1..m\}$: $z \leq D_i$ then all deadlines are met.

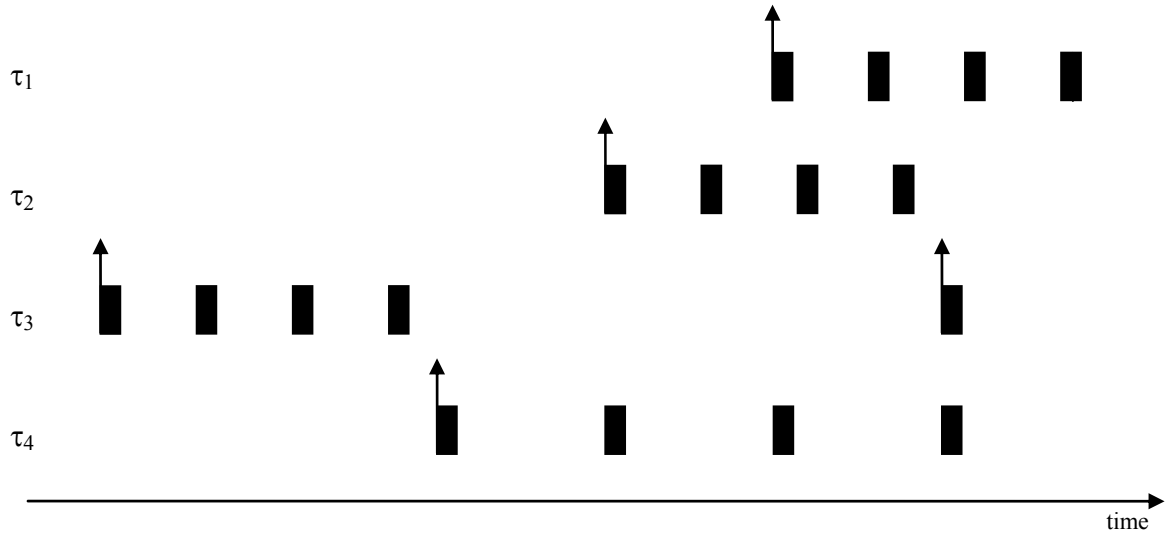


Fig. 3. Consider Δ:*s* that are selected based on the assumption a transmission request on a node occurs at most once. If these Δ:*s* are used for sporadic message streams with $T_1 = T_2 = T_3 = T_4 = z$ then a deadline miss can occur. All replicas from τ_4 collide and τ_4 misses its deadline.

Unfortunately, this is false, as illustrated by Figure 3, even if $T_1 = D_1 = T_2 = D_2 = \ldots = T_m = D_m$. A correct schedulability analysis is that if $\forall i \in \{1..m\}: z + \max(T_j: \tau_j \neq \tau_i) \leq D_i$ then all deadlines are met. It is pessimistic however for cases where message streams have very different T_i. For this reason, a better schedulability analysis technique is developed. It turns out that the development of such a schedulability analysis is simplified by adding (carefully selected) constraints to the optimization problem (described in Section 2) that finds Δ:*s*. Thus, Section 3.1 presents an alternative algorithm to assign Δ:*s*. While that algorithm implies a slightly higher *z*, it makes the analysis possible. Based on these Δ:*s*, the schedulability analysis formulation is then presented in Section 3.2. An algorithmic approach to assign the number of replicas to nodes is presented in Section 3.3, and the Δ:*s* are computed. Finally, Section 3.4 discusses how the protocol and the analysis can be extended for the case where there are many message streams assigned to a node.

3.1. Alternative algorithms to assign Δ:s

In the optimization problem phrased by (1)-(3) only constraints that were necessary to ensure that at least one replica from a message is collision-free where added. But, by enforcing a certain structure on

$\Delta_{i,j}$, it is possible to reduce the number of sums that (3) can generate, and this also makes finding out if a sum in (3) causes a collision easier. Selecting

$$\Delta_{i,1} = \Delta_{i,2} = \Delta_{i,3} = ... = \Delta_{i,(nreplicas(\tau_i)-1)} \tag{4}$$

is an example of such a structure. It is advantageous because the number of sums of subsequences that can be created does not grow very rapidly with $nreplicas(\tau_i)$. For this reason, the remainder of this section assumes that (4) must be satisfied, and for convenience we let Δ_i denote $\Delta_{i,j}$.

By rewriting (2) it results that z must satisfy the following condition:

$$\begin{gathered} \forall i \in \{1,..,m\}: \Delta_i \times (nreplicas(\tau_i) - 1) + 1 \le z \\ \forall i \in \{1,..,m\}: 0 \le \Delta_i \end{gathered} \tag{5}$$

If the sums of two subsequences differ by one then a collision can occur. Therefore it is required that:

$$\forall i \in \{1,..,m\}: \Delta_i \text{ is even and } \Delta_i \ge 2 \tag{6}$$

Let us now consider an arbitrary message from τ_u and another arbitrary message from τ_v. The number of collisions between these messages is thus given by:

$$coll_{u,v} = \left\lfloor \frac{L_{u,v}}{lcm(\Delta_u, \Delta_v)} \right\rfloor + 1 \tag{7}$$

where $L_{u,v} = \min(\Delta_u \times (\, nreplicas(\tau_u) - 1\,), \Delta_v \times (\, nreplicas(\tau_v) - 1)\,)$ assuming that $u \neq v$. For $u = v$ then it results that $coll_{u,v} = 0$. Rewriting (1) yields:

$$nreplicas(\tau_i) = \left(\sum_{v=1, v \neq i}^{m} coll_{i,v} \right) + ncollisionfree(\tau_i) \tag{8}$$

One can satisfy (5)-(8) with different choices of $nreplicas(\tau_i)$ and Δ:*s*. In order to simplify the problem, the least number of replicas is privileged, and hence the following constraint must be added:

$$\forall i \in \{1,..,m\}, v \in \{1,..,m\}, i \neq v: coll_{i,v} = 1 \tag{9}$$

The objective is now to minimize z subject to (5)-(9). Then, applying (4) provides values for the Δ:*s*. The following example (Example 2) illustrates this approach.

Example 2. Consider $m = 4$ and $\forall i \in \{1..m\}$: $ncollisionfree(\tau_i) = 1$ to be solved using (5)-(9). Applying (8) and (9) gives that $nreplicas(\tau_i) = 4$. Then solving (5)-(9) permits obtaining the following values:

$$\begin{aligned} \Delta_1 &= 2 \quad nreplicas(\tau_1) = 4 \\ \Delta_2 &= 8 \quad nreplicas(\tau_2) = 4 \\ \Delta_3 &= 10 \quad nreplicas(\tau_3) = 4 \\ \Delta_4 &= 14 \quad nreplicas(\tau_4) = 4 \end{aligned}$$

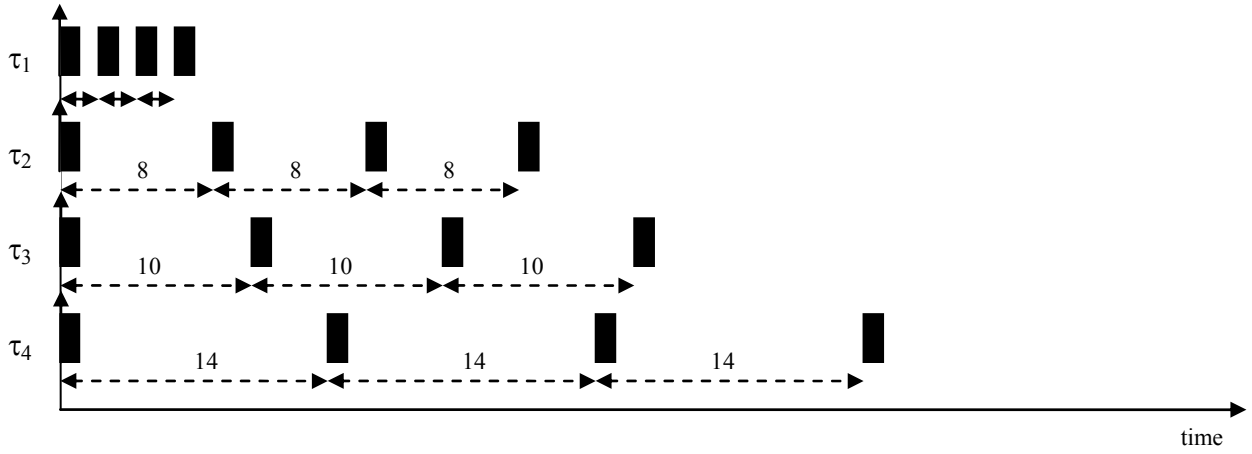
The same Δ will be used for all replicas of a message by applying (4). Therefore, the following solution will be obtained.

$$\begin{array}{lll} \Delta_{1,1} = 2 & \Delta_{1,2} = 2 & \Delta_{1,3} = 2 \\ \Delta_{2,1} = 8 & \Delta_{2,2} = 8 & \Delta_{2,3} = 8 \\ \Delta_{3,1} = 10 & \Delta_{3,2} = 10 & \Delta_{3,3} = 10 \\ \Delta_{4,1} = 14 & \Delta_{4,2} = 14 & \Delta_{4,3} = 14 \end{array}$$

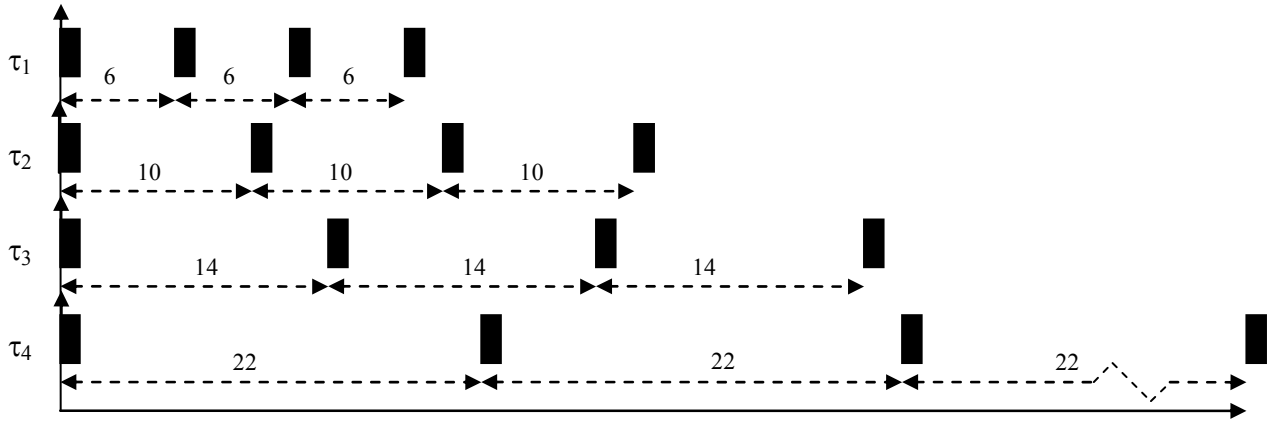
This is illustrated in Figure 4a. This solution can be compared with the solution in Example 1. One aspect to note is that z becomes 7% larger as compared to its value in Example 1. □

With the additional constraints in this subsection, we are no longer constrained by $m \leq 6$; we can assign Δ:s to nodes even when $m > 6$ by simply enumerating solutions. However, the denominator in (7) is highly nonlinear and non-differentiable, and so its is advisable to add more (carefully selected) constraints on the Δ:s in order to find solutions when m is really large.

Observe (from (5) and the numerator of (7)) that one should try to assign small numbers to Δ:s. But on the other hand, it can be seen (from the denominator of (7)) that one should assign Δ:s such that for a pair Δ_u, Δ_v it holds that $lcm(\Delta_u, \Delta_v)$ is large. Clearly these two requirements are contradictory. However, if every pair Δ_u, Δ_v is assigned $\Delta_u = 2 \times A_u$ and $\Delta_v = 2 \times A_v$ where A_u and A_v are small integers and relative prime, then $lcm(\Delta_u, \Delta_v)$ may still be fairly large. Solving this when m is large is still computationally expensive. For this reason, we will assume that A_u and A_v are prime numbers and there is no pair A_u and A_v where $A_u = A_v$.



(a) Δ:*s* for all replicas of one message are the same.



(b) Δ:*s* for all replicas of one message are the same and they are prime numbers.

Fig. 4. An example where $m = 4$ and $\forall i$: $nreplicas(\tau_v) = 4$. Figure 4a and 4b show different techniques of selecting Δ:*s*.

Let $primes(j)$ be defined as the j^{th} prime number. As an illustration of this definition, consider: $primes(1) = 2$; $primes(2) = 3$; $primes(3) = 5$; $primes(4) = 7$; $primes(5) = 11$ and $primes(6) = 13$. Then, Δ:*s* can be assigned as follows:

$$\Delta_i = 2 \times primes(k + i - 1) \tag{10}$$

where k is a design parameter which controls the magnitude of the least prime numbers being used. Observe that the "2" in (10) assures that (6) is true. Given this restriction, now there is only one free variable k. Applying (10) results in the following optimization problem: minimize z subject to (10) and

$$1 \le k \tag{11}$$

and (5), (7), (8) and (9). This can be solved by simply testing $k = 1, 2, \ldots$, until all constraints are satisfied. Example 3 illustrates this.

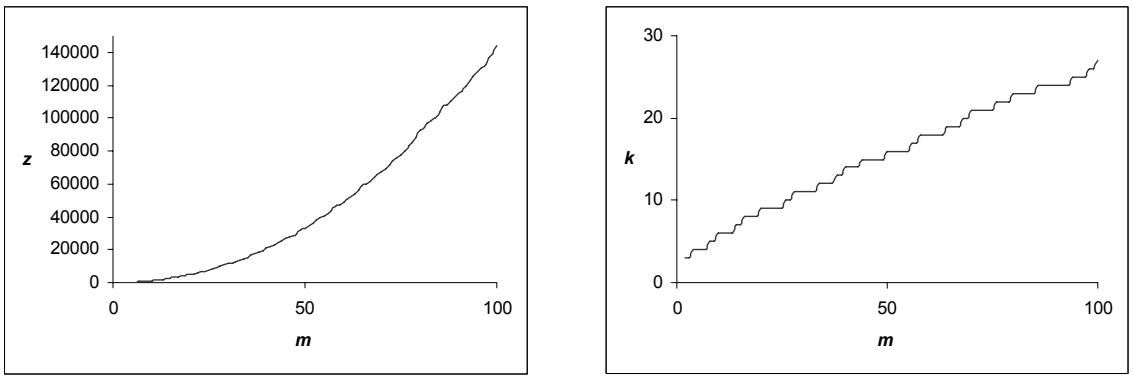
Example 3. Consider $m = 4$ and $\forall i \in \{1..m\}$: $ncollisionfree(\tau_i) = 1$ to be solved using (10)-(11) and (5), (7), (8) and (9). From (8) and (9) it results that $\forall i \in \{1..m\}$: $nreplicas(\tau_i) = 4$. Trying $k = 1$ results that: $\Delta_1 = 4$, $\Delta_2 = 6$, $\Delta_3 = 10$, $\Delta_4 = 14$. Unfortunately, with this choice, $\Delta_1 = 4$, $\Delta_2 = 6$ inserted in (7) gives $coll_{1,2} = 2$, which violates (9). Hence, one can try $k = 2$. This results that $\Delta_1 = 6$, $\Delta_2 = 10$, $\Delta_3 = 14$, $\Delta_4 = 22$. This satisfies all constraints, and so the solution is:

$$\begin{array}{ll} \Delta_1 = 6 & nreplicas(\tau_1) = 4 \\ \Delta_2 = 10 & nreplicas(\tau_2) = 4 \\ \Delta_3 = 14 & nreplicas(\tau_3) = 4 \\ \Delta_4 = 22 & nreplicas(\tau_4) = 4 \end{array}$$

This is illustrated in Figure 4b. By comparing this solution with the solution in Example 1 it is possible to observe that z becomes 63% larger. □

With this technique, we have obtained Δ:s for 2048 nodes. Figure 5 illustrates the magnitude of the parameters that were obtained for m ranging from 2 up to 100. The following conclusions can be drawn. First, clearly a higher k is needed when the number of nodes is high. This is natural because with more nodes there are more possibilities for collisions between replicas. This increase in k causes Δ:s to increase as the number of nodes increase. It can also be seen that z (the upper bound on the time span starting when an application requests to transmit a message until at least one replica is transmitted without collision) increases as m increases. There are two reasons for that: (i) $nreplicas(\tau_i)$ increase and (ii) Δ:s values increase.

Overall it can be seen that the overhead is quite high. Unfortunately, the current state-of-the-art offers no better solution. In applications with high reliability, the ultimate goal is to ensure that receivers receive the messages that they should receive. There are two possible threats: (i) collisions and (ii) noise



a) **b)**

Fig. 5. Illustration on how the parameters z and k vary as the number of computer nodes increases. The technique with equal pauses between replicas of the same message and prime numbers is used.

that causes messages to be lost or corrupts bits leading to CRC errors. Although our protocol deals with the first threat, it is desirable to deal with the latter as well. Forward error correcting codes are helpful for the data bits but messages can still be lost due to frame synchronization errors. Regardless of the cause of lost messages or corrupt messages, our scheme can be used to combat them; we can require that $ncollisionfree(\tau_i) \geq 2$. Example 4 will study the impact of this in the overhead of our protocol.

Example 4. Consider $m = 4$ and $\forall i \in \{1..m\}$: $ncollisionfree(\tau_i) = 2$ to be solved using (5), (7), (8) and (9). Using (8) and (9) it results that $nreplicas(\tau_i) = 5$. The same solution as the one obtained in Example 2 results, but now with $z = 89$. By increasing $ncollisionfree(\tau_i)$, the following parameters are obtained (Table 1).

Table 1. Δ:s and z for Example 4

$ncollisionfree(\tau_i)$	Δ_1	Δ_2	Δ_3	Δ_4	z
3	10	14	22	26	131
4	10	14	22	26	157
5	14	22	26	34	239

Observe that z increases slowly with $ncollisionfree(\tau_i)$. □

Example 4 shows that although our MAC protocol has a fairly high overhead for $ncollisionfree(\tau_i) = 1$, the overhead increases (relatively) slowly as $ncollisionfree(\tau_i)$ grows.

3.2. Schedulability analysis formulation

In order to find out whether all deadlines are met, now we analyze how one message from a message stream τ_i is affected by the other message streams. *nreplicas*(τ_i) must be sufficiently large; we choose the smallest number which satisfies the constraint on the number of collision-free messages. Hence, *nreplicas*(τ_i) should be selected as follows:

$$nreplicas\,(\tau_i) \geq \#collisions\ during\big(i,[r_i, r_i + D_i)\big) + ncollisionfree\,(\tau_i) \tag{12}$$

#collisions during $[r_i,r_i + D_i)$ denotes a upper bound on the number of collisions that a message from message stream τ_i can suffer from during the time interval of length D_i. Let r_i denote an arbitrary arrival time of a message from message stream τ_i. The *#collisions during* $[r_i,r_i+D_i)$ can be computed as follows:

$$\#collisions\ during\ \big(i,[r_i, r_i + D_i)\big) = \sum_{v=1, v\neq i}^{m} \#collisions_{i,v}[r_i, r_i + D_i) \tag{13}$$

$\#collisions_{i,v}[r_i,r_i + D_i)$ denotes an upper bound on the number of collisions from message stream v on a message from message stream i during a time interval of length D_i.

For computing $\#collisions_{i,v}[r_i,r_i + D_i)$, the interval $[r_i,r_i + D_i)$ can be divided into subintervals. Let t_0 denote the first time that τ_v requests to transmit after r_i. This time interval $[r_i, t_0)$ is called the *head*. Let t_1 denote the largest number such that $(t_1 - t_0) / T_v$ is an integer and $t_1 \leq r_i + D_i$. The time interval $[t_0, t_1)$ is called the *body*. Finally, the interval $[t_1, r_i + D_i)$ is called the *tail*.

Each of these intervals (*head, body* and *tail*) can contain a certain amount of collisions. Let x denote the length of the head; that is, $x = t_0 - r_i$. Based on this, an upper bound on $\#collisions_{i,v}[r_i,r_i + D_i)$ can be computed as follows:

$$\max_{0\leq x\leq D_i}\left(collborder_{i,v}(x) + \left\lfloor \frac{D_i - x}{T_v} \right\rfloor \times coll_{i,v} + collborder_{i,v}\left(D_i - x - \left\lfloor \frac{D_i - x}{T_v} \right\rfloor \times T_v \right) \right) \tag{14}$$

Equation (14) is still complex and will be simplified later on in this paper. Observe that this expression (14) takes the maximum of all x in $[0, D_i]$ of an expression with three terms. The first term corresponds to the number of collisions in the head, the second term corresponds to the number of

collisions in the body and the third term corresponds to the collisions in the tail. The term $coll_{u,v}$ represents an upper bound on the number of collisions every time τ_u and τ_v request to transmit. The term $collborder_{u,v}(x)$ is similar to $coll_{u,v}$ but the number of collisions is only counted during a time interval of length x. Computing $coll_{u,v}$ is similar (7), but now D_u can be used to bound the time interval over which collisions can occur. Hence, by adapting (7) to the context of sporadic message streams, $coll_{u,v}$ is given by:

$$coll_{u,v} = \left\lfloor \frac{\min\left(L_{u,v}, D_u\right)}{lcm\left(\Delta_u, \Delta_v\right)} \right\rfloor + 1 \tag{15}$$

where $L_{u,v} = \min(\Delta_u \times (\ nreplicas(\tau_u) - 1\ ), \Delta_v \times (\ nreplicas(\tau_v) - 1)\ )$. Observe that $coll_{u,v} \neq coll_{v,u}$.

Calculating $collborder_{u,v}(x)$ is very similar but now there is an additional limitation on the length of the time interval during which collisions can occur. Adapting (15) gives:

$$collborder_{u,v}(x) = \left\lfloor \frac{\min\left(L_{u,v}, D_u, x\right)}{lcm\left(\Delta_u, \Delta_v\right)} \right\rfloor + 1 \tag{16}$$

where $L_{u,v} = \min(\Delta_u \times (\ nreplicas(\tau_u) - 1\ ), \Delta_v \times (\ nreplicas(\tau_v) - 1)\ )$.

Once again, observe that $collborder_{u,v}(x) \neq collborder_{v,u}(x)$.

With (15) and (16), it is possible to simplify (14). If $x \geq T_v$ then T_v can be subtracted from x until $x < T_v$ and this maintains or increases the number of collisions. Since $0 \leq x < T_v$, one can find upper bounds for the first term in (14) and the sum of the other terms in (14). $\#collisions_{i,v}[r_i, r_i + D_i)$ can then be computed as follows:

$$\#collisions_{i,v}\left[r_i, r_i + D_i\right) = collborder_{i,v}\left(T_v\right) + \left\lfloor \frac{D_i}{T_v} \right\rfloor \times coll_{i,v} + collborder_{i,v}\left(D_i - \left\lfloor \frac{D_i}{T_v} \right\rfloor \times T_v\right) \tag{17}$$

Transferring (5) to the sporadic model implies that the following must be satisfied:

$$\begin{gathered} \forall i \in \{1,..,m\} : \Delta_i \times \left(nreplicas(\tau_i) - 1\right) + 1 \leq D_i \\ \forall i \in \{1,..,m\} : 0 \leq \Delta_i \end{gathered} \tag{18}$$

This finalizes the reasoning on the schedulability analysis.

3.3. Assigning number of replicas and sequences of Δ:s to message streams

We will now design the MAC protocol to satisfy the constraints (12)-(13) and (15-18). It can be seen that (14)-(16) are highly non-linear and non-differentiable and so a heuristic will be designed.

First, an algorithm that selects $nreplicas(\tau_i)$ when Δ:*s* are known and fixed is developed. Then an algorithm that finds the correct Δ:*s* and uses the previous algorithm as a subroutine will be designed.

§Δ:*s* are fixed. Consider the case when Δ:*s* are fixed and the goal is to find $nreplicas(\tau_i)$. The technique employed to solve this is based on fixed-point iteration. Before doing so, some lemmas need to be established. The proof of these lemmas is based on direct inspection of (12), (13) and (15-18).

Let $nreplicas(\tau_1)^*$, $nreplicas(\tau_2)^*$,...., $nreplicas(\tau_m)^*$ denote a solution to (12), (13) and (15-18). Let us consider a vector $nreplicas(\tau_1)$, $nreplicas(\tau_2)$,...., $nreplicas(\tau_m)$ such that $\forall i \in \{1..m\}: 0 \le nreplicas(\tau_i) \le nreplicas(\tau_i)^*$.

Lemma 1. $nreplicas(\tau_1)$, $nreplicas(\tau_2)$,...., $nreplicas(\tau_m)$ is a solution to (13) and (15-18).

Observe that Lemma 1 does not say anything about whether (12) is satisfied or not.

Lemma 2. Consider the case $\forall i \in \{1,..,m\}: 0 \le nreplicas(\tau_i) \le nreplicas(\tau_i)^*$. If one makes the assignment $nreplicas(\tau_i) \leftarrow$ right hand side (RHS) of (12) for any *i* then the resulting vector of $nreplicas(\tau_i)$ still satisfies

$$\forall i \in \{1,..,m\}: nreplicas(\tau_i) \le nreplicas(\tau_i)^*.$$

Lemma 2 can be generalized to the case where a subset is updated. Lemma 3 does that.

Lemma 3. Consider the case $\forall i \in \{1,..,m\}: 0 \le nreplicas(\tau_i) \le nreplicas(\tau_i)^*$. If one makes the assignment $nreplicas(\tau_i) \leftarrow$ RHS of (12) for any subset of *i*:*s*, then the resulting vector of $nreplicas(\tau_i)$ still satisfies

$$\forall i \in \{1..m\}: nreplicas(\tau_i) \le nreplicas(\tau_i)^*.$$

Lemma 1 gives a suggestion on where to find an initial solution, and Lemma 3 gives a suggestion on how to iterate towards a solution. Based on this, Algorithm 1 (in Figure 6) is proposed. From Lemmas 1-3, the following theorem (Theorem 2) can be stated.

Algorithm 1: Assigning *nreplicas*(τ_i) to nodes.

```
1.  assign ∀i∈{1..m}: nreplicas(τ_i)←2
2.  while (18) is satisfied for all τ_i do
3.          calculate the RHS from (13) and (15)-(17) and then calculate the RHS of (12)
4.          if ∀i∈{1..m}: (12) is satisfied then
5.                  declare SUCCESS
6.          else
7.                  for ∀i∈{1..m}: nreplicas(τ_i) such that (12) is not satisfied then for those i, do:
8.                          nreplicas(τ_i) ← RHS of (12) for that i
9.                  end for
10.         end if
11. end while
```

Fig. 6. An algorithm for assigning *nreplicas*(τ_i) to nodes when the Δ:*s* are already assigned. This algorithm is designed for the case when a node has a single sporadic message stream.

Theorem 2. Consider the case where Δ:*s* are fixed. If there is a solution to (12), (13) and (15-18) then Algorithm 1 will declare success. If there is no solution to (12), (13) and (15-18) then Algorithm 1 will declare failure.

Proof: Follows from Lemmas 1-3.

Although Algorithm 1 is useful, it requires the assignment of values to the Δ:*s* first. That will be addressed below.

§Δ:*s* are not assigned yet. Consider now the case when Δ:*s* are not yet assigned values. Observe (from (18)) that message streams with small Δ:*s* tend to be able to satisfy short D_i.

The algorithm is called Algorithm 2 and it is described in Figure 7. It assumes message streams are sorted already and the idea is similar to the one used in Algorithm 1.

In general, however, message streams are not sorted according to their deadlines. For this reason, Algorithm 3, described in Figure 7 sorts messages streams and applies then Algorithm 2. The use the Algorithm 3 is shown through Example 5.

Example 5. Consider $m = 4$ and $\forall i \in \{1..m\}$: *ncollisionfree*(τ_i) = 1 and the set of message streams as shown in Table 2. It is noteworthy to see that this set of message streams cannot be guaranteed to meet deadlines with optimal Δ:*s* and the schedulability test "if $\forall i \in \{1..m\}: z + \max(T_j: \tau_j \neq \tau_i) \leq D_i$ then

Algorithm 2: Assigning Δ:s to nodes, assuming $D_1 \le D_2 \le \ldots \le D_m$.

1. $k \leftarrow 0$
2. repeat
3. $k \leftarrow k + 1$
4. assign Δ:*s* according to (10)
5. if there is an *i* that violates (18) when $nreplicas(\tau_i)=2$ is inserted in (18) then
6. declare FAILURE
7. end if
8 assign $nreplicas(\tau_i)$ according to Algorithm 1.
9. until line 8 declared FAILURE
10. declare SUCCESS

Algorithm 3: Assigning Δ:s to nodes.

1. Sort message streams such that $D_1 \le D_2 \le \ldots \le D_m$.
2. Run Algorithm 2.
3. Assign to node N_i: the $\Delta_{i,k}$ $\forall k \in \{1..\ nreplicas(\tau_1)\}$

Fig. 7. An algorithm for assigning Δ:s to nodes.

all deadlines are met". Algorithm 3 will now be applied on this example.

Table 2. Message streams used in Example 5.

message streams	T_i	D_i
τ_1	35	35
τ_2	92	92
τ_3	184	184
τ_4	550	550

First, Algorithm 3 sorts message streams according to their deadlines. It can be seen in Table 2 that they are already sorted. Then Algorithm 2 is called and *k* is initialized to 0. *k* is incremented so k = 1 and line 4 in Algorithm 2 provides $\Delta_1 = 4$, $\Delta_2 = 6$, $\Delta_3 = 10$, $\Delta_4 = 14$. Inserting $nreplicas(\tau_i) = 2$ in (18) and inserting the Δ:*s* gives the following tests:

$$4 \times (2-1) + 1 \le 35$$
$$6 \times (2-1) + 1 \le 92$$
$$10 \times (2-1) + 1 \le 184$$
$$14 \times (2-1) + 1 \le 550$$

It can be seen that all of these inequalities are true, and Algorithm 2 proceeds to line 8 and it executes Algorithm 1. Line 1 in Algorithm 1 assigns $nreplicas(\tau_1) = 2$, $nreplicas(\tau_2) = 2$, $nreplicas(\tau_3) = 2$,

nreplicas(τ_4) = 2. The inequalities of (18) are tested again and they are true. The execution of line 3 in Algorithm 3 gives:

$$\begin{array}{ll} \#collisions_{1,2}\left[r_1, r_1 + D_1\right) = 2 & \#collisions_{3,1}\left[r_3, r_3 + D_3\right) = 7 \\ \#collisions_{1,3}\left[r_1, r_1 + D_1\right) = 2 & \#collisions_{3,2}\left[r_3, r_3 + D_3\right) = 3 \\ \#collisions_{1,4}\left[r_1, r_1 + D_1\right) = 2 & \#collisions_{3,4}\left[r_3, r_3 + D_3\right) = 2 \\ \#collisions_{2,1}\left[r_2, r_2 + D_2\right) = 4 & \#collisions_{4,1}\left[r_4, r_4 + D_4\right) = 17 \\ \#collisions_{2,3}\left[r_2, r_2 + D_2\right) = 2 & \#collisions_{4,2}\left[r_4, r_4 + D_4\right) = 7 \\ \#collisions_{2,4}\left[r_2, r_2 + D_2\right) = 2 & \#collisions_{4,3}\left[r_4, r_{41} + D_4\right) = 4 \end{array}$$

Combining these (using (13)) yields:

$$\begin{array}{ll} \#collisions\ during\ \left(1, \left[r_1, r_1 + D_1\right)\right) = 6 & \#collisions\ during\ \left(3, \left[r_3, r_3 + D_3\right)\right) = 12 \\ \#collisions\ during\ \left(2, \left[r_2, r_2 + D_2\right)\right) = 8 & \#collisions\ during\ \left(4, \left[r_4, r_4 + D_4\right)\right) = 28 \end{array}$$

Applying this in (12) yields that:

$$\begin{array}{ll} 2 \geq 6 + 1 & 1 \geq 12 + 1 \\ 2 \geq 8 + 1 & 1 \geq 28 + 1 \end{array}$$

must be tested, and it can be seen that all of them are false. Hence, Algorithm 1 proceeds to line 7 and line 8, where it assigns: *nreplicas*(τ_1) = 7; *nreplicas*(τ_2) = 9; *nreplicas*(τ_3) = 13 and *nreplicas*(τ_4) = 29. Algorithm 1 and Algorithm 2 proceed in the same way. Finally, it results in: *nreplicas*(τ_1) = 9; *nreplicas*(τ_2) = 14; *nreplicas*(τ_3) = 18 and *nreplicas*(τ_4) = 33. This satisfies the deadlines. □

3.4. Many messages streams per node

Now, considering the case where each computer node has many message streams assigned to it, and that every message stream can request to transmit a message.

This case can be simply dealt with by treating each message stream as if it was on its own node and assign *nreplicas*(τ_i) and Δ_i, accordingly. If they are on their own nodes they will meet all deadlines. Message streams should be assigned to the nodes where they should be and the number of collisions will not be higher. Example 6 illustrates this.

Example 6. Consider 4 message streams with timing parameters as given by Table 2. Message stream τ_1 should be assigned to N_1 and τ_2 should be assigned to N_2. But node N_3 is special; it is assigned two messages streams τ_3 and τ_4. The approach is considering 4 network nodes and 1 message stream on each one of them. The following values can then be taken from the results in Example 5: $nreplicas(\tau_1) = 9$; $nreplicas(\tau_2) = 14$; $nreplicas(\tau_3) = 18$ and $nreplicas(\tau_4) = 33$; and $\Delta_1 = 4$; $\Delta_2 = 6$; $\Delta_3 = 10$; $\Delta_4 = 14$. These values can be used for the 4 message streams that are assigned to the 3 nodes. □

4. Implementation and Experiments

Having seen that the replication scheme can guarantee that $ncollisionfree(\tau_i)$ replicas are collision-free in theory, we now turn to practice. We want to address the following hypotheses:

1. The replication scheme is easy to implement.
2. The number of lost or corrupted messages at the receiver is smaller when the replication scheme in this paper is used, as compared to a replication scheme with random pauses. This applies even if the random scheme transmits only a single replica per message.
3. The replication scheme guarantees that $ncollisionfree(\tau_i)$ replicas are indeed collision-free.
4. If a link is bidirectional then our replication scheme can be extended so that it still offers a bounded number of collisions but it also has a low average-case overhead.

In order to test these hypotheses, we implement the replication protocol both on a real platform and use simulation[1]. The following sections describe the implementation, experimental setup and results obtained. But first we turn our attention to how the Δ:*s* for the experimental setup were determined.

4.1. Finding Near Optimal Δ:s

To test the hypotheses stated above, the network should be as loaded as possible (high utilization). The way to achieve this would be to employ optimal Δ:*s* (the approach described in Section 2.3) because

[1] Both implementations and all parameters used for the simulation runs can be downloaded from http://www.hurray.isep.ipp.pt/hydra/

they minimize z, allowing T_i to be low and this causes the utilization to be high. However finding optimal Δ:s for $m > 6$ is currently not possible, and therefore another technique to obtain Δ:s was developed. This technique gives a near-optimal solution to (1)-(3).

The algorithm for doing that (Algorithm 4) is presented in Figure 8. The algorithm does not find Δ:s directly; it finds sequences of integers and the goal is to find m sequences. One sequence is assigned to each network node and each sequence contains $(m - 1)$ positive even integers. Consider a node i, which is assigned the sequence S_i. Then $\Delta_{i,k}$ is assigned the k^{th} number in sequence S_i.

The algorithm works as follows. First, a "guess" on an upper bound on z is put forward and this guess is denoted *maxz* (line 1). Then, sequences of numbers are generated (on lines 2-13). It is known that if the sum of the numbers in the sequence exceeds *maxz* then such a sequence should not be used because it cannot produce the set of sequences with the minimum z (assuming that $z \leq maxz$). Hence, such a sequence should not be considered further. Not all possible sequences are enumerated. This would be too time-consuming. Instead those sequences that are likely to be useful are enumerated. It was previously seen, in Section 3, that if Δ:s in a sequence are the same, then the number of unique sums that can be generated is small. For this reason such sequences are considered (the variable `sequences` contains all sequences that will be considered). However, it is desirable to obtain a z that is smaller than the one obtained in Section 3.1. For this reason, different values in a sequence are allowed. However, to ensure that the number of unique sums that can be generated do not increase too much, special care is taken: (i) all numbers in a sequence should have a large common denominator and (ii) the sequence should be symmetric in that the sequence should be the same if the order of the elements are reversed. After generating those sequences that appear promising, a selection of a subset of them (lines 23-26) is performed. Sequences S and S' are said to *collide* if there is a subsequence of S and a subsequence of S' such that the sums of the elements in the subsequences are equal. When sequences are selected, one must ensure that for every pair S, S' of selected sequences it holds that S and S' do not collide. This is

Algorithm 4: The algorithm for finding near-optimal Δ:*s*

Input: *m*
Output: sequences of Δ:*s*
const NDELTAS = *m*-1
type sequence = record
the_numbers : array[1..NDELTAS] of integer
the_sums : set of integer
n_conflict_with : integer
endrecord
asequence : sequence
sequences : set of sequence initialized to ∅
selected_sequences : set of sequence initialized to ∅
1. Guess an upper bound on z. It is denoted $maxz$
2. for *factor* ← 2 to ⌈$maxz$ /NDELTAS⌉ step 2 loop
3. *asequence.the_numbers* ← < *factor*, *factor*, ..., *factor* >
4. *sequences* ← *sequences* ∪ {*asequence*}
5. for *indent* ← 1 to ⌊ NDELTAS/2 ⌋ loop
6. for *multiplier* ← 2 to ⌊$maxz$ /factor⌋ loop
7. for *k* ← 1+*indent* to NDELTAS - *indent* loop
8. *asequence.the_numbers*[*k*] ← *asequence.the_numbers*[*k*] × *multiplier*
9. if sum of all numbers in *asequence* ≤ $maxz$ then
10. *sequences* ← *sequences* ∪ {*asequence*}
11. endfor
12. endfor
13. endfor
14. for each sequence *S* in *sequences* do
15. *S.the_sums* ← ∅
16. for each subsequence *S´* that can be created from *S* such that consecutive elements in *S´* are also consecutive in *S*
17. *S.the_sums* ← *S.the_sums* ∪ {sum of all elements in *S´*}.
18. end for
19. endfor
20. for each sequence *S* in *sequences* do
21. *S.n_conflict_with* ← | { *S´* : (*S´* ∈ *sequences*)∧((*S.the_sums* ∩ *S´.the_sums*) ≠ ∅) } |
22. endfor
23. for each sequence *S* in *sequences* in ascending order of *n_conflict_with* do
24. if there is no sequence *S´* in *selected_sequences* such that (*S.the_sums* ∩ *S´.the_sums* ≠ ∅) then
25. *selected_sequences* ← *selected_sequences* ∪ {*S*}
26. endfor
27. return *selected_sequences*

Fig. 8. An algorithm for finding and assigning Δ:s.

checked on line 24. When sequences are considered for selection, they are considered in the reverse order of how many collide with them.

4.2. Implementation and Experimental Setup

The replication protocol was implemented on the MicaZ platform [10], and this implementation was dubbed HYDRA. MicaZ is a sensor network platform offering a low power microcontroller, 128 Kbytes

of program flash memory and an IEEE 802.15.4 compliant radio transceiver, capable of 250 kbps data rate. The MicaZ supports running TinyOS [15], an open-source operating system designed for wireless sensor networks. This platform was found to be attractive for the implementation of our experiments because of some particularly relevant characteristics: (i) it allowed us to replace the MAC protocol; (ii) the timers available where reasonably precise for our application; (iii) the radio transceiver makes automatic CRC checks and inserts a flag indicating the result of this check along with the packet, and (iv) the spread spectrum modulation used makes data frames resistant to noise and distortion. Hence, collisions due to medium access are the main source of lost frames or corrupted frames.

The experimental application setup was composed of one receiving node and a number of sending nodes. Efforts where made such that the experiments took place under a similar, noise-free, environment. The sending nodes send messages with sequence numbers so that the receiving node can detect when a message has been lost. Additionally, the receiver collected some other statistics, such as total number of replicas and redundant replicas received (by redundant replicas we mean replicas for which a previous replica of the same message has already been received). The time to transmit a replica is 928 us. So, we let one time unit represent 1 ms to improve robustness against time of flight and clock inaccuracies.

First, to acquire the probability that a replica is not correctly received (this is due to noise or distortion), we set up a scenario with one sending node (N_1) and one receiving node (N_2). Node N_1 transmitted 2 replicas per message and N_2 gathered statistics on the number of received replicas. We obtained that the probability of a having a replica lost is approximately 0.002737%. If the events "a replica is lost" were independent, we would expect that the probability that two consecutive replicas are lost is 0.00002737^2. Hence we would expect the probability that a message was lost is 0.00002737^2 as well. However, we observed a 0.00153% probability for messages loss; this indicates that errors are correlated, which was expected.

After that, we ran experiments with different number of nodes, for three different MAC protocols: (i) one where we use our scheme with deterministic Δ:s (HYDRA); (ii) another where we used a similar

Table 3. Parameters of experimental scenarios

m	*Inter-arrival time (ms)*	*MAC Protocol and Configuration*		
		HYDRA (ms)	*RHYDRA (ms)*	*RMAC (ms)*
2	[10,12]	$\Delta_{1,1}$=2 $\Delta_{1,2}$=4	[1,9]	[0,9]
3	[34,42]	$\Delta_{1,1}$=$\Delta_{1,2}$=6 $\Delta_{2,1}$=$\Delta_{2,2}$=2 $\Delta_{3,1}$=$\Delta_{3,2}$=8	[1,16]	[0,33]
4	[86,107]	$\Delta_{1,1}$=$\Delta_{1,2}$=$\Delta_{1,3}$=10 $\Delta_{2,1}$=$\Delta_{2,2}$=$\Delta_{2,3}$=14 $\Delta_{3,1}$=$\Delta_{3,2}$=$\Delta_{3,3}$=8 $\Delta_{4,1}$=$\Delta_{4,2}$=$\Delta_{4,3}$=2	[1,28]	[0,85]
5	[178,222]	$\Delta_{1,1}$=$\Delta_{1,2}$=$\Delta_{1,3}$=$\Delta_{1,4}$=22 $\Delta_{2,1}$=$\Delta_{2,2}$=$\Delta_{2,3}$=$\Delta_{2,4}$=14 $\Delta_{3,1}$=$\Delta_{3,2}$=$\Delta_{3,3}$=$\Delta_{3,4}$=10 $\Delta_{4,1}$=$\Delta_{4,2}$=$\Delta_{4,3}$=$\Delta_{4,4}$=18 $\Delta_{5,1}$=$\Delta_{5,2}$=$\Delta_{5,4}$=2; $\Delta_{5,3}$=74	[1,44]	[1,177]
6	[262,327]	$\Delta_{1,1}$=$\Delta_{1,2}$=$\Delta_{1,3}$=$\Delta_{1,4}$=$\Delta_{1,5}$=26 $\Delta_{2,1}$=$\Delta_{2,2}$=$\Delta_{2,3}$=$\Delta_{2,4}$=$\Delta_{2,5}$=22 $\Delta_{3,1}$=$\Delta_{3,2}$=$\Delta_{3,3}$=$\Delta_{3,4}$=$\Delta_{3,5}$=14 $\Delta_{4,1}$=$\Delta_{4,2}$=$\Delta_{4,3}$=$\Delta_{4,4}$=$\Delta_{4,5}$=18 $\Delta_{5,1}$=$\Delta_{5,2}$=$\Delta_{5,5}$=2; $\Delta_{5,3}$=$\Delta_{5,4}$=58 $\Delta_{6,1}$=$\Delta_{6,2}$=$\Delta_{6,3}$=$\Delta_{6,4}$=$\Delta_{6,5}$=19	[1,52]	[0,261]
7	[314,392]	$\Delta_{1,1}$=$\Delta_{1,2}$=$\Delta_{1,3}$=$\Delta_{1,4}$=$\Delta_{1,5}$=$\Delta_{1,6}$=26 $\Delta_{2,1}$=$\Delta_{2,2}$=$\Delta_{2,3}$=$\Delta_{2,4}$=$\Delta_{2,5}$=$\Delta_{2,6}$=22 $\Delta_{3,1}$=$\Delta_{3,2}$=$\Delta_{3,3}$=$\Delta_{3,4}$=$\Delta_{3,5}$=$\Delta_{3,6}$=14 $\Delta_{4,1}$=$\Delta_{4,2}$=$\Delta_{4,3}$=$\Delta_{4,4}$=$\Delta_{4,5}$=$\Delta_{4,6}$=18 $\Delta_{5,1}$=$\Delta_{5,2}$=$\Delta_{5,3}$=$\Delta_{5,5}$=$\Delta_{5,6}$=2; $\Delta_{5,4}$=142 $\Delta_{6,1}$=$\Delta_{6,2}$=$\Delta_{6,3}$=$\Delta_{6,4}$=$\Delta_{6,5}$=$\Delta_{6,6}$=16 $\Delta_{7,1}$=$\Delta_{7,2}$=$\Delta_{7,3}$=$\Delta_{7,4}$=$\Delta_{7,5}$=$\Delta_{7,6}$=10	[1,52]	[0,313]
8	[534,667]	$\Delta_{1,1}$=$\Delta_{1,2}$=$\Delta_{1,3}$=$\Delta_{1,4}$=$\Delta_{1,5}$=$\Delta_{1,6}$=$\Delta_{1,7}$=38 $\Delta_{2,1}$=$\Delta_{2,2}$=$\Delta_{2,3}$=$\Delta_{2,4}$=$\Delta_{2,5}$=$\Delta_{2,6}$=$\Delta_{2,7}$=34 $\Delta_{3,1}$=$\Delta_{3,2}$=$\Delta_{3,3}$=$\Delta_{3,4}$=$\Delta_{3,5}$=$\Delta_{3,6}$=$\Delta_{3,7}$=26 $\Delta_{4,1}$=$\Delta_{4,2}$=$\Delta_{4,3}$=$\Delta_{4,4}$=$\Delta_{4,5}$=$\Delta_{4,6}$=$\Delta_{4,7}$=22 $\Delta_{5,1}$=$\Delta_{5,2}$=$\Delta_{5,3}$=$\Delta_{5,4}$=$\Delta_{5,5}$=$\Delta_{5,6}$=$\Delta_{5,7}$=32 $\Delta_{6,1}$=$\Delta_{6,2}$=$\Delta_{6,3}$=$\Delta_{6,4}$=$\Delta_{6,5}$=$\Delta_{6,6}$=$\Delta_{6,7}$=36 $\Delta_{7,1}$=$\Delta_{7,2}$=$\Delta_{7,3}$=$\Delta_{7,6}$=$\Delta_{7,7}$=2; $\Delta_{7,4}$=$\Delta_{7,5}$=120 $\Delta_{8,1}$=$\Delta_{8,2}$=$\Delta_{8,3}$=$\Delta_{8,4}$=$\Delta_{8,5}$=$\Delta_{8,6}$=$\Delta_{8,7}$=14	[1,76]	[0,533]

scheme, but where the Δ:*s* were random variables within an interval between 1 ms and $(T_i - 1)/(nreplicas(\tau_i) - 1)$ time units, which was named Random HYDRA (RHYDRA) and (iii) finally a third MAC protocol where only one replica is sent at a random time within the interval $[0, T_i - 1]$ time units after the message was requested, which will be referred to as Random MAC (RMAC). The Δ:*s* were obtained from the algorithm are described in Section 4.1. From these Δ:*s*, we derived *z* and T_i; the application message periods in Table 3 are between T_i and $T_i \times 1.25$. Table 3 also shows all the

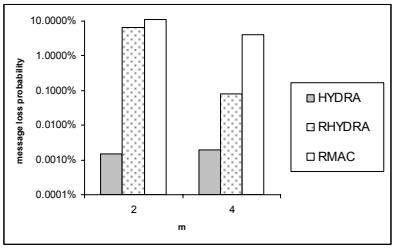


Fig. 9. Message loss ratio of experiments with MicaZ platforms

parameters for the several experimental setups. Random times are represented as intervals and are all uniformly distributed.

The experiments where performed until each node transmitted 100000 messages, for $m = 2$ and $m = 4$. The resulting message loss rate is shown in Figure 9, which is presented in a logarithmic scale. By these results, we can observe that HYDRA obtained a message loss rate always better to the replica loss rate (0.002737%) previously obtained, indicating that noise was the cause for application message loss.

Performing statistically significant experiments with the actual implementations was very time consuming. Therefore, in order to test our protocol further, a simulation model for the protocol in OMNeT++ [16] was implemented. With this model we study the message loss ratio for different numbers of nodes with HYDRA, RHYDRA and RMAC. The parameters of the simulation experimental setup are given in Table 3 as well. The simulator assumes that replicas cannot get lost or corrupted due to noise, but it does model collisions which is the only source of lost messages.

All simulations were executed for a length of 10 simulated hours. For simulations involving random numbers generation, several independent runs were executed to verify the statistical validity of the results. The results of the simulations are given in Figure 10 with respective error bars which are mostly not visible due to the small variation found throughout the simulation runs. Observe that the application message loss for the scheme using deterministic Δ:s is always zero. This is expected as the simulation only models collisions, no noise in transmission was introduced, whereas the other schemes suffer from application message loss.

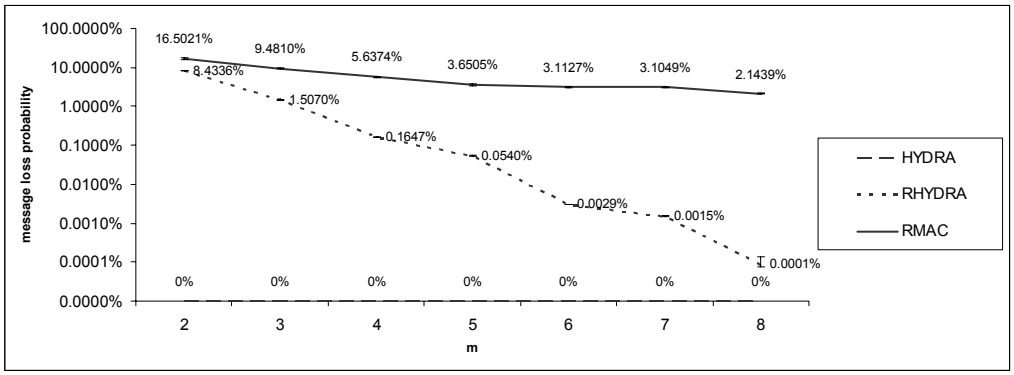


Fig. 10. Message loss ratio in simulation

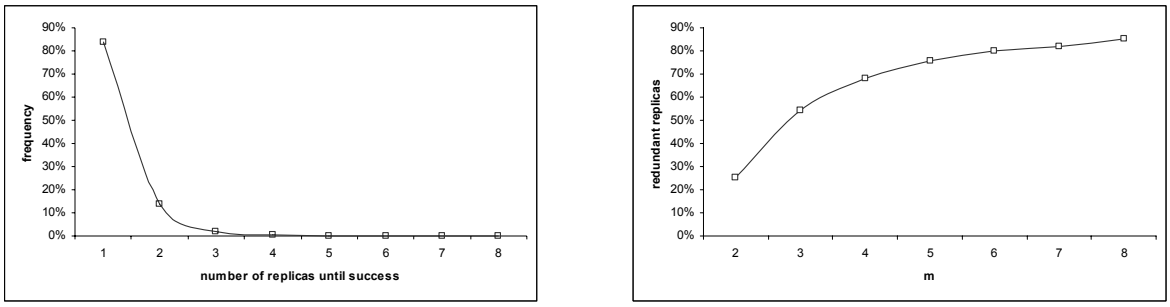
4.3. Support of Hypotheses

§Hypothesis 1. In order to test Hypothesis 1 the time required to implement HYDRA was measured. We spent approximately 7 days on implementing the protocol and running experiments. Almost a third of this time was spend getting familiar with the platform details. The time for coding the protocol was less than a day and we encountered no relevant bugs that were related to the implementation of the protocol. We did however encountered and fixed some bugs related to the platform. This suggests that Hypothesis 1 withstood our test.

§Hypothesis 2. The experiments presented in Section 4.2. corroborate Hypothesis 2.

§Hypothesis 3. Testing Hypothesis 3 is difficult because it is difficult to know if a lost frame is due to a collision or due to noise/distortion. Corrupt CRC may be because of noise or it may be because of collisions. Based on the experiments with the actual implementation of HYDRA in Section 4.2, it results that the number of lost messages is less than the probability of a single message with a single sender being lost; this corroborates our hypothesis that the implementation of our protocol indeed guarantees that *ncollisionfree*(τ_i) replicas are collision-free. Furthermore, we have run simulations during a period of 100 simulated hours for the scheme using deterministic Δ:s for $2 \leq m \leq 8$ and found that no application messages were lost during these simulation runs. This suggests that Hypothesis 3 withstood our test.

§Hypothesis 4. In order to test Hypothesis 4, we considered the experiments used to test Hypothesis 3



a) **b)**

Fig. 11. The frequency of the number of necessary replicas and variation of the number of redundant replicas with m.

and acquired both the frequency of the number of replicas necessary until the first replica is transmitted without collision (Figure 11a) and the number of redundant replicas for $2 \leq m \leq 8$ (Figure 11b). Observe, in Figure 11b, that for the case with $m = 8$ we obtain that approximately 84% of the first replicas of a message are collision-free. Hence, if most (but not all) links are bidirectional and we would have used a scheme where the receiver sends an acknowledgement when it receives the first successful replica then in approximately 84% of the cases the sender N_s only needs to send one replica. Hence, in 84% of the cases, N_s can send 7 non real-time messages instead of the replicas that N_s would normally send. This discussion supports, Hypothesis 4.

In order for the acknowledgement scheme described above to be efficient, it is necessary that the time required to send acknowledgements is negligible. Nonetheless, it could easily been the case by using longer packets (say 1500 bytes) for data and short packets (say 20 bytes) for the acknowledgements. But unfortunately this is not supported by our experimental platform so we did not implement it.

5. Discussion and Previous Work

Bidirectional links are useful for MAC and routing protocols. Let us categorize a MAC protocol based on whether it can suffer from collisions. If it can suffer from collisions then a sender typically retransmits data packets until it receives an acknowledgement from the intended receiver. Typically the data and the acknowledgement are transmitted on the same link, and so this requires bidirectional links. This is exemplified by ALOHA [17] and some CSMA/CA protocols. MAC protocols that are collision-

free typically rely on that senders receive feedback from the intended receiver. Some protocols, such as MACA [18] do this using an RTS/CTS dialog before the data packet is sent. In other protocols, a receiver sends a busy tone when it receives a packet and other senders can hear thus avoiding the collision. Common to all these MAC protocols is that they depend on bidirectional links. Routing algorithms also typically assume that links are bidirectional, being one notable exception the Dynamic Source Routing (DSR) [19]. We can conclude that the current communication protocols are heavily dependent on bidirectional links.

Unfortunately, unidirectional links are not rare and they are caused by a variety of reasons such as: (i) differences in antenna and transceivers even from the same type of devices; (ii) differences in the voltage levels due to different amounts of stored energy in the battery; (iii) different properties of the medium in different directions (anisotropic medium) and (iv) different interferences from neighboring nodes.

Given that protocol stacks tend to be implemented based on the assumption that unidirectional links do not exist, three techniques have been used to "hide" the unidirectional links: (i) *tunneling*; (ii) *blacklisting* and (iii) *transmission power increase*. If a link from node u to v is unidirectional, the tunneling approach attempts to find a path from v to u and give higher level protocols the illusion of a link from v to u. In order to achieve this, some routing functionality has to be performed at the lower layers of the protocol stack [20]. Packets sent across the tunnel have larger delays because they have to cross several hops. This is not too important though, because often the tunnel is used only for acknowledgements to packets that were sent across the unidirectional link. It is important however to avoid the *ACK explosion* [21]. Consider a unidirectional link from node N_u to node N_v. Consider also that there is a path from N_v to N_u. A data message has been sent across the link N_u to N_v and now the node N_v should send an ACK across the path back to N_u. However, the path from N_v to N_u contains a unidirectional link too. This link is from node N_x to N_y. When a packet has crossed the hop from N_x to N_y, node N_y should send an ACK to N_x. In order to do this, it may have to find a path to N_x. It is possible that

the path from N_y to N_x uses the link from N_u to N_v. This may generate an ACK from N_u to N_v and this process continues forever.

The technique of blacklisting detects unidirectional links when sending data messages, and does not use them in the future. The technique "hello" is similar but here “hello” messages are exchanged so a node *i* knows about the existence of a neighbor and whether they can hear *i*. This exchange is periodic and occurs regardless of whether the nodes are involved in routing data traffic or not. These techniques are sometimes called *ignoring* [22] or *check symmetry* [1]. Yet another technique to ignore unidirectional links is to treat it as a fault. This technique has been applied in conjunction with Ad-hoc On-Demand Distance Vector Routing (AODV) and it works as follows. When a source node attempts to find a route to the destination, it floods the network with Route-Request (RREQ) packets. In the normal AODV, when RREQ packet reaches a node which knows a route to the destination, this node sends Route Reply (RREP) back on the same paths as the RREQ was sent on. With the normal AODV, RREP would fail on a unidirectional links but instead this technique attempts to find a new path back to the source. When it finds a node with RREQ it knows a route back to the source node [23]. A similar scheme was proposed in [1] called *Bidirectional flooding*. Another technique (which we call “transmission power increase”) lets a downstream node of a unidirectional link to temporarily increase its power for sending responses such as acknowledgements and clear-to-send [24]. This technique is based on the sender to piggyback its geographical position obtained by GPS and the receiver should use this information to calculate the distance, which in turn is used to know how much the transmission power should be increased. We think the idea of increasing transmission power is interesting but in [24] the authors do neither give any details on how this increase transmission power is computed nor state the assumed path loss. Common to these techniques is that they require no or minimal changes to routing protocols.

Several routing algorithms have been proposed for unidirectional links. A common challenge that faces routing with unidirectional links is *knowledge asymmetry*; that is, if a link from *u* to *v* is unidirectional, only *v* can detect the existence of the link (by hearing a broadcast from *u*) but *u* is the one

that will use the knowledge of the link for routing purposes. One technique builds on *distance vector*. The classic distance vector algorithm maintains a vector at each node and this vector stores the hop count to every other node N_i and the next node that should be used for forwarding to this node N_i (sometimes a sequence number is added too; it is used for updates).

Consider a node N_u with a neighbor N_v. Node N_v knows a route to node N_w. The number of hops from N_u to N_w is no larger than the number of hops from N_v to N_w plus one. If the link N_u to N_v is bidirectional this fact can be easily exploited in the design of a routing protocol because the length of the route N_v to N_w can simply be communicated over one hop to N_u. However, if the link N_u to N_v is unidirectional this is more challenging.

One extension of distance vector [22] however stores all distance vectors of all nodes in the network (hence it requires $O(m^2)$ storage). Another extension [25] sends information "downstream" until every node knows a circuit to itself. The node selects the shortest circuit and informs its upstream neighbors, and then the standard distance vector algorithm is used. Other techniques [20, 26] and [27] disseminate link state information across a limited number of hops. This is based on the assumption that the reverse path of a unidirectional link is short and this assumption has been supported empirically [20].

Pure link-state routing disseminates the topology information to all nodes and then the routes are calculated. This avoids the problem of asymmetric information (mentioned earlier) but the overhead of this scheme is large already.

In order to reduce the routing cost in networks with unidirectional links, it has been suggested that a subset of nodes should be selected and only they should maintain routing information about all nodes in the network. It is required that all nodes which are not in this subset have a link from the subset and a link to the subset. Algorithms for selecting this subset of nodes have been proposed and they have very low overhead [28].

It has often been pointed out that unidirectional links should be avoided altogether because existing MAC protocols cannot deal with them (as we already mentioned MACA, which was the basis for the

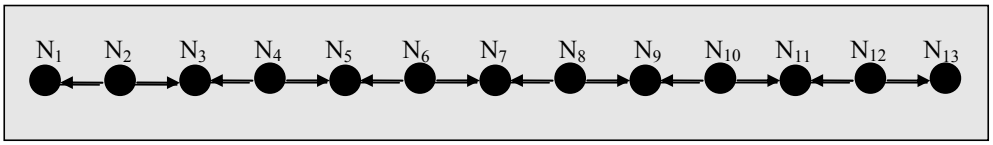
RTS/CTS dialogue in IEEE 802.11 relies on bidirectional links). But recently, this view has been challenged. For example [29] mentioned that their routing protocol works well for multicast and that it could be used for unicast routing as well – if there was a MAC protocol for unidirectional links.

To the best of our knowledge, the only previous MAC protocols that work for unidirectional links require synchronized clocks and it suffers from (an unbounded number of) collisions [9].

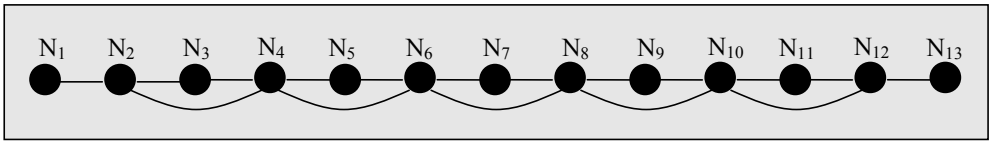
The technique in [9] addresses medium access control on unidirectional links. The technique generates pseudo-random numbers on each node and these numbers act as priorities. Every node knows the seed of the pseudo-random numbers on other nodes and hence a node knows if it has a higher priority than its neighbors. If it has then it is the winner; otherwise it is not a winner. If it is a winner then it transmits in that time slot. Every new time slot, a new pseudo-random number is generated. This protocol is designed to deal with hidden nodes in the following way: if a node N_i has a neighbor with higher priority two hops way then node N_i simply does not transmit. This scheme is collision-free but it depends on synchronized clocks and the MAC protocol does not take deadlines into account in its decisions. Our protocol does not have those shortcomings.

One of our schemes to determine $\Delta{:}s$ depends on the use of prime numbers. This has been observed in nature, where cicadas sleep and periodically wakes up every 7, 9 or 11 years. It has been explained that this minimizes "collisions" with predatory animals with other periods [30].

We have also borrowed ideas from real-time scheduling theory and in particular: static-priority preemptive scheduling of uniprocessor system. Our concept of message stream is equivalent to a task in processor scheduling theory. Algorithm 3, the algorithm that assigns small $\Delta{:}s$ to message streams with a small D_i is similar to the deadline-monotonic priority-assignment scheme [31]. Our equation for computing the number of collisions has similarities to a sufficient schedulability tests [32] and our iterative procedure (in Algorithm 2) is similar to the response-time calculation [33]. The argument for correctness of the iterative procedure (in Algorithm 2) is similar to the argument [34] why the iterative



(a) Connectivity graph.



(b) Interference graph.

Fig. 12. An example of how the performance of our MAC protocol can be significantly improved if the topology is known. If we assume the topology is unknown, then we must assume that all 13 nodes can transmit simultaneously and can collide. This gives us (using Section 3) $\Delta_1 = 22$, Δ_2=26, $\Delta_3 = 34$, $\Delta_4 = 38$, $\Delta_5 = 46$, $\Delta_6 = 58$, $\Delta_7 = 62$, $\Delta_8 = 74$, $\Delta_9 = 82$, $\Delta_{10} = 86$, $\Delta_{11} = 94$, $\Delta_{12} = 106$, $\Delta_{13} = 118$ and $z = 1417$. The interference graph is shown in (b). We observe that every node has at most 4 links. This gives us $m = 5$, and we calculate the following Δ:*s*: 6, 10, 14, 22, 26. Now we can assign $\Delta_1 = 6$, $\Delta_2 = 10$, $\Delta_3 = 14$, $\Delta_4 = 22$, $\Delta_5 = 26$, $\Delta_6 = 6$, $\Delta_7 = 10$, $\Delta_8 = 14$, $\Delta_9 = 22$, $\Delta_{10} = 26$, $\Delta_{11} = 6$, $\Delta_{12} = 10$, $\Delta_{13} = 14$. Observe that we reuse Δ:*s* and this does not cause any collisions. In this way, we obtain $z = 105$, which is significantly lower.

response-time calculation finds a solution to the equation if and only if there is a solution. There is a difference however in that the equation used in the response-time calculation [33] is a necessary and sufficient schedulability test; whereas the inequalities that we use in our schedulability test is only sufficient. For this reason, our schedulability test is only sufficient; it is not necessary.

In the theory we assumed that $tof = 0$. We can easily extend the theory for the case when $tof > 0$. We can do it as follows. Select the time unit such that $(1 - tof)$ is the time it takes to transmit a replica. Hence, if $tof = 1\mu s$ and the time to transmit a replica is 1 ms, then let 1.001 ms denote a time unit.

In the paper, we assumed topology is not known. However, if the topology is known we can perform significantly better (assuming that we also know the interference graph). Every node in the connectivity graph also exists in the interference graph. The links in the interference graph are non-directed. The links in the interference graph cannot simply be computed from the connectivity graph. However, there are some links in the interference graph that are necessary. Consider two nodes in the connectivity graph N_i and N_j. If there is a link from N_i to N_j or from N_j to N_i then there is a link between N_i and N_j in the interference graph as well. If there is a node N_k with a link from N_i to N_k and a link from N_j to N_k then there is a link between N_i and N_j in the interference graph as well. Figure 12 illustrates this. In general

this requires solving the problem Achromatic Number which is known to be NP-hard (see page 191 in [35]) but several approximation algorithms are available. We can see from Figure 12 that the z is unaffected by the size of the network; only the number of neighbors 2-hops away matters. Hence, this approach is efficient in large networks if they are not dense.

6. Conclusions and Future Work

We have presented the first MAC protocol that can guarantee that the time from when an application requests to transmit until the message is transmitted is bounded even in the presence of unidirectional links and without using synchronized clocks or taking advantage of topology knowledge. A schedulability analysis technique was proposed for sporadic message streams. We implemented the protocol and observed (i) the effort required to implement it is small, (ii) by observing the number of lost messages we found that the implementation guaranteed that at least one replica of a message is collision-free and (iii) the number of lost messages at the receiver is significantly lower using our protocol than a replication scheme with random delays between replicas. We also run a scheme with random time for transmission with only one replica; this should perform similar to ALOHA [17], and found that our protocol performed significantly better.

We consider for future work (i) the development of even better techniques for computing Δ:s and (ii) schedulability analysis techniques with probabilistic guarantees but with fewer replicas and hence lower overhead.

References

[1] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic, "Impact of Radio Irregularities on Wireless Sensor Networks," presented at International Conference on Mobile Systems, Applications, and Services, 2004.

[2] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," presented at Conference On Embedded Networked Sensor System, Los Angeles, California, USA, 2003.

[3] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," presented at Conference On Embedded Networked Sensor Systems, Los Angeles, California, USA, 2003.

[4] A. Cerpa, N. Busek, and D. Estrin, "SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments," UCLA Center for Embedded Network Sensing (CENS), Technical report 0021 September 2003.

[5] D. Ganesan, D. Estrin, A. Woo, A. Culler, B. Krishnamachari, and B. Wicker, "Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks," 2002.

[6] D. Kotz, C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Ellliot, "Experimental Evaluation of Wireless Simulation Assumptions," presented at International Workshop on Modelling Analysis and Simulation of Wireless and Mobile Systems, 2004.

[7] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin, "Statistical Model of Lossy Links in Wireless Sensor Networks," presented at ACM/IEEE Fourth International Conference on Information Processing in Sensor Networks (IPSN'05), Los Angeles, California, USA, 2005.

[8] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Models and Solutions for Radio Irregularity in Wireless Sensor Networks," ACM Transactions on Sensor Networks, 2006.

[9] L. Bao and J. J. Garcia-Luna-Aceves, "Channel access scheduling in Ad Hoc networks with unidirectional links," presented at Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications, Rome, Italy, 2001.

[10] Crossbow, "MICAz - Wireless Measurement System Product Datasheet," 2005.

[11] 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.

[12] "AMPL, www.ampl.com."

[13] "LOQO,http://www.princeton.edu/~rvdb/."

[14] 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.

[15] J. Hill, "System Architecture for Wireless Sensor Networks," in Computer Science Department: University of California, Berkeley, 2003.

[16] A. Varga, "OMNeT++ Discrete Event Simulation System," Tech. University of Budapest, Budapest, User Manual; June, 15th 2003.

[17] N. Abrahamson, "The ALOHA system - another alternative for computer communications," presented at 1970 fall joint computer communications, AFIPS Conference Proceedings, Montvale, 1970.

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

[19] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in Mobile Computing, T. Imielinski and H. Korth, Eds.: Kluwer Academic Publishers, 1996.

[20] V. Ramasubramanian, R. Chandra, and D. Mossé, "Providing a Bidirectional Abstraction for Unidirectional Ad Hoc Networks," presented at IEEE INFOCOM, New York NY, 2002.

[21] S. Nesargi and R. Prakash, "A Tunneling Approach to Routing with Unidirectional Links in Mobile Ad-Hoc Networks," presented at Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN), Las Vegas, 2000.

[22] R. Prakash, "A routing algorithm for wireless ad hoc networks with unidirectional links," Wireless Networks, vol. 7, pp. 617 - 625, 2001.

[23] M. K. Marina and S. R. Das, "Routing performance in the presence of unidirectional links in multihop wireless networks," presented at Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, Lausanne, Switzerland, 2002.

[24] D. Kim, C.-K. Toh, and Y. Choi, "GAHA and GAPA : Two Link-level Approaches for Supporting Link Asymmetry in Mobile Ad Hoc Networks," IEICE Transaction on Communication, vol. E-86B, pp. 1297-1306, 2003.

[25] M. Gerla, L. Kleinrock, and Y. Afek, "A Distributed Routing Algorithm for Unidirectional Networks.," presented at Proceedings of IEEE GLOBECOM, 1983.

[26] L. Bao and J. J. Garcia-Luna-Aceves, "Unidirectional Link-State Routing with Propagation Control," presented at Proceedings of IEEE Mobile Multimedia Communications (MoMuC), Tokyo, Japan, 2000.

[27] T. Ernst, "Dynamic Routing in Networks with Unidirectional Links," in Sophia Antipolis: INRIA, 1997.

[28] J. Wu and H. Li, "Domination and Its Applications in Ad Hoc Wireless Networks with Unidirectional Links," presented at Proceedings of the 2000 International Conference on Parallel Processing, Toronto, Ontario, Canada, 2000.

[29] M. Gerla, L. Y.-Z., J.-S. Park, and Y. Yi, "On Demand Multicast Routing with Unidirectional Links," presented at Proceeding of IEEE Wireless Communications & Networking Conference (WCNC), New Orleans, LA, USA., 2005.

[30] E. Goles, O. Schulz, and M. Markus, "Prime number selection of cycles in a predator-prey mode," Complexity, vol. 6, pp. 33 - 38, 2001.

[31] 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.

[32] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings, "Hard Real-Time Scheduling: The Deadline-Monotonic Approach," presented at Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software, 1991.

[33] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," The Computer Journal, British Computer Society, vol. 29, pp. 390-395, 1986.

[34] M. Sjödin and H. Hansson, "Improved Response-Time Analysis Calculations," presented at Real-Time Systems Symposium, Madrid, Spain., 1998.

[35] M. R. Garey and D. S. Johnson, Computers and Intractability A guide to the Theory of NP-Completeness New York: W. H. Freeman and Company, 1979.