Technical Report

Slotted WiDom: Schedulability Analysis and its Experimental Validation

Maryam Vahabi and Björn Andersson

HURRAY-TR-110403

Version:

Date: 04-14-2011

Slotted WiDom: Schedulability Analysis and its Experimental Validation

Maryam Vahabi and Björn Andersson

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:

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

Abstract

WiDom is a wireless prioritized medium access control protocol which offers a very large number of priority levels. Hence, it brings the potential to employ non-preemptive static-priority scheduling and schedulability analysis for a wireless channel assuming that the overhead of WiDom is modeled properly. One schedulability analysis for WiDom has already been proposed but recent research has created a new version of WiDom (we call it: Slotted WiDom) with lower overhead and for this version of WiDom no schedulability analysis exists. In this paper we propose a new schedulability analysis for slotted WiDom and extend it to work also for message streams with release jitter. We have performed experiments with an implementation of slotted WiDom on a real-world platform (MicaZ). We find that for each message stream, the maximum observed response time never exceeds the calculated response time and hence this corroborates our belief that our new scheduling theory is applicable in practice.

Slotted WiDom: Schedulability Analysis and its Experimental Validation

Maryam Vahabi[1] and Björn Andersson[21]
[1] CISTER Research Unit, Polytechnic Institute of Porto (ISEP/IPP), Porto, Portugal
[2] Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA
mmvi@isep.ipp.pt, baandersson@sei.cmu.edu

Abstract

WiDom is a wireless prioritized medium access control protocol which offers a very large number of priority levels. Hence, it brings the potential to employ non-preemptive static-priority scheduling and schedulability analysis for a wireless channel assuming that the overhead of WiDom is modeled properly. One schedulability analysis for WiDom has already been proposed but recent research has created a new version of WiDom (we call it: Slotted WiDom) with lower overhead and for this version of WiDom no schedulability analysis exists. In this paper we propose a new schedulability analysis for slotted WiDom and extend it to work also for message streams with release jitter. We have performed experiments with an implementation of slotted WiDom on a real-world platform (MicaZ). We find that for each message stream, the maximum observed response time never exceeds the calculated response time and hence this corroborates our belief that our new scheduling theory is applicable in practice.

1. Introduction

Wireless communication in embedded computer systems is spreading and it is an enabler for many future applications such as (i) wireless sensor networks (WSN) for environmental monitoring, (ii) wire replacement, particularly in automation, (iii) collaborative robotics, (iv) inter-vehicle communication and (v) smart materials.

These applications tend to have real-time requirements. The scientific community has already created solutions to fulfill real-time requirements. The most well-known of these solutions is the Generalized Rate-Monotonic Analysis which allows designers to prove in advance that all deadlines are met at run-time. This analysis is matured into a fully fledged theory for uniprocessor systems and for a wired communication channel. However, it is not well-developed for wireless channels. Creating a rate-monotonic analysis for wireless channels requires that:

R1. A prioritized Medium Access Control (MAC) protocol should exist for a wireless channel. This protocol grants, among all computer nodes that request to transmit, the right to transmit on the channel to the computer node with the highest priority message;

R2. The overhead related to the arbitration of the prioritized MAC protocol grows slowly with the number of priority levels;

R3. The overhead related to the arbitration of the prioritized MAC protocol should be low;

R4. A schedulability analysis should exist for the prioritized MAC protocol.

Unfortunately, the current state of art cannot fulfill all these requirements. There exists a prioritized MAC protocol, the Controller Area Network (CAN) [1], for a wired channel that offers many priority levels (hence fulfilling R2). A wireless version of CAN bus has been proposed and dubbed WiDom [2] which provides a corresponding schedulability analysis as well (hence fulfilling R1, R2 and R4). The problem with this protocol was that it imposes a large overhead (missing R3). On this account, researchers have developed a new version [3] of WiDom (we call it *slotted WiDom*) which offers low overhead (hence fulfilling R1, R2 and R3) but no schedulability analysis is available for it. The development of a schedulability analysis for slotted WiDom would, however, offer us the missing piece in fulfilling the four requirements above.

In this paper, we present schedulability analysis for slotted WiDom and extend it to model release jitter. Previous work [2] has already validated the schedulability of WiDom by running experiments, measuring response times of messages and comparing the maximum measured response time to the calculated upper bound. But this validation was applicable only to the schedulability analysis of the previous WiDom—not-slotted WiDom. In this paper, we experimentally validate our new schedulability analysis for slotted WiDom. We run experiments in an office environment and find that in this environment (i) there are packet drops but it is not common (packet drop rate is less than 1%) and (ii) all packets that are not dropped meet their deadlines.

The remainder of this paper is organized as follows: In Section 2 we first present a brief background on schedulability analysis of static-priority scheduling on CAN bus and the previous version of WiDom.

Following an introduction about the mechanism of the WiDom protocol in Section 3, we provide a brief explanation of the extended hardware used in slotted WiDom design. In Section 4 we show the response time analysis for slotted WiDom according to new timing requirements of the add-on hardware. The experimental evaluation of the protocol is then presented in Section 5 followed by the conclusion in Section 6.

2. Background on schedulability analysis of non-preemptive static-priority scheduling

The schedulability analysis presented in this paper is based on previously proposed analysis of WiDom which follows the concept of schedulability analysis of CAN bus. Therefore, we first provide the necessary background on this analysis and then discuss the presented analysis in Section 4.

2.1. Controller area network (CAN)

The CAN bus implements non-preemptive static-priority scheduling on a wired channel and for this reason, early researchers [4] realized that the uniprocessor preemptive static-priority scheduling theory [5] could be modified for non-preemptiveness and applied to CAN. Davis et al. [6] proposed the first correct analysis of the CAN bus by revising this analysis by considering the fact that in the non-preemptive static-priority scheduling, for a given message *m*, a higher priority message can be awaiting for transmission when message *m* completes transmission. Thus the busy period can extend beyond the period of message *m*. To be more accurate in the calculation they first determine the duration of level-*m* busy period as follows:

$$t_m = B_m + \sum_{\forall i \in hp(m) \cup m} \left\lceil \frac{t_m + J_i}{T_i} \right\rceil \times C_i \qquad (1)$$

where $hp(m) \cup m$ is the set of message streams with priority *m* or higher assuming that all priorities are unique; B_m is maximum blocking time that can be imposed by a lower priority message; release jitter or queuing jitter [4], J_i , is defined as the largest difference between initiating time of the event and the time in which that message has been queued; C_i and T_i are transmission time and minimum inter-arrival time of message stream *i* respectively. Then for calculating the response time for message stream *m*, the response time for all the instances of this message stream located in the level-*m* busy period should be calculated. Finally the response time of a message instance which has the largest value among other instances during the busy period will be considered as the Worst Case Response Time (WCRT) of the message stream *m* and is computed as follows:

$$R_m = \max_{q=0,\ldots,Q_m-1} (J_m + w_{m,q} - qT_m + C_m) \qquad (2)$$

where Q_m is the number of message instances located in the level-*m* busy period and is given by:

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \qquad (3)$$

and $w_{m,q}$ can be defined as follows:

$$w_{m,q} = B_m + qC_m + \sum_{\forall i \in hp(m) \cup m} \left\lceil \frac{w_{m,q} + J_i + \tau_{bit}}{T_i} \right\rceil \times C_i. \qquad (4)$$

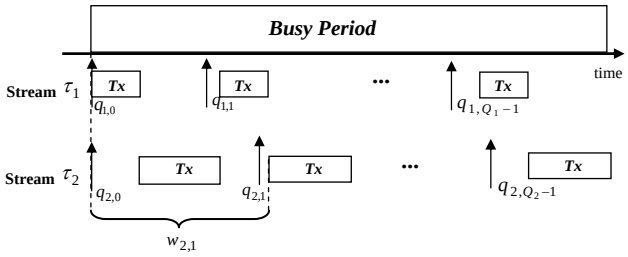


Figure 1. An Example of $w_{i,q}$.

2.2. Wireless Dominance MAC (WiDom)

The existing response time analysis [2] of WiDom follows the same concept of non-preemptive static-priority scheduling of CAN bus and provides the feasibility test based on WCRT analysis. Two major differences between WiDom and CAN bus analysis are: (i) WiDom needs to incorporate the time needed for synchronization purpose and (ii) it assumes there is no release jitter for any message stream.

As described earlier, the WCRT of a message stream is the longest response time among all of its message instances *q* that request for transmission during a period of time which is called busy period, so:

$$R_i = \max_{q=0,\ldots,Q_i-1} \left(w_{i,q} + C_i'' - q \times T_i \right) \qquad (5)$$

where Q_i is given by:

$$Q_i = \left\lfloor \frac{L_i}{T_i} \right\rfloor + 1 \qquad (6)$$

and L_i is the length of the longest level-*i* busy period and can be formulated as follows:

$$L_i = \max_{j \in lp(i)} \left(C_j' - Q_{bit} \right) + \sum_{j \in hp(i) \cup i} \left\lceil \frac{L_i}{T_j} \right\rceil \times C_j'' \qquad (7)$$

where $hp(i)$ is similarly the set of message streams with priority higher than *i*, and $lp(i)$ is the set of message streams with priority lower than that. Chip duration Q_{bit} is the time granularity which is similar to τ_{bit} in CAN analysis. In the current implementations of WiDom [3,4], the radio uses direct-sequence spread-spectrum (DSSS) in which every 4 bits is modulated as 32 chips so that the data rate reaches 2Mchip/s which is

equivalent to 250 Kbits/s. For such a platform, we have Q_{bit} =4/250000=16μs. C_i'' is the time span needed to finish transmission. It consists of synchronization time, *F*, together with tournament duration, C_i'. The longest time from the start of the busy period to the time in which instance q begins transmission successfully ($w_{i,q}$) — see Figure 1, is given by:

$$w_{i,q} = q \times C_i'' + \max_{j \in lp(i)} \left(C_j' - Q_{bit} \right) \quad (8)$$

$$+ \sum_{j \in hp(i)} \left\lceil \frac{w_{i,q} + F + E + \max(TFCS, SWX) + H + Q_{bit}}{T_j} \right\rceil \times C_j''$$

where *F* is a long period of silence that nodes should wait before contending for the channel and *E* is the duration of time that is considered for compensating clock drift between the nodes and it is also used to guarantee that all nodes have time to listen for *F* time units of silence — see Figure 2. Time For Carrier Sense (*TFCS*) is the duration of time that a node needs for detecting a carrier transmission. In order to have a good perception of these parameters it is necessary to know how WiDom works. In the next section we will describe the functioning of WiDom in brief.

3. Background on the WiDom protocol

We will first describe the initial WiDom mechanism. We will not use this version of WiDom since it has large overhead. Understanding its operation however is useful for understanding slotted WiDom, a more recent version which has much smaller overhead and is the one we will use in the remainder of this paper.

3.1. WiDom without master node (initial WiDom)

As mentioned earlier, WiDom is a prioritized MAC protocol for wireless networks and hence the message with the highest priority (corresponding to the lowest priority number) is granted the channel. When messages contend for the channel, a conflict resolution phase (called *tournament*) similar to the dominance/binary countdown arbitration [7] is performed. During the tournament nodes transmit the priority of the message contending for the medium bit-by-bit. A bit is said to be dominant if it is "0"; it is said to be recessive if it is "1". The protocol is composed of three phases, namely: synchronization, tournament and data exchange — see Figure 2. The synchronization is needed to provide a common reference point in time so that all nodes can start the competition at the same time. Hence, the synchronization should happen before the tournament and finally a node that wins the tournament starts transmission.

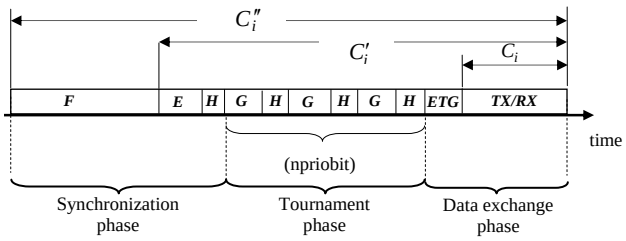


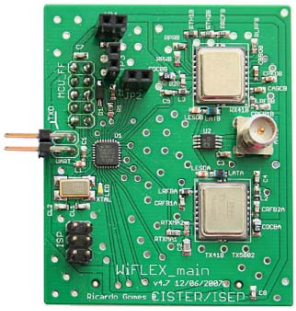
Figure 2. Timing order of WiDom.

In this version of WiDom, at the start of synchronization, nodes should wait for a long period of idle time *F*— see Figure 2, such that no node disrupts an ongoing tournament. Then nodes with a pending message wait for another time span *E* to compensate for the potential clock drift and also ensuring that all nodes have enough time to listen for *F* time units. Afterwards, nodes start sending a carrier pulse for a duration of *H* time units that signals the start of a tournament and establishes a common time reference. To do so, they have to switch from receive mode to send mode which takes *SWX* units of time. By sending this signal, all nodes restart their timers and synchronization ends.

In the tournament phase if a node loses the contention of a bit (i.e. it transmits a recessive bit and receives a dominant bit), it does not continue further bits and only proceed listening to the medium to find out the priority of the winner. If a node does not lose the contention during the current bit it will proceed with the contention for the next bit. The nodes that have dominant bit, start transmitting a pulse of carrier for duration of *H* units of time, while nodes with recessive bit, perform carrier sensing. Also, note that the fact that wireless transceivers are not able to send and transmit simultaneously poses no problem to WiDom since when a node has a dominant bit, there is no need for reception and when a node has a recessive bit, it sends nothing; it performs carrier sensing. There is also a guarding time interval *G* to separate pulses of carrier wave. 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. At the end of the tournament, the node that does not receive a pulse wins the competition and waits for *ETG* time units before starting data transmission.

3.2. WiDom with master node (slotted WiDom)

The aforementioned timing order belongs to the primary implementation of WiDom which is based on off-the-shelf WSN platforms. Most WSN platforms use the Chipcon CC2420 [8] that does not offer the desirable characteristics needed for WiDom



(a) WiFLEX_main board

(b) WiFLEX_daughter board

(c) WiFLEX_daughter board stacked on the WiFLEX_main board

(d) WiFLEX platform stacked on Firefly sensor

(e) WiFLEX platform stacked on MicaZ sensor

Figure 3. Hardware platform.

implementation and introduces a large overhead on its functionality. This overhead mainly comes from (i) the switching time between transmission and reception mode and (ii) the time needed to perform carrier sensing in order to detect priority bits of other nodes. To alleviate this problem researchers develop an add-on board platform that can be plugged into common WSN platforms such as MicaZ [9] and FireFly [10] — see Figure 3.

In slotted WiDom [3] all sensor nodes are equipped with the extended add-on board (called WiFLEX) and a more powerful node (master node) is used to broadcast synchronization pulses periodically on a separate radio channel. That is why the term "WiDom with master node" is also used for this implementation of WiDom. We might use the term "slotted WiDom" or "WiDom with master node" interchangeably in the rest of the paper. The extended platform is composed of two small boards namely WiFLEX_main board (simply main board), Figure 3(a) and WiFLEX_daughter board (daughter board), Figure 3(b). The main board sends and receives pulses during the tournament while the daughter board which is also known as WiFLEX_rxsync is merely responsible for receiving synchronization pulses. Figure 3(c) shows how these two boards are assembled. The main board is equipped with a low-power Micro-Controller Unit (MCU) in order to run the MAC protocol on both WiFLEX and host platform concurrently and provide a mechanism for higher level communication between WiFLEX and WSN platform. The MCU controls two independent radio modules embedded in the main board: (i) one transmitter and (ii) one receiver that share a single antenna. This antenna is assigned to each radio module through a high-frequency switch under MCU supervision. Another single receiver module has been used in the daughter board which is always ready to receive the synchronization signal. The advantage of using the separate receiver (daughter board) is the possibility of setting it perpetually in reception mode and eliminating the switch time and finally maintaining accurate synchronization. Furthermore, by utilizing out-of-band signalling for synchronization, nodes are not forced to wait for a long duration of F time units which reduces the overhead. The receiver devices on both main and daughter board use amplifier-sequenced hybrid (ASH) technology that A.C. couples[1] the data by use of a series capacitor. This is done to remove the D.C. offsets in the amplifiers and comparators. Having poor D.C. balanced data (large number of consecutive 1's or 0's) results in having an unreliable receiver [11]. To avoid this problem it is best to use data that is D.C. balanced. We say a data is D.C. balanced when the number of 1's and the number of 0's of that data in a period of time are equal. To provide a desirable data for receiver a bit stuffing technique is exploited during priority bit exchange (tournament phase).

[1] A.C. coupling is referred to the transfer of energy to different devices linked together through an electrical network.

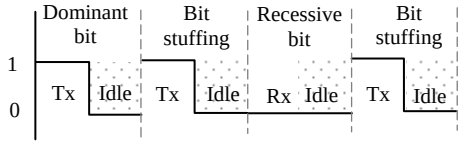


Figure 4. Dominant and recessive signal sequence with bit stuffing.

The principle behind this technique is to introduce redundant information to maintain channel activity. To do so, a dominant bit is coded as a "1" + "0" signal sequence and a recessive bit as two consecutive "0" bits and then introducing a bit stuffing composed of a "1" + "0" signal sequence after each dominant or recessive bit — see Figure 4. Although the utilized bit stuffing technique cannot provide a perfect D.C. balanced data, but the achieved D.C. balance is satisfactory for a correct A.C. coupling which allows correct data recovery at the receiver.

The same policy is also applied in the synchronization signal transmission. To maintain channel activity, a burst of consecutive sequence of "0" + "1" with same pulse duration is used. A longer duration of "1" pulse is then considered to announce the reception of a synchronization signal — see Figure 5.

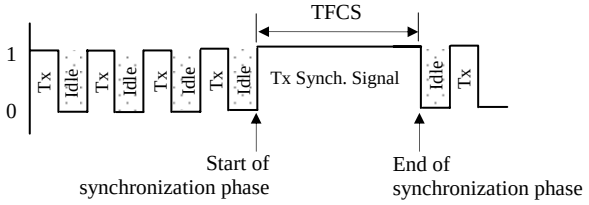
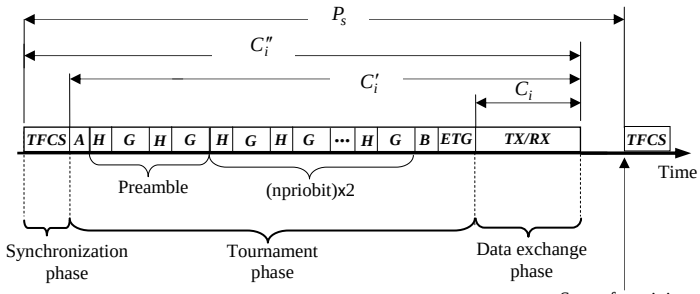


Figure 5. Synchronization signal burst.

Another important issue for receiving a correct data is to extract or synchronize the Real-Time Clock (RTC) to transmitted data [12]. In the tournament phase the radio module on the main board exchanges the data only during priority bit exchange and after that it stays idle until next synchronization signal reception. Since the communication is not always maintained we may lose the synchronization of the RTC for sampling the output data of the receiver. To solve this problem, a special start symbol or preamble has been introduced at the beginning of the tournament phase. Two consecutive dominant bits is sufficient to allow a correct detection between noise and the start of an incoming data without imposing high overhead.

4. Calculating response time

We now compute the WCRT of a given message stream. We develop this analysis by considering release jitter and assuming that all nodes are located in a single broadcast domain.



A: Transferring priority from MicaZ to WiFLEX (PRIO_TRA)
B: Transferring winner priority from WiFLEX to MicaZ (WIN_PRIO)

Figure 6. Timing order of WiDom with master node.

Nodes are equipped with the add-on board circuitry and there is a master node that sends pulses periodically on a separate channel to announce the start of the tournament. The following subsections describe the new developed schedulability analysis for slotted WiDom and present the calculated response time accordingly.

4.1. Response time analysis

So far we have explained some hardware issues to make it easy following the new timing order of WiDom with master node. As mentioned earlier, immediately after receiving the synchronization signal, nodes start executing the tournament. In the tournament phase before going through the competition, priority of the enqueued message should be notified to the WiFLEX board and at the end of that; the priority of the winner should be reported back to the WSN host platform. Thus we need to consider two time intervals (A and B) for boards' communications in the slotted WiDom — see Figure 6. This figure shows the timing order of slotted WiDom. Each slot starts with the reception of synchronization signal and then two dominant bits as preamble are sent to indicate the start of a tournament. Immediately afterwards nodes send their priorities bit-by-bit including one bit stuffing after each dominant or recessive bit. At the end of the competition, WiFLEX reveals the priority of the winner to its host platform and finally the winner sends its message after a small duration of *ETG* time units. Having these new timing requirements we obtain the value of C_i' and C_i'' as follows:

$$C_i' = C_i + 2(H+G) + PRIO\_TRA + 2(H+G)(npriobits) + WIN\_PRIO + ETG \quad (9)$$

$$C_i'' = C_i' + TFCS \quad (10)$$

In general, queuing of a message can occur with jitter [13]. To provide a precise response time calculation, we follow the previous analysis by

Table 1. Timeout values.

Parameter	Value
$H+G$	$110\,\mu s$
TFCS	$300\,\mu s$
PRIO_TRA	$139\,\mu s$
WIN_PRIO	$235\,\mu s$
ETG	$555\,\mu s$

considering this release jitter. Accordingly the WCRT can be computed as follows:

$$R_i = \max_{q=0...Q_i-1}\left(w_{i,q} + J_i + C_i'' - q \times T_i\right) \quad (11)$$

where Q_i is given by:

$$Q_i = \left\lfloor \frac{L_i + J_i}{T_i} \right\rfloor + 1 \quad (12)$$

L_i is the length of level-*i* busy period and the key characteristic of a busy period is that all messages of priority *i* or higher queued strictly before the end of the busy period are transmitted during this period. Therefore, level-*i* busy period is the smallest value given by:

$$L_i = P_S + \sum_{j \in hp(i) \cup i} \left\lceil \frac{L_i + J_j}{T_j} \right\rceil \times P_S \quad (13)$$

where $hp(i)$ is similarly the set of message streams with priority higher than *i*. The first term of the Equation (13) refers to the blocking time that may be caused by a lower priority message stream and is equal to one slot. The duration of the slot is given by P_S that is the periodicity in which the synchronization signal is broadcast through the network. This period should be chosen in a way that a message with the longest transmission time (C_i) could be able to finish its transmission before the start of next synchronization signal. This constraint is formulated as follows:

$$P_S \geq TFCS + PRIO\_TRA + 2(H+G)(npriobits+1) + ETG + WIN\_PRIO + \max(C_i) \quad (14)$$

Finally,the longest time from the start of the busy period to the time in which instance *q* begins transmission successfully is given by:

$$w_{i,q} = q \times P_s + P_s + \sum_{j \in hp(i)} \left(\left\lceil \frac{w_{i,q} + J_j + Q_{bit}}{T_j} \right\rceil \right) \times P_s . \quad (15)$$

4.2. Analytical results

Now we apply the response time analysis to calculate the upper-bound of response time for each message streams according to Equation (9)-(15). First, it is necessary to select the timeout parameters according to their constraint or by measuring them in the hardware. We assume that the packet length (including PHY, MAC headers, CRC and payload) is

Table 2. Calculated response time for scenario "1" (6 nodes).

i	1	2	3	4	5	6
$T_i(\mu s)$	30,000	80,000	150,000	300,000	700,000	1,800,000
$C_i(\mu s)$	4,096	4,096	4,096	4,096	4,096	4,096
$C_i'(\mu s)$	8,545	8,545	8,545	8,545	8,545	8,545
$C_i''(\mu s)$	8,845	8,845	8,845	8,845	8,845	8,845
$R_i(\mu s)$	18,405	27,965	37,525	56,645	66,205	85,325

Table 3. Calculated response time for scenario "2" (10 nodes).

i	1	2	3	4	5
$T_i(\mu s)$	30,000	70,000	120,000	300,000	900,000
$C_i(\mu s)$	4,096	4,096	4,096	4,096	4,096
$C_i'(\mu s)$	8,545	8,545	8,545	8,545	8,545
$C_i''(\mu s)$	8,845	8,845	8,845	8,845	8,845
$R_i(\mu s)$	18,405	27,695	37,525	56,645	66,205
i	6	7	8	9	10
$T_i(\mu s)$	1,900,000	3,700,000	5,400,000	5,400,000	5,400,000
$C_i(\mu s)$	4,096	4,096	4,096	4,096	4,096
$C_i'(\mu s)$	8,545	8,545	8,545	8,545	8,545
$C_i''(\mu s)$	8,845	8,845	8,845	8,845	8,845
$R_i(\mu s)$	94,885	114,005	123,565	171,365	180,925

128 bytes since it is the maximum packet length supported by CC2420 radio [8]. Considering data rate of 250 Kb/s the time needed to transmit a packet is:

$$\forall i \in \{1...n\}: C_i = 128 \times 8 \times \frac{1}{250000} = 4096 \mu s \quad (16)$$

Applying Equation (9) and (10) we have:

$$\forall i \in \{1...n\}: C_i' = 8545 \mu s \quad (17)$$

$$\forall i \in \{1...n\}: C_i'' = 8845 \mu s \quad (18)$$

These two values are calculated with the timeouts given in Table 1 that is measured on a real platform. Considering *npriobits*=15 and all mentioned timeout values and considering the constraint (14), we choose the periodicity of the synchronization signal as:

$$P_S = 9560 \mu s \quad (19)$$

We explore two different scenarios; (i) scenario "1" including 6 nodes and (ii) scenario "2" including 10 nodes. Deadline monotonic priority assignment is used for priority assignment and we consider implicit deadline for message streams (i.e., D_i=T_i). The reason of having these two scenarios is to investigate the large difference between the smallest and greatest T_i, 180 times (scenario "2"), and also comparably smaller difference, 60 times (scenario "1"). In both of these scenarios we assume the release jitter is $1ms$, that is, $\forall i \in \{1...n\}: J_i = 1ms$. Table 2 and Table 3 show the calculated response time

Table 4. Experimentally measured response time for scenario "1" (6 nodes).

i	1	2	3	4	5	6
$T_i(\mu s)$	30,000	80,000	150,000	300,000	700,000	1,800,000
$R_i(\mu s)$	18,405	27,965	37,525	56,645	66,205	85,325
$R_i'(\mu s)$	18,348	27,583	37,128	55,982	59,184	64,834
Deadline miss ratio	0%	0%	0%	0%	0%	0%

Table 5. Experimentally measured response time for scenario "2" (10 nodes).

i	1	2	3	4	5
$T_i(\mu s)$	30,000	70,000	120,000	300,000	900,000
$R_i(\mu s)$	18,405	27,965	37,525	56,645	66,205
$R_i'(\mu s)$	18,343	27,584	37,147	56,225	55,428
Deadline miss ratio	0%	0%	0%	0%	0%
i	6	7	8	9	10
$T_i(\mu s)$	1,900,000	3,700,000	5,400,000	5,400,000	5,400,000
$R_i(\mu s)$	94,885	114,005	123,565	171,365	180,925
$R_i'(\mu s)$	58,019	39,509	62,490	34,464	62,403
Deadline miss ratio	0%	0%	0%	0%	0%

of first and second scenario respectively by applying Equations (9)-(15) and using the given timeout values in Table 1 (All the time values are given in μs and we separate each three digit with a comma to make numbers more legible). It can be easily observed that in both scenarios the calculated WCRT of all message streams are smaller than their relative deadlines (i.e., $\forall i \in \{1..n\} : R_i \le T_i$).

5. Experimental evaluation

In order to validate the calculated upper-bound, we have implemented the same scenarios mentioned in the previous section using a real-world platform. In the next section, we first explain the experimental setup followed by the results achieved.

5.1. Experimental Setup

In the experimental test-bed we used MicaZ motes [9] (featuring an Atmel ATmega128L 8-bit microcontroller with 128 kB of in-system programmable memory) equipped with the WiFLEX add-on board — see Figure 3(e). Each sensor node runs WiDom protocol implemented on Nano-RK [14] operating system. Nano-RK is a real-time operating system (RTOS) designed for wireless sensor networks that supports multi-hop networking. Nano-RK employs a novel energy-efficient time management scheme using one-shot timer interrupts instead of polling interrupt. Utilizing one-shot timer interrupts policy, the next timing interrupt is triggered when either a task is scheduled to be awakened because of an event or because it becomes eligible for scheduling. The Nano-RK timer tick is approximately 1*ms* so any time event scheduled in the future will be rounded to the nearest timer tick. Using this insight, one can infer that all the time related events such as task periods and event wake-ups will experience at most 1*ms* of jitter with respect to their target time and that is the reason of considering 1*ms* of jitter in the previous section. Despite the release jitter, it should be mentioned that this jitter does not affect our delay measurement since we used a hardware-timer with time resolution of 1μs, so, all the measured values have the precision of 1μs.

Each node has a task, (*Send-Task*), running on Nano-RK which is set to be requested periodically. By each request we increase a variable that is called *generated-packet* by one. Then nodes contend for the channel for sending the packets. They use their given ID numbers as their packet priority. If a node succeeds to send its packet by its deadline then another variable named *transmitted-packet* is incremented by one. Those packets that do not have the chance to be transmitted by their deadlines will be dropped. At the end of the experiment, if the value of *generated-packet* is not equal to the value of *transmitted-packet* then it implies that a deadline miss would occur. To measure the response time, each node counts the time from when the *Send-Task* is requested until that packet is actually transmitted. Then this waiting time (W_i) piggybacks on the packet payload. Upon receiving packet, the receiver extracts the waiting time and calculates the response time according to the following:

$$R_i = W_i + C_i \tag{20}$$

We have run two experiments as explained in Section 4.3. The first experiment includes 6 nodes aligned in a row and the receiver located 1m away from them, while the second scenario has 10 nodes placed in a circle with the radius of 1m and the receiver located at the centre [15]. Every node had one message stream and the experiments were run for 12,000 number of requests for message transmission.

5.2. Experimental results

Table 4 and 5 show the results achieved from the experimentally setup of described scenarios. The maximum value of response time obtained through the experiment is denoted by (R_i'). Analytically calculated response time (R_i) is shown again for the sake of convenience in comparison. It can be observed that the calculated response time is always greater than the maximum measured value, besides there is no deadline miss in the system which simply means that our

analysis provides a valid upper bound for any given message stream. There is not a great deal of difference between the measured and calculated response time for the message streams with smaller T_i while it can be considerable for those message streams with greater T_i. One possible reason is that the message streams with greater T_i transmit fewer packets throughout the experiment so they hardly experience those instances close to the worst-case scenario.

To check the ratio of deadline miss occurrence we insert the value of *generated-packet* (*Gnt_Pkt*) and *transmitted-packet* (*Tx_Pkt*) in the payload and then compute the deadline miss ratio as below:

$$Deadline\_miss\_ratio = \frac{(Gnt\_Pkt) - (Tx\_Pkt)}{(Gnt\_Pkt)} \times 100 \quad (21)$$

Recalling from the previous subsection, it is worth to noted again that the value of *transmitted-packet* (*Tx_Pkt*) increments only if the message has been sent within its deadline. After receiving each packet, the receiver increases the number of *received-packet* (*Rx_Pkt*) variable related to the sender by one. Doing so, it is possible to calculate the packet loss rate for any node by utilizing the following equation:

$$Packet\_loss\_rate = \frac{(Tx\_Pkt) - (Rx\_Pkt)}{(Tx\_Pkt)} \times 100 \quad (22)$$

To find out the total packet loss rate, one can simply add the values of *received-packet* in each node and calculate the total number of received packet, then apply the same policy for calculating total number of transmitted packet and then use the Equation (22). In both experiments we had a very small packet loss rate (less than 1%). The cause for this is that WiDom is a collision-free MAC protocol.

6. Conclusion

In this paper we focused on a recent prioritized MAC protocol, WiDom and opted for a more recent low overhead implementation of this MAC design that is called slotted WiDom. In this implementation a special node, master node, is responsible for synchronization process. It broadcasts periodically a signal on a separate channel in order to provide an accurate synchronization. Each node in slotted WiDom is equipped with an extended hardware, WiFLEX board, to obtain two main goals of (i) supporting a reliable contention in the tournament phase and (ii) achieving more efficient synchronization. We have developed the schedulability analysis of slotted WiDom by considering release jitter and presented an upper bound on the queuing time of a given message stream. To validate the calculated upper bound two different experiments were conducted by using real-world platform. The experimental result certifies our findings in the analytical calculations and by offering more than 99% successful transmission; it asserts the property of reliable prioritized collision-free characteristic for the slotted WiDom protocol.

References

[1] I. 11898:1993, “Road vehicles -- Interchange of digital information -- Controller Area Network (CAN) for high-speed communication,” ed, 1993.

[2] N. Pereira, B. Andersson and E. Tovar, “WiDom: a dominance protocol for wireless medium access,” Industrial Informatics, IEEE Transactions on, vol. 3, pp. 120-130, 2007.

[3] N. Pereira, R. Gomes, B. Andersson and E. Tovar, “Efficient aggregate computations in large-scale dense WSN,” in Real-Time and Embedded Technology and Applications Symposiumin, 2009, 15th IEEE International, pp. 317-326, 2009.

[4] K. Tindell, H. Hansson and A. J. Wellings, “Analysing real-time communications: controller area network (CAN),” in Real-Time Systems Symposium, pp. 259-263, 1994.

[5] M. Joseph and P. Pandya, “Finding response times in a real-time system,” Computer journal, vol. 29, pp. 390-395, 1986.

[6] R. Davis, A. Burns, R. Bril and J. Lukkien, “Controller Area Network (CAN) schedulability analysis: refuted, revisited and revised,” Real-Time Systems, vol. 35, pp. 239-272, 2007.

[7] A. Mok and S. Ward, “Distributed broadcast channel access,” Computer Networks, vol. 3, pp. 327–335, 1979.

[8] T. Instruments, “CC2420 datasheet,” http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

[9] CROSSBOW-Datasheet: MICAz. San Jose, USA. Crossbow Technology, Inc., 2004.

[10] R. Mangharam, A. Rowe and R. Rajkumar, “FireFly: a cross-layer platform for real-time embedded wireless networks,” Real-Time Systems, vol. 37, pp. 183-231, 2007.

[11] RF MONOLITHICS INC., “ASH Transceiver Designer’s Guide”, Dallas, Texas. RF Monolithics, Inc., 19 May 2004.

[12] B. NELSON, “Simple Clock & Data Recovery”, Dallas, Texas. RF Monolithics, Inc., 2 February 2003.

[13] N. Audsley, A. Burns, M. Richardson, K. Tindell and A. J. Wellings, “Applying new scheduling theory to static priority pre-emptive scheduling,” Software Engineering Journal, vol. 8, pp. 284-292, 1993.

[14] A. Eswaran, A. Rowe and R. Rajkumar, “Nano-RK: an Energy-aware Resource-centric RTOS for Sensor Networks”, IEEE Real-Time Systems Symposium, pp.256-265, 2005.

Appendix A

The experiments were conducted in an office environment where the noise floor was approximately -94dBm. Like most common office areas there were some Wi-Fi access points in the room; these access points are the main source of noise for our test bed. In order to have a precise measurement of the response time and for excluding communication related problem

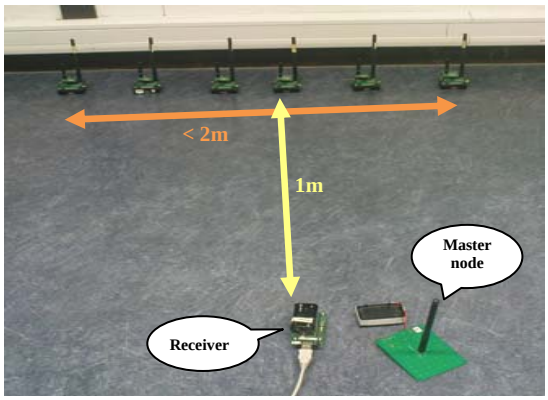


Figure 7. 6-node deployment.

(e.g. unreliability of the link that may happen due to unknown radio signal propagation pattern), we placed nodes close enough (less than 2.5 meters away) so that a robust and reliable communication could be achieved. Figure 7 shows the node placement in first scenario including 6 nodes.All nodes are placed on the floor and equipped with 2 AA batteries. Only the receiver is connected by an USB connector to the PC to log data.

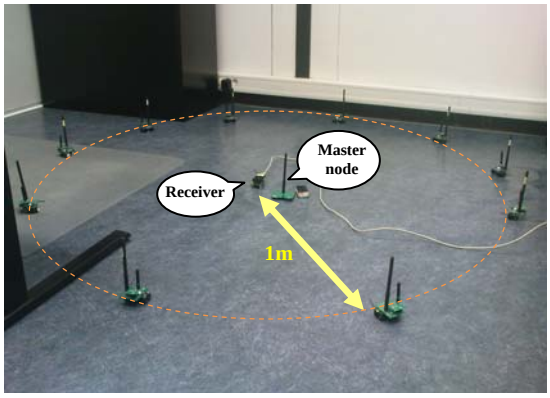


Figure 8.10-node deployment.

For the second scenario we used 10 nodes as illustrated in Figure 8. Nodes are placed in a circle and the receiver was located at the center of the circle. Radio module on the WiFLEX_main board works on the 418MHz frequency and transmits signal with 0dBm power. To assure the reception of synchronization signal by all nodes, the radio transmitter of the master node is chosen to broadcast the signal on lower frequency (315MHz) with the power of 1.5dBm. Both experiments were run for 12,000 transmission request over all computer nodes.

Appendix B

Table 4 and Table 5 in Subsection 5.2 show the experimental results for implicit-deadline message streams (i.e., $\forall i \in \{1..n\}$: $D_i=T_i$). However, the main advantage of WiDom is the ability of scheduling constrained-deadline message streams (i.e., $\forall i \in \{1..n\}$: $D_i<=T_i$).

In order to show this virtue of WiDom protocol, we have run another experiment including 10 nodes. All nodes had the same period as described in the second scenario and there is at most 1*ms* of release jitter. Nodes were located in a circle shown in Figure 8 and the experiment was carried out for 12,000 transmission requests. The main difference in this experiment is that there is an explicit deadline (D_i) related to each message stream in which all messages should reach the receiver before their deadline.

Table 6. Experimentally measured response time for constrained-deadline message streams.

i	1	2	3	4	5
$T_i(\mu s)$	30,000	70,000	120,000	300,000	900,000
$D_i(\mu s)$	20,000	50,000	100,000	100,000	100,000
$R_i'(\mu s)$	18,345	27,596	37,076	51,754	65,614
Dead miss probability	0%	0%	0%	0%	0%
i	6	7	8	9	10
$T_i(\mu s)$	1,900,000	3,700,000	5,400,000	5,400,000	5,400,000
$D_i(\mu s)$	150,000	150,000	200,000	200,000	200,000
$R_i'(\mu s)$	65,151	60,569	61,528	73,117	51,298
Dead miss probability	0%	0%	0%	0%	0%

Table 6 shows the result for the constrained-deadline message streams. As it is shown all message streams have been transmitted before their deadlines and there is no deadline miss occurrence. It should be noted that there are some message streams that have deadlines (D_i) much smaller than their periods (T_i). Most of the MAC protocols proposed in the literature are not able to accommodate such message streams. In TDMA-based MAC protocols all nodes should be triggered according to the smallest available deadline which is inefficient, while contention-based MAC protocols could not guarantee the reception of packets before their constrained deadlines. This experiment reveals the merit of utilizing WiDom protocol for accommodating constrained-deadline message streams.