The Utilization Bound of Non-Preemptive Rate-Monotonic Scheduling in Controller Area Networks is 25%

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

***Abstract*—Consider a distributed computer system comprising many computer nodes, each interconnected with a Controller Area Network (CAN) bus. We prove that if priorities to message streams are assigned using rate-monotonic (RM) and if the requested capacity of the CAN bus does not exceed 25% then all deadlines are met.**

I. Introduction

The Controller Area Network (CAN) bus [1] is one of the most widely used communication technologies in embedded/real-time systems as witnessed by the fact that 400 million CAN-enabled microcontrollers are manufactured each year [2]. The CAN bus offers three appealing features (i) CAN controllers are available at low cost and can use low-cost cabling, (ii) CAN networks are flexible in the sense that new computer nodes can be connected without changing the configuration of existing computer nodes, and (iii) the medium access control protocol used in the CAN bus implements non-preemptive static-priority scheduling [3], [4]. The latter implies that efficient real-time scheduling techniques (such as rate-monotonic (RM)) can be used and they can be analyzed using real-time scheduling theory for systems using non-preemptive scheduling [2], [3]. This scheduling theory is particularly useful for so-called *sporadic message streams*, that is, message streams where the exact time of an individual messages transmission request is unknown, but the minimum inter-arrival time between messages transmission requests from the same message stream is known.

A common way to characterize the performance of scheduling algorithms is to use the notion of a *utilization bound.* The utilization bound of a real-time scheduling algorithm A is the maximum number such that if a resource is scheduled by algorithm A and if the requested utilization of the resource does not exceed the utilization bound of A then all deadlines are met. Because of this property it gives designers an intuitive idea of the expected performance of a real-time scheduling algorithm.

Knowing the utilization bound of a real-time scheduling algorithm is important. A designer using a real-time scheduling algorithm with a non-zero utilization bound can rest assured that all deadlines can be met by using a sufficiently high speed of processors/networks (assuming that such components are available). A designer using a real-time scheduling algorithm with a utilization bound of zero may not know whether a higher speed will be sufficient.

Utilization bounds are well-established in the real-time scheduling literature. For example, on a single processor, the utilization bound of rate-monotonic (RM) is 69% [5] and on a multiprocessor a utilization bound of 33% [6] has been achieved with static-priority scheduling. The role of the utilization bound in communication systems is recently recognized [7] with links scheduled by weighted round-robin scheduling. Unfortunately, the utilization bound is currently unknown for CAN.

In this paper, we prove the utilization bound for non-preemptive RM scheduling on CAN. The utilization bound for CAN2.0A is 47/182. This is approximately 25%. The utilization bound for CAN2.0B is 67/227. This is approximately 29%. These non-zero utilization bounds of a system with non-preemptive scheduling are possible thanks to the fact that the CAN standard requires that the data payload of a message be at most 8 bytes and consequently only messages that comply with that requirement need to be considered.

The remainder of this paper is organized as follows. Section II gives a more precise statement of our assumptions, the terminology used and the problem statement. Section III presents a method that decides if and only if a set of message streams meets deadlines on a CAN bus. It also discusses the impact of non-preemptiveness on the utilization bound. Section IV presents a new result: the proof of the utilization bound of non-preemptive scheduling with restrictions on message transmission times. This result is applied in Section V which presents our main result: the utilization bound of CAN. Section VI studies the average-case performance. Section VII presents previous work on analysis of CAN and its context. Section VIII gives conclusions and future work.

II. Assumptions, Terminology and Problem statement

Consider a distributed computer system comprising computer nodes, each interconnected with a Controller Area Network (CAN) bus. The workload is described by a set of n message streams. This set is partitioned into subsets; one subset for each computer node. Each subset is assigned to a computer node and consequently, each message stream is

assigned to exactly one computer node. It is permitted for many message streams to be assigned to a single computer node. We do not make any assumptions on the assignment of message streams to computer nodes.

A message stream τ_i generates a (potentially infinite) sequence of messages. The time when these messages arrive cannot be controlled by the CAN controllers. It is assumed that the time between two consecutive arrivals of messages from the same message stream τ_i is at least T_i. A message from message stream τ_i has a message transmission time C_i; this includes the frame header and the arbitration for the CAN bus. R_i denotes the response time of message stream τ_i. R_i is defined as the minimum number such that for every message transmission from τ_i, it holds that the time from the message transmission request until the message of that transmission request has finished transmission is at most R_i time units. If $R_i \leq T_i$ then we say that the message stream τ_i meets its deadlines; otherwise it misses its deadline. It is assumed that $0 < C_i \leq T_i$, and that T_i and C_i are multiples of QUANTUM where QUANTUM denotes the time duration of a single bit in the CAN bus. (Many previous works on schedulability analysis of CAN used the symbol τ_{bit} to denote the duration of single bit. We do not use that notation because, as follows from the Liu-and-Layland notation, we use τ_i to denote a message stream.)

We assume that a message stream is assigned a priority, an integer. This integer is unique, that is, if τ_i has priority x then there is no other message stream τ_j with priority x. We assume that message streams are assigned priorities according to rate-monotonic (RM) with the identifier used as a tie-breaker, that is:

$$(T_i < T_j) \vee ((T_i = T_j) \wedge (i < j)) \\ \Rightarrow \tau_i \text{ has higher priority than } \tau_j$$

When we say that τ_i has higher priority than τ_j then it means that τ_i has lower priority number than τ_j. We assume that a message has the same priority as the priority of the message stream it belongs to. For convenience we use the following notations:

$$\text{hep}(i) = \{j : \tau_j \text{ has higher priority than } \tau_i \\ \text{or } \tau_j \text{ has equal priority to } \tau_i\}$$

and

$$\text{hp}(i) = \{j : \tau_j \text{ has higher priority than } \tau_i\}$$

and

$$\text{lp}(i) = \{j : \tau_j \text{ has lower priority than } \tau_i\}$$

We assume that all outstanding message transmission requests on a computer node are sorted in a priority queue and the highest-priority message transmission request competes with the highest-priority transmission request on the other nodes. The message transmission request with the highest priority "wins" and it sends its message on the CAN bus. All these assumptions are typical to CAN and its intended use in industrial systems. The utilization of a set of message streams is defined as:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

We address the problem of finding the utilization bound of CAN using RM. The utilization bound of CAN using RM is the maximum number such that if a set of message streams has a utilization that does not exceed the utilization bound of CAN and if messages are scheduled by CAN and if priorities are assigned according to RM then all deadlines are met. In order to solve this problem, it is helpful to understand non-preemptive RM scheduling.

III. Background on Non-Preemptive Rate-Monotonic Scheduling

The CAN bus uses non-preemptive static-priority scheduling and consequently the scheduling theory for non-preemptive static-priority scheduling can be used to compute the response time R_i.

It is known from previous research [2], [3] that R_i can be correctly computed by the following steps. B_i denotes the time that a message may need to wait for a lower-priority message to finish its transmissions. B_i is computed as:

$$B_i = \max_{k \in \text{lp}(i)} C_i \tag{1}$$

The method for computing R_i explores all messages released from message stream τ_i during an interval whose length is t_i, computed as follows:

$$t_i^0 = C_i \tag{2}$$

and iterate according to:

$$t_i^{k+1} = B_i + \sum_{\forall j \in \text{hep}(i)} \lceil \frac{t_i^k}{T_j} \rceil \cdot C_j \tag{3}$$

When Equation 3 converges with $t_i^{k+1} = t_i^k$ then this is the value of t_i.

We can now compute $w_i(q)$, the queuing time of the q^{th} message in the interval of duration t_i. It is computed iteratively until we obtain convergence, $w_i^{k+1}(q) = w_i^k(q)$ for the following iterative procedure:

$$w_i^0(q) = B_i + q \cdot C_i \tag{4}$$

and iterate according to:

$$w_i^{k+1}(q) = B_i + q \cdot C_i + \\ \sum_{\forall j \in \text{hep}(i)} \lceil \frac{w_i^k(q) + QUANTUM}{T_j} \rceil \cdot C_j \tag{5}$$

when Equation 5 converges with $w_i^k(q) = w_i^{k+1}(q)$ then this is the value of $w_i(q)$. We compute the response-time for the q^{th} arrival in the priority level-i busy period as:

$$R_i(q) = w_i(q) - q \cdot T_i + C_i \tag{6}$$

This is used to calculate the response time:

$$R_i = \max_{q=0..Q_i} R_i(q) \tag{7}$$

where Q_i is defined as:

$$Q_i = \lceil \frac{t_i}{T_i} \rceil$$

This analysis works for the case that

$$\sum_{i=1}^{n} \frac{C_i}{T_i} < 1 \tag{8}$$

From the definition of R_i it clearly holds that: If and only if τ_i: $R_i \leq T_i$ then all deadlines are met.

This exact schedulability analysis for non-preemptive scheduling is useful but unfortunately it does not offer a utilization bound. In fact, it is known [8] that all non-preemptive scheduling algorithms have a utilization bound zero. Since this result is important for our purpose, studying utilization bound, we formalize it in Example 1.

Example 1: Consider two message streams to be scheduled with non-preemptive RM. Let message stream τ_1 be characterized as $T_1 = \epsilon$ and $C_1 = 2\epsilon^2$. Let message stream τ_2 be characterized as $T_2 = 1$ and $C_2 = 2\epsilon$. We choose $\epsilon < 1/4$. We let O_i denote the time when τ_i arrives for the first time. The scheduling algorithm is not permitted to choose O_i. In order to make the discussion as general as possible, we consider both the case where the scheduling algorithm is not allowed to insert idle time and the case where the scheduling algorithm is allowed to insert idle time.

Let us consider an arbitrary transmission request by τ_2; Figure 1 illustrates this. Let A_2 denote the arrival time of that transmission request and let s_2 denote the start time of the transmission of the message of that transmission request. Because the transmission time of τ_2 is twice as long as T_1 we know that every time τ_2 transmits a message, it is possible (according to the sporadic model) that τ_1 will perform a message transmission request such that $s_2 < A_1 < A_1 + T_1 < s_2 + C_2$, and hence there is a time when both τ_1 and τ_2 must transmit simultaneously in order to meet deadlines. Hence a deadline is missed. Here, $U = 4\epsilon$. Choosing $\epsilon \to 0$ yields that the utilization bound is zero. This example shows that for every non-preemptive scheduling algorithm, the utilization bound is zero.

Based on Example 1, it would be tempting to believe that the utilization bound of CAN is zero. It can be seen however that in this example, the transmission time of one message stream is infinitely larger than the transmission time of another message stream. The CAN standard stipulates bounds on the message transmission times; these are a consequence of the bounds on the number of bytes in the data payload allowed by the CAN standard. As a result, the behavior as illustrated in Example 1 cannot happen. Example 2 shows a scenario which (as we will see later) is the scenario with the lowest utilization such CAN using RM misses a deadline.

Example 2: Consider two message streams to be scheduled with non-preemptive RM. Let message stream τ_1 be characterized as $T_1 = 182$ and $C_1 = 47$. Let message stream τ_2 be characterized as $T_2 = L$ and $C_2 = 136$. L is a positive integer. We let O_i denote the time when τ_i arrives for the first time. The scheduling algorithm is not permitted to choose O_i.

Let us consider an arbitrary transmission request by τ_2.; Figure 2 shows this. Let A_2 denote the arrival time of that transmission request and let s_2 denote the start time of the transmission of that transmission request. It can be seen that τ_1 misses a deadline. Here, $U = 47/182 + 136/L$. Choosing $L \to \infty$ yields that the utilization is $47/182$ which is approximately 25%.

For this reason, we will (in Section IV) prove the utilization bound of non-preemptive RM scheduling with restrictions on message transmission times. And then (in Section V) we will apply that utilization bound by applying the values required by CAN.

IV. New Results on Non-Preemptive Rate-Monotonic Scheduling

Let us first prove two lemmas (Lemma 1 and Lemma 2) that are instrumental and then see a theorem (Theorem 1) which states that if the relationship between transmission times is known and the utilization does not exceed a certain bound then all deadlines are met.

Lemma 1: Consider a message stream τ_i. If it holds that all message streams in $\mathrm{hp}(i)$ meet their deadlines and it holds that for τ_i that

$$B_i + C_i + \sum_{\forall j \in \mathrm{hp}(i)} (3 \cdot \frac{C_j}{T_j} \cdot T_i) \leq T_i$$

then τ_i meets all its deadlines.

Proof: The proof is by contradiction. Let us assume that the lemma was false. Then it must be that:

$$B_i + C_i + \sum_{\forall j \in \mathrm{hp}(i)} (3 \cdot \frac{C_j}{T_j} \cdot T_i) \leq T_i \tag{9}$$

and a deadline of τ_i was missed.

Of all messages of τ_i that missed a deadline let M denote the message with the earliest deadline. And let t_1 denote the deadline of message M. Let t_0 denote the arrival time of M. We know that the time interval $[t_0, t_1)$ belongs to a priority level-i busy period (if this was not the case then message M would have been transmitted and met its deadline). We know that:

$$\text{During } [t_0, t_1), \text{ a message of lower priority than } M \text{ can transmit for at most } B_i \text{ time units, where } B_i \text{ is given by Equation 1.} \tag{10}$$

We also know that:

$$\text{During } [t_0, t_1), \text{ a message } M \text{ can transmit for at most } C_i \text{ time units} \tag{11}$$

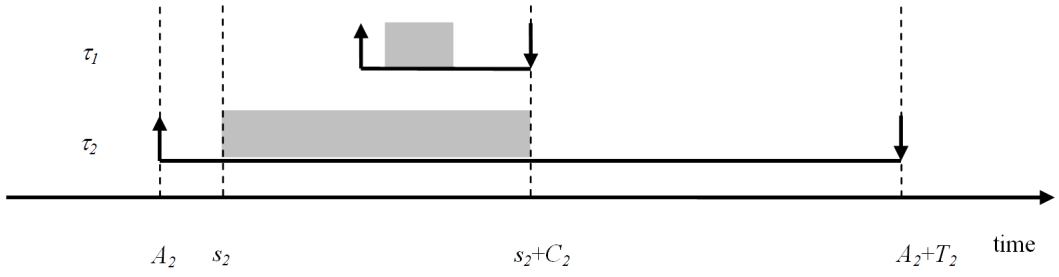

Fig. 1. All non-preemptive scheduling algorithms have the utilization bound 0.

Let us now study messages from a message stream τ_j, which has higher priority than τ_i. The time interval $[t_0,\ t_1)$ can be subdivided into $[t_0,\ t')$ and $[t',t'')$ and $[t'',\ t_1)$ such that t' is the earliest arrival time of τ_j in $[t_0,\ t_1)$ and t'' is the latest arrival time of τ_j in $[t_0,\ t_1)$. Based on this decomposition and using the knowledge that $T_j \leq T_i$, it is easy to see that:

$$\text{During } [t_0, t_1),\ \text{a message from } \tau_j \text{ can transmit}$$
$$\text{for at most } \lfloor \frac{T_i}{T_j} \rfloor \cdot C_j + 2 \cdot C_j \text{ time units} \quad (12)$$

It is easy to show (using our knowledge that $T_j \leq T_i$) that:

$$\lfloor \frac{T_i}{T_j} \rfloor \cdot C_j + 2 \cdot C_j \leq 3 \cdot \frac{C_j}{T_j} \cdot T_i \quad (13)$$

Combining Inequality 12 with Inequality 13 and applying it on all message streams with higher priority than τ_i yields:

$$\text{During } [t_0, t_1),\ \text{messages from hp}(i) \text{ can transmit}$$
$$\text{for at most } \sum_{j \in \text{hp}(i)} (3 \cdot \frac{C_j}{T_j} \cdot T_i) \text{ time units} \quad (14)$$

Combining Inequality 10, Inequality 11 and Inequality 14 yields:

$$\text{During } [t_0, t_1),\ \text{the channel is busy for at most}$$
$$B_i + C_i + \sum_{j \in \text{hp}(i)} (3 \cdot \frac{C_j}{T_j} \cdot T_i) \text{ time units} \quad (15)$$

In order to miss a deadline, it follows from (i) the fact that $[t_0,\ t_1)$ is a priority level-i busy period and (ii) the use of Inequality 15 that:

$$B_i + C_i + \sum_{\forall j \in \text{hp}(i)} (3 \cdot \frac{C_j}{T_j} \cdot T_i) > T_i \quad (16)$$

But this contradicts Inequality 9 and hence the lemma is true. ■

Lemma 2: If it holds for all τ_i that

$$B_i + C_i + \sum_{\forall j \in \text{hp}(i)} (3 \cdot \frac{C_j}{T_j} \cdot T_i) \leq T_i$$

then all deadlines of all message streams are met.

Proof: Follows by induction on i and using Lemma 1. ■

Theorem 1: Let x denote a real number such that $2 \leq x$. We claim that: If it holds that

$$\forall i, j : \frac{C_i}{C_j} \leq x$$

and if all message streams are scheduled with non-preemptive RM and if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq \frac{1}{x+1}$$

then all deadlines are met.

Proof: The proof is by contradiction. Let us assume that the theorem was false. Then it must be that there is an assignment to x such that $2 \leq x$ and there is a set of message streams such that:

$$\forall i, j : \frac{C_i}{C_j} \leq x \quad (17)$$

and message streams are scheduled with non-preemptive RM and

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq \frac{1}{x+1} \quad (18)$$

and a deadline was missed.

Since a deadline was missed it follows (from Lemma 2) that there is an i such that

$$B_i + C_i + \sum_{\forall j \in \text{hp}(i)} (3 \cdot \frac{C_j}{T_j} \cdot T_i) > T_i \quad (19)$$

Rewriting (19) yields:

$$1 < \frac{B_i}{T_i} + \frac{C_i}{T_i} + \sum_{\forall j \in \text{hp}(i)} (3 \cdot \frac{C_j}{T_j}) \quad (20)$$

Rewriting (18) yields:

$$(\sum_{\forall j \in \text{hp}(i)} \frac{C_j}{T_j}) + (\sum_{\forall j \in \{i\} \cup \text{lp}(i)} \frac{C_j}{T_j}) \leq \frac{1}{x+1} \quad (21)$$

Multiplying (21) by x+1 yields:

$$(\sum_{\forall j \in \text{hp}(i)} (x+1) \cdot \frac{C_j}{T_j}) + (\sum_{\forall j \in \{i\} \cup \text{lp}(i)} (x+1) \cdot \frac{C_j}{T_j}) \leq 1 \quad (22)$$

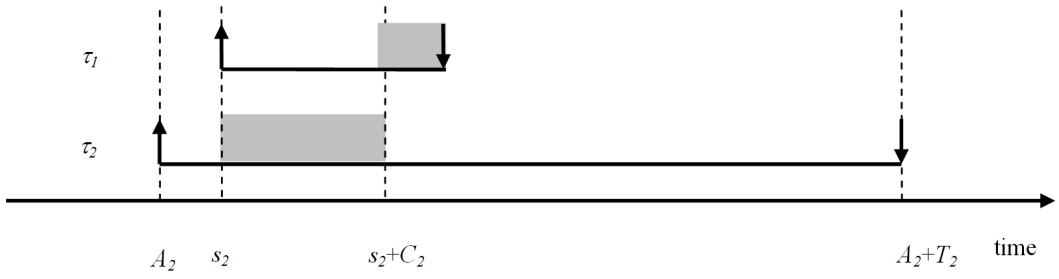

Fig. 2. An example that shows where a non-preemptive scheduling algorithm misses a deadline at a non-zero utilization.

Since $3 \leq x+1$ we have:

$$(\sum_{\forall j \in \mathrm{hp}(i)} 3 \cdot \frac{C_j}{T_j}) + (\sum_{\forall j \in \{i\} \cup \mathrm{lp}(i)} (x+1) \cdot \frac{C_j}{T_j}) \leq 1 \quad (23)$$

Combining (23) with (20) yields:

$$(\sum_{\forall j \in \mathrm{hp}(i)} 3 \cdot \frac{C_j}{T_j}) + (\sum_{\forall j \in \{i\} \cup \mathrm{lp}(i)} (x+1) \cdot \frac{C_j}{T_j}) < \frac{B_i}{T_i} + \frac{C_i}{T_i} + \sum_{\forall j \in \mathrm{hp}(i)} (3 \cdot \frac{C_j}{T_j}) \quad (24)$$

Simplifying (24) yields:

$$(\sum_{\forall j \in \{i\} \cup \mathrm{lp}(i)} (x+1) \cdot \frac{C_j}{T_j}) < \frac{B_i}{T_i} + \frac{C_i}{T_i} \quad (25)$$

We can rewrite (25) into:

$$(x+1) \cdot \frac{C_i}{T_i} + (\sum_{\forall j \in \mathrm{lp}(i)} (x+1) \cdot \frac{C_j}{T_j}) < \frac{B_i}{T_i} + \frac{C_i}{T_i} \quad (26)$$

Case 1. $\mathrm{lp}(i)$ is empty

Since $\mathrm{lp}(i)$ is empty, it clearly holds that $B_i = 0$. Using this on (26) yields:

$$(x+1) \cdot \frac{C_i}{T_i} < \frac{C_i}{T_i} \quad (27)$$

Multiplying (27) by T_i and using the knowledge that $2 \leq x$ yields:

$$3 \cdot \frac{C_i}{T_i} < \frac{C_i}{T_i} \quad (28)$$

It is easy to see that (28) is impossible. (End of Case 1.)

Case 2. $\mathrm{lp}(i)$ is empty

We know from (1) that B_i is obtained as:

$$B_i = \max_{k \in \mathrm{lp}(i)} C_k$$

Let k denote the index of the message stream with the maximum C_k among all indices in $\mathrm{lp}(i)$. Since a sum cannot be less than one of its terms it clearly holds that:

$$(x+1) \cdot \frac{C_k}{T_k} \leq (\sum_{\forall j \in \mathrm{lp}(i)} (x+1) \cdot \frac{C_j}{T_j}) \quad (29)$$

Applying (29) on (26) yields:

$$(x+1) \cdot \frac{C_i}{T_i} + (x+1) \cdot \frac{C_k}{T_k} < \frac{B_i}{T_i} + \frac{C_i}{T_i} \quad (30)$$

And using the knowledge that $B_i = C_k$ yields:

$$(x+1) \cdot \frac{C_i}{T_i} + (x+1) \cdot \frac{C_k}{T_k} < \frac{C_k}{T_i} + \frac{C_i}{T_i} \quad (31)$$

Multiplying with T_i yields:

$$(x+1) \cdot C_i + (x+1) \cdot \frac{C_k}{T_k} \cdot T_i < C_k + C_i \quad (32)$$

Clearly in general it holds that:

$$0 \leq (x+1) \cdot \frac{C_k}{T_k} \cdot T_i \quad (33)$$

Combining (33) and (32) yields:

$$(x+1) \cdot C_i < C_k + C_i \quad (34)$$

Rewriting (34) yields:

$$((x+1)-1) \cdot C_i < C_k \quad (35)$$

Finally, simplifying (35) yields:

$$x < \frac{C_k}{C_i} \quad (36)$$

But this contradicts (17). (End of Case 2.)

We can see that regardless of which case occurs we have a contradiction. Hence the theorem is correct. ■

V. The Utilization Bound of CAN

The CAN standard specifies that the data payload is at most 8 bytes and at least 0 bytes. Let S_i denote the number of bytes of payload of a packet and recall that QUANTUM denotes the time duration of a single bit. CAN uses a technique called *bit-stuffing* that ensures that not too many consecutive zeros or consecutive ones are on the bus. Based on this bit stuffing and using the notation, a well-known expression [2], [3] exists for computing the maximum transmission time and for convenience we state it below:

$$C_i = (g + 8 \cdot S_i + 13 + \lfloor \frac{g + 8 \cdot S_i - 1}{4} \rfloor) \cdot \mathrm{QUANTUM} \quad (37)$$

where g is the number of bits in the frame header that is subject to bit-stuffing. For CAN2.0A (which have 11-bit priorities) we have g=34 and for CAN2.0B (which have 29-bit priorities) we have $g = 54$ [2], [3]. Let CMAX denote the maximum transmission time and CMIN denote the minimum transmission time. By applying $S_i = 8$ on (37) we obtain:

$$CMAX = (g + 64 + 13 + \lfloor \frac{g + 64 - 1}{4} \rfloor) \cdot \text{QUANTUM} \quad (38)$$

By applying $S_i = 0$ on (37) and observing that the expression in the floor in (37) represents the extra time due to bit-stuffing, which may be zero, we obtain:

$$CMIN = (g + 13) \cdot \text{QUANTUM} \quad (39)$$

We will explore both CAN2.0A and CAN2.0B.

A. CAN2.0A

Applying $g = 34$ on (38) yields:

$$\begin{aligned} CMAX &= (34 + 64 + 13 + \lfloor \frac{34 + 64 - 1}{4} \rfloor) \cdot \text{QUANTUM} \\ &= 135 \cdot \text{QUANTUM} \end{aligned} \quad (40)$$

and applying $g = 34$ on (39) yields:

$$\begin{aligned} CMIN &= (g + 13) \cdot \text{QUANTUM} \\ &= 47 \cdot \text{QUANTUM} \end{aligned} \quad (41)$$

Applying these values gives us Theorem 2.

Theorem 2: If messages are scheduled with CAN2.0A and if messages comply with the length restrictions stipulated by CAN2.0A and if message streams are scheduled with non-preemptive RM and if

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq \frac{47}{182}$$

then all deadlines are met.

Proof: Follows from applying $x = 135/47$ in Theorem 1. ■

It is worth to observe that 47/182 is approximately 0.2582 and hence the utilization bound of CAN2.0A is 25%.

B. CAN2.0B

Applying $g = 54$ on (38) yields:

$$\begin{aligned} CMAX &= (56 + 64 + 13 + \lfloor \frac{56 + 64 - 1}{4} \rfloor) \cdot \text{QUANTUM} \\ &= 160 \cdot \text{QUANTUM} \end{aligned} \quad (42)$$

and applying $g = 54$ on (39) yields:

$$\begin{aligned} CMIN &= (g + 13) \cdot \text{QUANTUM} \\ &= 67 \cdot \text{QUANTUM} \end{aligned} \quad (43)$$

Applying these values gives us Theorem 3.

Theorem 3: If messages are scheduled with CAN2.0B and if messages comply with the length restrictions stipulated by CAN2.0B and if message streams are scheduled with non-preemptive RM and if

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq \frac{67}{227}$$

then all deadlines are met.

Proof: Follows from applying x=160/67 in Theorem 1. ■

It is worth to observe that 67/227 is approximately 0.2951 and hence the utilization bound of CAN2.0B is 29%.

VI. Average-case Performance

Having seen that the utilization bound of CAN is 25% we know that all message streams with a utilization of at most 25% which are complying to the CAN standard meets deadlines when scheduled by RM on CAN. This result does however not rule out the possibility that deadlines can be met at a utilization bound greater than 25%. It is therefore interesting to know if the CAN bus can be used at a higher bus load. Previous work [2] has hinted that much higher utilization can be used. In order find out the answer, we perform a simulation study.

In the simulation study, we consider CAN2.0A with a bitspeed of 1Mbit/s. We generate a set of message streams randomly. S_i is a uniformly distributed random variable from {1,2,...,8}, given in bytes. C_i is computed from S_i. The parameter T_i is a uniformly distributed random variable from {270,271,...,5000000} (T_i is given in μs). The number of message streams in a set is between two and 50.

We generate 7 million sets of message streams. For each set, we calculate the utilization and put it in one of 100 buckets. For example if a set of message streams has a utilization 0.6734 then it belongs to the bucket of sets with a utilization in the range [0.67,0.68). We count the number of sets in a bucket. For each set in the bucket we also decided (based on the exact schedulability analysis in Section IV) whether all messages meet deadlines. A set of message streams that meet deadlines is said to be successful. The success ratio of a bucket is said to be the number of message streams that are successful in a bucket divided by the number of message streams in the bucket.

Figure 3 shows the result. It can be seen that with up to 25% utilization, all message streams meet deadlines. This is expected considering Theorem 2, which states that the utilization bound is 25%. It can also be seen that indeed, even at a utilization as high as 80%, RM on a CAN bus can schedule most sets of message streams.

VII. Context and Previous work

In the 1970s, the principle of using bit-dominance was invented [9], [10]. Here, computer nodes go through a contention-phase which ends by electing a winner node. Each message has an integer, a priority, attached to it. Nodes contend for the medium by comparing their most significant bit; if a node has the most significant bit being "1" but another node has the most significant bit being "0" then it loses the

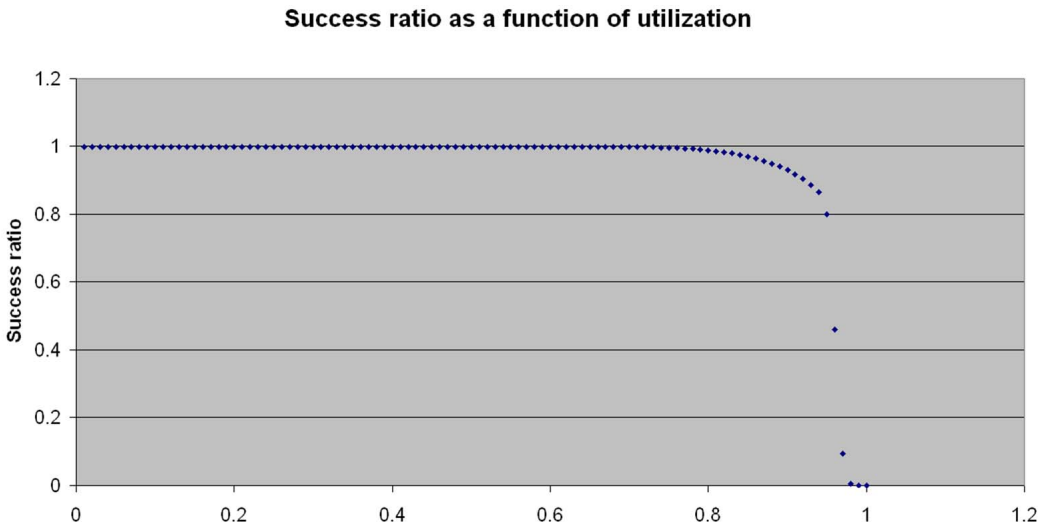

Fig. 3. Average-case performance of exact schedulability analysis of message streams scheduled on CAN using RM.

contention. The nodes that have not yet lost, continue the contention with their 2nd most significant bit, and so on. Finally, only one node is a winner, assuming that priorities are unique.

It is notable that early workers used the bit-dominance principle for parallel busses as well as serial buses [9]. Early research also explored the use of an entire packet as a ”bit”.

The bit-dominance principles became popularized through the introduction of the CAN bus in the mid 1980s, introduced by Bosch [1] for the purpose of replacing dedicated wires in vehicles with a shared communication bus. Since several nodes share the bus, the communication delay experienced by one node depends on the traffic generated by other nodes. Designers in the early 1990s therefore often used a rule of thumb: the CAN bus should be utilized to at most 30% [2].

Research in the mid 1990s discovered however that the CAN bus performs contention according to static-priority and hence the real-time scheduling theory for static-priority scheduling can be applied [4]. Consequently, a utilization of 80% of the CAN bus became common [2].

Given the commercial success of CAN, the scientific community has been eager to contribute with improvements. Proposals have been made on how to perform Earliest-Deadline-First scheduling on CAN [11]. And ideas on how to analyse response times in the presence of retransmissions due to communication faults [12] and the so-called babbling idiot problem [13]. The analysis of the CAN bus can be conveniently incorporated into the holistic analysis framework [14] which gives designers the capability of analysing end-to-end delays in distributed real-time systems (for example in a vehicle). In such setting, a large number of recurrence equations are set up and solving them can be time consuming and for this reason, speed-up techniques were introduced [15].

One common criticism of CAN is its low bit-rate originating from the fact that each node must ”see” a bit before transmitting another. This limitation is originating partly due to the limited speed of light but also because of constraints on resistance values in the bus. For this reason, a CAN bus cannot achieve a bit-rate greater than 1Mbit/s. It is currently unknown however whether an arbitration technique (different from CAN) but with a large number of priorities can be achieved while eliminating the limitation on the bit-speed.

VIII. Conclusions and Future work

We have proven that if priorities to message streams are assigned using rate-monotonic (RM) and if the requested capacity of the CAN bus does not exceed 25% then all deadlines are met. This bound is tight, that is, no greater bound can be proven for the CAN bus. This can be seen by considering Example 2 and let each time unit represent QUANTUM. We left open the question whether a non-zero utilization bound can be achieved for other communication technologies where the length of the data payload is bounded.

Acknowledgment

This work was partially funded by the Portuguese Science and Technology Foundation (Fundação para a Ciência e Tecnologia - FCT) and the European Comission through grant ArtistDesign ICT-NoE- 214373.

References

[1] *CAN Specification, ver. 2.0*, Bosch GmbH, Stuttgart, Germany, 1991.
[2] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, “Controller area network (CAN) schedulability analysis: Refuted, revisited and revised,” *Journal of Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.
[3] L. George, N. Rivierre, and M. Spuri, “Preemptive and non-preemptive real-time uniprocessor scheduling,” INRIA, Technical Report RR-2966, Tech. Rep., 1996.
[4] K. Tindell, H. Hansson, and A. Wellings, “Analysing real-time communications: Controller area network (CAN),” in *15th Real-Time Systems Symposium (RTSS’94)*, 1994, pp. 259–263.
[5] C. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” *Journal of the Association for Computing Machinery*, vol. 20, pp. 46–61, 1973.

[6] B. Andersson, S. Baruah, and J. Jonsson, "Static-priority scheduling on multiprocessors," in *22nd IEEE Real-Time Systems Symposium (RTSS'01)*, 2001, pp. 193–202.
[7] J. Wu, J.-C. Liu, and W. Zhao, "Utilization-bound based schedulability analysis of weighted round robin schedulers," in *28nd IEEE Real-Time Systems Symposium (RTSS'07)*, 2007, pp. 435–446.
[8] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of periodic and sporadic tasks," in *12nd IEEE Real-Time Systems Symposium (RTSS'91)*, 1991, pp. 129–139.
[9] A. K. Mok and S. Ward, "Distributed broadcast channel access," *Computer Networks*, vol. 3, pp. 327–335, 1979.
[10] L. Niesnevich and E. Strasbourger, "Decentralized priority in data communication," in *Second Internatial Symposium on Computer Architecture*, 1975, pp. 1–6.
[11] M. DiNatale, "Scheduling the CAN bus with earliest deadline techniques," in *21nd IEEE Real-Time Systems Symposium (RTSS'00)*, 2000, pp. 259–268.
[12] S. Punnekkat, H. Hansson, and C. Norström, "Response time analysis under errors for CAN," in *6th IEEE Real-Time Technology and Applications Symposium (RTAS'00)*, 2000, pp. 258–265.
[13] G. Buja, J. R. Pimentel, and A. Zuccollo, "Overcoming babbling-idiot failures in CAN networks: A simple and effective bus guardian solution for the FlexCAN architecture," *IEEE Transactions on Industrial Informatics*, vol. 3, pp. 225–233, 2007.
[14] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming*, vol. 40, pp. 117–134, 1994.
[15] M. Sjödin and H. Hansson, "Improved response-time analysis calculations," in *19th IEEE Real-Time Systems Symposium (RTAS'98)*, 1998, pp. 399–408.