

Schedulability Analysis for CAN-based Control Applications with Dynamic Bandwidth Management

Manel Velasco, Pau Martí, and Josep M. Fuertes

Automatic Control Department, Technical University of Catalonia
Barcelona, Spain
<http://dcs.upc.edu>

Research Report: ESAII-RR-08-04

November 2008

Abstract

This report presents the schedulability analysis for control messages when networked control loops built on top of the Controller Area Network (CAN) are dynamically allocated bandwidth in terms of their controlled plants' dynamics. The bandwidth allocation policy is theoretically described by an optimization problem and practically solved by the distributed bitwise arbitration of CAN messages when message identifiers, i.e., priorities, reflect control applications demands. This poses the problem of assessing whether the set of real-time messages will meet their deadlines regardless of run-time priority changes. This is solved by an schedulability analysis based on recent results on worst-case response time techniques for real-time CAN applications. The analysis ends up with the schedulability test for this type of applications.

Keywords: Networked control systems, Controller Area Network, schedulability analysis, dynamic bandwidth management, optimal sampling period selection.

Contents

| | | |
|----------|-------------------------------------------------------------------|-----------|
| 1 | Introduction | 4 |
| 2 | Background | 6 |
| 2.1 | Problem to be solved, solution and practical considerations . . . | 6 |
| 2.2 | Application profile | 7 |
| 3 | Validation and discussion | 9 |
| 4 | Schedulability Analysis: Preliminaries | 11 |
| 4.1 | Motivating example | 11 |
| 4.2 | CAN worst-case response time analysis revisited | 13 |
| 4.3 | Notation | 15 |
| 5 | Schedulability Analysis: Case Study | 15 |
| 5.1 | Case 0 | 16 |
| 5.2 | Case 1 | 16 |
| 5.3 | Case 2 and Case 3 | 18 |
| 5.4 | Case 4 | 19 |
| 6 | Schedulability Test | 20 |
| 6.1 | Extension | 21 |
| 7 | Conclusions | 22 |

1 Introduction

In competitive markets such as the automotive sector, the Controller Area Network (CAN) [1] is being widely used in real-time control applications. Its applicability covers many areas, including battery management, steer-by-wire systems or suspension control [2, 3, 4], and its properties and functionalities in terms of reliability, automatic node discovery, flexibility or fault tolerance are constantly being enhanced [5, 6, 7, 8, 9].

Competitiveness demands products, e.g. cars, providing more and better functionalities without rising costs. This implies that networked embedded applications have to include new capabilities or improve existing ones without increasing the available computing resources (processors, bandwidth, batteries, etc).

For control applications where several control loops share limited computing resources, the problem of efficiently managing these resources with respect to control performance has been recently achieved by resource allocation (RA) policies. Most of the existing work on RA concentrates on performance optimization in processor-based embedded control systems [11, 12, 13, 14, 16, 15, 17, 18, 19, 20]. Many of them are based on specifying and solving an optimization problem that relates control performance and CPU utilization. Their solutions indicate that dynamic re-allocation of processor shares by means of readjusting tasks periods at run-time as a function of the controlled plants' dynamics is the best practice.

Looking at maximizing control performance under communication bandwidth constraints when several control loops are closed through a real-time network, i.e., networked control system (NCS, [21]), different approaches have been presented [22, 23, 24, 25, 26, 27, 28, 29]. Like in the case of the processor, and already pointed out in [30], run-time bandwidth re-allocation in terms of plants' dynamics is the best choice. However, in NCS, a key aspect has to be considered due to the inherent distributed architecture. When control tasks share a CPU, the real-time kernel can perform the run-time RA optimization procedure because the information required such as plants' current states can be easily accessed. That is, the information is centralized. However, in a distributed architecture such as for NCS, the information is physically distributed. As a consequence, the run-time optimization can not be performed straightforward.

The previous observation explains why many of the approaches trading off control performance and bandwidth for NCS provide off-line solutions which may be enhanced by heuristic run-time adaptations (e.g., [23], [25], or [27]). Other solutions, where run-time centralized optimization is carried

out, skip the implementation problems posed by the distributed architecture, such as in [24]. On the contrary, the run-time RA approaches presented in [22], [26], [28] and [29] take into account the distributed nature of NCS. They provide solutions taking into account stability guarantees [22], or maximization of control performance [26], [28], [29]. In particular, [29] presented a distributed resource allocation approach targeting control performance optimization in CAN-based control applications while [28] targets IP-like networks. The work by [26] does not target any specific networking technology but fails at specifying how each node can compute the available bandwidth, information required for applying its solution.

The approach presented by [29] is theoretically described by an optimization problem and practically solved by the distributed bitwise arbitration of CAN messages when message identifiers, i.e., priorities, reflect control applications demands. The solution, that applies to single-input/single-output networked control systems, requires using an application profile that mandates to specify types of nodes (sensors, controllers and actuators) and defines how messages identifiers should be specified with respect to the controlled plant dynamics, which ends up with the definition of four message classes.

To structure messages in classes is not a new approach in CAN. For example, it was done in the mixed traffic scheduling scheme [31] or more recently in the hybrid priority scheduling scheme [32]. In [31] message classes were defined to improve schedulable utilization while [29] targets to efficiently use the available schedulable utilization in order to improve control applications' performance. In [32] different messages classes were defined by hybrid priority schemes with the objective of improving control loops stability performances. Their approach can be considered as complementary to the profile defined in [29]. However, they do not explicitly address schedulability issues as we do.

The application of the profile defined by [29] provokes changes in message priorities because control demands evolve at run-time. This poses the problem of assessing whether the set of real-time messages will meet their deadlines regardless of run-time priority changes. The contribution of this report, which extends [29], is to present the schedulability analysis for these systems. The schedulability analysis is based on recent results on worst-case response time techniques for real-time CAN applications [33], but with the property that message are divided into classes, and that message priorities (within a class) can change at run-time. The analysis ends up with the schedulability test for this type of applications.

The rest of this paper is organized as follows. Section 2 summarizes pre-

vious results [29] in order to provide the appropriated background. Section 3 validates that the application profile achieves the desired functionality. Section 4 establishes the preliminaries for the schedulability analysis, which is developed in Section 5. Section 6 presents the schedulability test and extends its applicability. Section 7 concludes the paper.

2 Background

The networked architecture considered in this work is the following: each controller and plant constitute a control loop closed over the network, and each controller is assumed to be implemented in separate nodes. That is, sampling, control algorithm computation and actuation are performed in separated nodes. These nodes exchange sample and control signal information through the network shared by all nodes of all control loops. Therefore, network bandwidth is the scarce resource that must be allocated among the control loops.

2.1 Problem to be solved, solution and practical considerations

The problem to be solved is how to assign the scarce network bandwidth to the set of control loops such that all messages are schedulable and overall control performance is maximized, knowing that the controllers will provide better performance when given more network bandwidth. The key assumption is that all controllers can not run, simultaneously, at their highest possible sampling frequency due to bandwidth limitations, so they cannot provide the best possible control performance. In [29] it is shown that the problem to be solved can be formulated as a constrained optimization problem whose solution dictates that, after a minimum bandwidth share is guaranteed to each control loop, the remaining available bandwidth should be assigned to the control loop with largest error (possibly weighted), where the error is defined as a function of the plant state.

Therefore, the optimal solution requires to compute a function, *largest error*, that takes parameters, *errors*, from different nodes. This function has to be computed efficiently and with a small (and bounded) number of messages. In fact, since control systems are extremely sensitive to timing variations [21], it is required that the time spent on computing and executing the bandwidth reallocation solution should be negligible with respect the dynamics of the set of controlled plants. Therefore, it would be desirable to

have a solution with a time-complexity that does not depend on the number of nodes.

Depending on the type of communication protocol considered in the implementation, the previous requirement may not be met. In [29] it is discussed that the optimal policy could be achieved using centralized protocols. However, the primary limitation of these approaches is that they assume the existence of a master node that knows the current state of all controlled plants. Maintaining a global state can be problematic in practice, introducing a central point of failure, and increasing overall network traffic and latencies.

Without assuming that all the information required to solve the allocation problem is localized in a single node, a different solution is required. The key idea in [29] is to observe that CAN permits to schedule messages on a priority based semantics, and that a feasible implementation of the optimal bandwidth allocation policy could take advantage of this property. Intuitively, for the distributed implementation of the optimal solution, [29] proposes to encode plants' errors into each message identifier in order to have the solution implemented at the CAN bitwise arbitration resolution. Therefore, structural properties of CAN permit to efficiently implement the optimal policy because calculating the largest error of all plants reduces to resolve the arbitration and to observe its winner.

2.2 Application profile

The previous intuitive solution is achieved in [29] by defining an application profile for CAN extended frames (formerly known as CAN 2.0 B) but the extension to CAN base frames (formerly known as CAN 2.0 A) is straightforward.

The application profile allows the following functionalities. First, control signal messages are given the highest priority in order to be guaranteed and thus minimize latencies in each control loop operation. Also, sampler nodes have to be assigned a minimum bandwidth share (periodicity) to guarantee stability. Secondly, non-control messages behave as expected in terms of being granted bus access once control messages and periodic sampler messages have been guaranteed. Finally, the sampler node with highest error is assigned the bandwidth left over. The accomplishment of these functionalities is achieved by deploying the set of networked control loops using different type of nodes and messages classes.

Four types of nodes are defined:

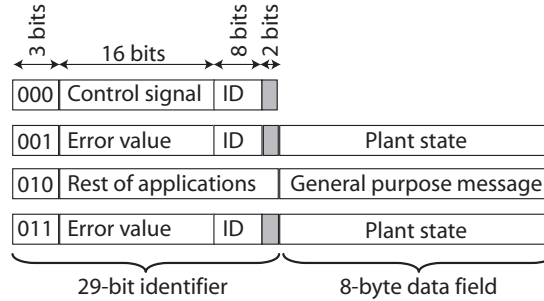


Figure 1: Bit encoding for message classes.

- **Sampler nodes:** they sample the plant at the maximum rate (given by the sensor hardware limitations), compute the plant error, and try to send this information (plant error and plant state) over the network.
- **Controller nodes:** each controller node, after receiving a sampler message, computes (using the plant state) and sends the control signal to the corresponding actuator.
- **Actuator nodes:** they apply the control signal to the plant upon reception of the control signal message.
- **Other nodes:** they perform other non-control activities.

Four messages classes are defined, which can be distinguished by the first 3 bits of the 29 bits message identifier as illustrated in Figure 1:

- **000** (control messages): This class contains the highest priority messages, which are always granted to win bus access. These messages will be used by controller nodes to send control signal messages to actuator nodes. They are only send upon reception of a sampler message.
- **001** (granted sensor messages): This class contains the second higher priority messages, which have to be also always granted in the bus. These messages, with a guaranteed minimum periodicity, will be used by sensor nodes to send plant states to controller nodes. Apart from the first 3 bits, the following 16 bits will encode the 1's complement of the plant error in such a way that the sample node with highest error

will always win access to the bus in front of the other sampler nodes in the case of collisions.

- **010** (general purpose messages): This class contains all non-control messages and their identifier encoding should be as thought originally, but with 010 in the first 3 bits.
- **011** (best effort sensor messages): This class contains the lowest priority messages, and they are not guaranteed. These messages will be used by all sampler nodes to send as many messages as possible. Apart from the first 3 bits, the following 16 bits will code the plant error like messages of class 001. By doing so, the bandwidth left over will be re-assigned at run-time to the node with highest error.

The four traffic classes are distinguished by the first 3 bits of the message identifier. Although 2 bits would suffice (as originally presented in [29]), practical considerations prevent using only 2 bits. CAN does not allow to have the most significant 7 bits recessive, i.e., 1. Therefore, if only 2 bits were used to code the class types, the last class would be distinguished by 11. This code together with the error value (see Figure 1) could result in 7 recessive bits. To avoid this potential problem, 3 bits are used.

3 Validation and discussion

Figure 2 illustrates the bit-wise arbitration of CAN applied to all possible combinations of collisions of messages of the application profile. As mandated by the application profile, a) control signal messages (class 000) will always win bus access in front of any message of the other classes, b) granted sampler messages (class 001) will always win bus access in front of general purpose messages (class 010) or best effort sampler messages (class 011), and c) best effort sampler messages will only be transmitted after all the other messages.

Figure 3 shows all possible precedence relations that can take place between messages of the four classes. Some precedence relations are mandatory whereas others are not possible or have a random relation.

After sampler messages (classes 001 and 011) will always be control signal messages (class 000). Here it is assumed that time spent by the controller node to both compute the control signal and place the message in a transmitter buffer is less than the inter-frame time (3 bit time), which

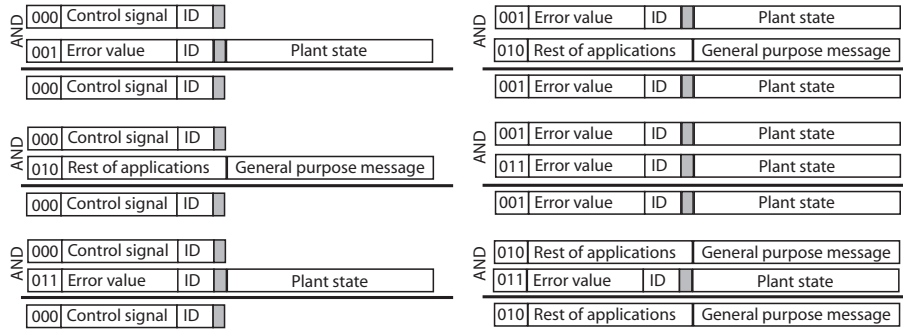


Figure 2: Message arbitration

is a realistic assumption taking into account standard CAN bit rates and current microprocessor's speed. In the case that this assumption can not hold, the precedence relations will be as shown in Figure 3 with the difference that after a sampler message, the bus will be occupied by any other class of message until the controller node places the control signal message in the transmitter buffer. The schedulability analysis presented next takes this assumption into account. However, later on, this assumption is removed to extend the analysis to all possible cases.

It is also interesting to point out that a control signal message (class 000) can not be followed by another control signal message because, as discussed in the description of message class 000, a control message is sent by a controller node upon reception of a sampler message. And only one controller node will respond to a sampler message.

Finally, a message of class 010 can not be followed by a control signal message because they can be only triggered by sampler messages.

By using this application profile, the bandwidth allocation policy mandated by the optimal policy is achieved without requiring a centralized bandwidth management.

Note that the number of closed-loop systems that the network may support will be limited by the bandwidth required by the granted messages (classes 000 and 001).

If the network bandwidth is equal to the bandwidth required by granted messages and non-control messages, control loops will behave correctly but no control performance optimization will take place. When not all the bandwidth is occupied (e.g., when non-control message bandwidth requirements decrease, or when the granted messages for all control loops can not occupy the bandwidth left by non-control messages), then, the control performance

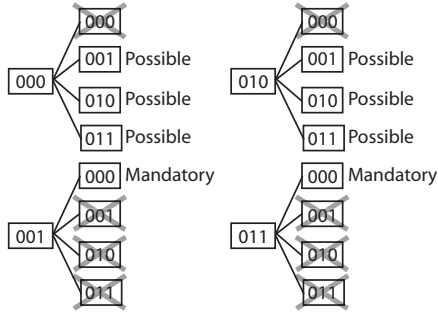


Figure 3: Precedence relations between messages

optimization will take place because messages of class 011 will start being transmitted, thus having more bandwidth for the control loop with highest error.

If the network bandwidth is less than the one required by granted messages and non-control messages, the overall system of course can not be implemented.

4 Schedulability Analysis: Preliminaries

This section motivates the need for the schedulability analysis, shortly reviews the latest results on worst-case response time analysis, and presents convenient notation.

4.1 Motivating example

Figure 4 shows an example of how the CAN traffic is affected when the application profile is applied. The top subfigure presents the traffic of CAN in standard operation. The bottom subfigure shows the traffic of CAN when using the profile. In both subfigures, general purpose traffic (dashed boxes, class 010) share the network with those messages required in the operation of two control loops closed over the network. For each control loop, black boxes represent control messages (class 000), white boxes represent granted sensor messages (class 001) and gray boxes represent best effort sensor messages (class 011) that only appear when the profile is used (bottom subfigure).

In the top subfigure, pairs of white and black boxes in the bottom correspond to the sampling and control messages of one control loop. Similarly,

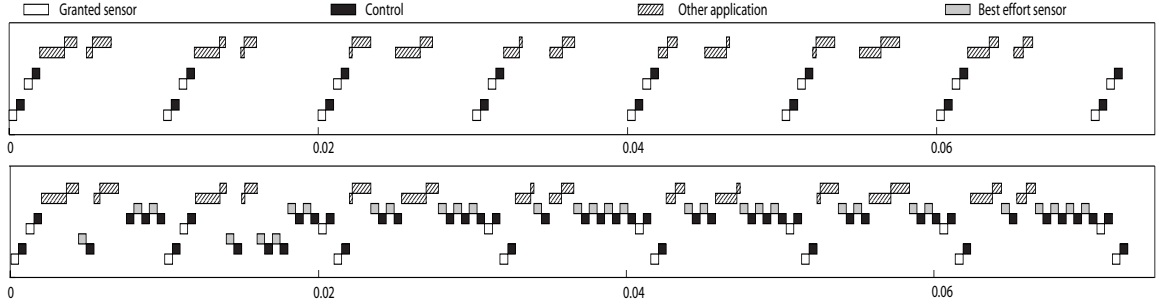


Figure 4: CAN traffic with (bottom subfigure) or without (top subfigure) the enhancement provided by the application profile.

pairs of white and black boxes in the middle of the top subfigure correspond to the sampling and control messages of the second control loop. As it can be seen, they are periodically triggered, and they have the same periodicity. When using the application profile, bottom subfigure, these pairs still appear. But in addition, more control messages (black boxes) are sent because they are triggered by best effort sensor messages (gray boxes). In fact, the new pairs of gray and black boxes apply to either control loop depending on which loop is experiencing the largest error. As it can be seen, the largest error is mainly being detected in the control loop whose messaging is in the middle of the bottom subfigure.

The use of the application profile, as it can be observed in Figure 4 or in Figure 5 (that shows details of the same CAN traffic around time 0.02s) slightly alters the transmission timing and ordering of the set of messages. Two main alterations can be observed:

- **Permutations:** By comparing the top and bottom subfigures of Figure 5, it can be observed that two pairs of granted sensor/control messages (white/black boxes) of the top subfigure right after time 0.02s affecting both control loops permute in time in the bottom subfigure. This is due to the fact that, using the application profile, granted sensor messages competing for the network win access to CAN depending on the errors that the sensed plants are experiencing.
- **Delayed transmissions:** In addition, the use of the application profile may delay pairs of granted sensor/control messages or general purpose messages. In particular, as it can be seen in Figure 4, the pair of granted sensor/control messages sent at time 0.04s in the top subfigure

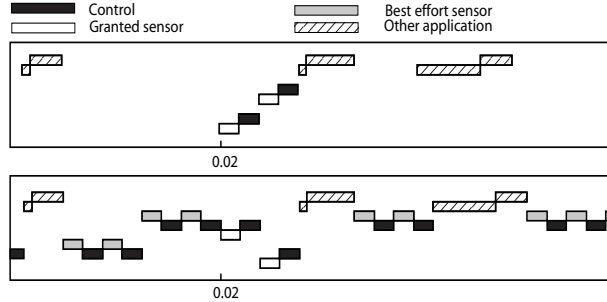


Figure 5: Zoom of the traffic around time 0.02.

are delayed in the bottom subfigure due to insertion in the bus of pairs of best effort sensor/control messages. For the same reason, as it can be seen in Figure 5, the fifth general purpose message (dashed box) starts later in the bottom subfigure.

In summary, comparing the standard CAN traffic and the resulting traffic when using the application profile, some anomalies can be detected and must be formally analyzed.

4.2 CAN worst-case response time analysis revisited

Response time analysis for CAN aims to provide a method of calculating the worst-case response time of each message. Then these values can be compared to the message deadlines to determine if the system is schedulable.

From [33], the worst-case response time of a message i on CAN without using the application profile is given by

$$R_i = J_i + W_i + C_i \quad (1)$$

where J_i is the queuing jitter, W_i is the queuing delay and C_i is the transmission time. Parameter J_i corresponds to the longest time between the initiating event and the message being queued and ready to be transmitted. Henceforth, and without losing generality, we will omit J_i because it is not affected by the application of the profile.

Parameter W_i corresponds to the longest time that the message can remain in the queue before commencing successful transmission on the bus. It comprises two elements: blocking time B_i due to messages which may be in the process of being transmitted and interference time due to higher priority messages which may win arbitration and be transmitted in preference to

message i . In [33] it is shown that

$$W_i = B_i + \sum_{j \in hp(i)} \left\lceil \frac{W_i + t_{bit}}{T_j} \right\rceil C_j \quad (2)$$

where $hp(i)$ is the set of messages with priorities higher than i , $\lceil a/b \rceil$ is notation for the *ceiling* function which returns the smallest integer greater than or equal to a/b , t_{bit} is the bit time, and T_j is the period of the j^{th} message. Equation (2) is solved using the recurrence relation

$$W_i^{k+1} = B_i + \sum_{j \in hp(i)} \left\lceil \frac{W_i^k + t_{bit}}{T_j} \right\rceil C_j \quad (3)$$

with $W_i^0 = B_i$ as a suitable initial condition. Denoting by D_i the i^{th} -message deadline, (3) iterates until, either

$$R_i^{k+1} = W_i^{k+1} + C_i > D_i \quad (4)$$

in which case the message is not schedulable, or

$$W_i^{k+1} = W_i^k \quad (5)$$

in which case the message i is schedulable provided

$$B_i = B^{MAX} \quad \text{and} \quad R_i^{k+1} = W_i^{k+1} + C_i \leq D_i. \quad (6)$$

The later condition has been shown (through equation (17) in [33]) to be a sufficient condition for schedulability test of the message i , where B^{MAX} is the blocking time that is equal to the transmission time of the longest possible CAN message (8 bytes), irrespective of the characteristics and priorities of the messages in the system.

The worst-case response time of the i^{th} message is given at the end of the recurrence iteration by

$$R_i^{k+1} = W_i^{k+1} + C_i = B^{MAX} + \sum_{j \in hp(i)} \left\lceil \frac{W_i^k + t_{bit}}{T_j} \right\rceil C_j + C_i. \quad (7)$$

The above schedulability analysis holds for messages with deadlines no greater than their periods [33].

4.3 Notation

The following notation is introduced:

- Ω denotes the set of messages to be scheduled.
- Ω_0 denotes the subset of control messages: $\Omega_0 = \{\tau_i \in \Omega : class(\tau_i) = 000\}$
- Ω_1 denotes the subset of granted sampler messages: $\Omega_1 = \{\tau_i \in \Omega : class(\tau_i) = 001\}$
- Ω_2 denotes the subset of general purpose messages: $\Omega_2 = \{\tau_i \in \Omega : class(\tau_i) = 010\}$
- Ω_3 denotes the subset of best effort sampler messages: $\Omega_3 = \{\tau_i \in \Omega : class(\tau_i) = 011\}$

If each granted sampler message or each best effort sampler message is followed by a control message, we can simplify the notation by joining Ω_1 or Ω_3 with Ω_0 as follows

- Ω_1^+ denotes the subset of join messages of Ω_0 and Ω_1 : $\Omega_1^+ = \{\tau_i \in \Omega : class(\tau_i) = 001 \text{ with } C_i = C_i^{001} + t_{interframe} + C_i^{000}\}$. That is, each i^{th} -message in Ω_1^+ has a transmission time that includes the transmission time of the i^{th} -granted sampler message, the interframe time, and the transmission time of the i^{th} -control message.
- Ω_3^+ denotes the subset of join messages of Ω_0 and Ω_3 : $\Omega_3^+ = \{\tau_i \in \Omega : class(\tau_i) = 011 \text{ with } C_i = C_i^{011} + t_{interframe} + C_i^{000}\}$. That is, each i^{th} -message in Ω_3^+ has a transmission time that includes the transmission time of the i^{th} -best effort sampler message, the interframe time and the transmission time of the i^{th} -control message.

5 Schedulability Analysis: Case Study

The schedulability analysis of CAN control applications using the application profile explores all the relevant cases that may happen due to the dynamic priority changes. The controlled plant dynamics causes priority changes in messages belonging to Ω_1^+ . Therefore, we have to analyze how these changes affect the worst case response time of messages belonging to

Ω_1^+ (*Case 1*), to Ω_2 (*Case 2*), and to Ω_3 (*Case 3*). For the sake of completeness of the analysis, we also have to analyze how the worst case response time of messages belonging to $\Omega_1^+ \cup \Omega_2$ is affected when best effort messages are introduced. This analysis demands a slight modification of (7), announced as *Case 0* and it is fully covered in *Case 4*.

5.1 Case 0

The worst case response time of a message was given in (7) where B^{MAX} is the blocking time corresponding to the transmission time of the longest possible CAN message. The application profile introduces new traffic: new pairs of best effort sensor/control messages appear in the network. Hence, (7) should also include the case that any message willing to enter the bus may be blocked by one sequence of these new pairs of messages. Formally, the recurrence when using the application profile should be

$$R_i^{k+1} = W_i^{k+1} + C_i = B^{MAX^+} + \sum_{j \in hp(i)} \left\lceil \frac{W_i^k + t_{bit}}{T_j} \right\rceil C_j + C_i \quad (8)$$

where

$$B^{MAX^+} = \max(B^{MAX}, \text{longest}(\tau_i) \in \Omega_3^+).$$

The situation analyzed in this case provokes delayed transmissions, as previously illustrated in Figures 4 and 5.

Observation 1 *When using the application profile, the worst case response time analysis of any message has to be performed using (8).*

5.2 Case 1

This case studies the worst case response time of message $\tau_k \in \Omega_1^+$ when a priority change occurs between two messages of Ω_1^+ . This case can be alternatively announced as follows. If R_k^{n+1} is the solution of (8) for message $\tau_k \in \Omega_1^+$ when the priority of $\tau_i \in \Omega_1^+$ is higher than the priority of $\tau_j \in \Omega_1^+$, then R_k^{n+1} may not be the solution of (8) for message τ_k when the priority of τ_j is higher than the priority of τ_i .

Proposition 1 *The priority change between any two messages of Ω_1^+ can alter the worst case response time of a message of Ω_1^+ .*

Proof.

The possible priority changes with respect to a message $\tau_k \in \Omega_1^+$ are:

1. $\tau_i \in hp(k)$ and $\tau_j \in hp(k)$ (the two messages that permute their priorities have higher priorities than τ_k): For this case, the worst case response time R_k^{n+1} of τ_k is not altered by the priority changes. R_k^{n+1} is given by

$$R_k^{n+1} = B^{MAX^+} + \sum_{j \in hp(k)} \left\lceil \frac{W_k^n + t_{bit}}{T_j} \right\rceil C_j + C_k. \quad (9)$$

Knowing that $hp(k)$ is the set of higher priority messages than τ_k , it holds that $hp(k)_{original} = hp(k)_{permuted}$. Therefore, to alter the order of two factors of the summand in (9) do not affect the result.

2. $\tau_i \notin hp(k)$ and $\tau_j \notin hp(k)$ (the two messages that permute their priorities have lower priorities than τ_k): Since there are no modifications in $hp(k)$ and therefore the worst case response time of τ_k remains the same.
3. $\tau_i \notin hp(k)$ and $\tau_j \in hp(k)$ (from the two messages that permute their priorities, one has higher priority than τ_k): We have to consider two scenarios. The first scenario consists in considering that all messages of Ω_1^+ always send 8 data bytes. This implies that all messages are equal in terms of transmission time if $T_i = T_j$ (their periods are the same), which implies that we can not make a difference between τ_i and τ_j . In this case, the worst case response time is not altered because

$$\left\lceil \frac{W_k + t_{bit}}{T_i} \right\rceil C_i = \left\lceil \frac{W_k + t_{bit}}{T_j} \right\rceil C_j \quad (10)$$

The second scenario consists in considering that messages of Ω_1^+ do not send the same number of bytes or that $T_i \neq T_j$. Therefore, condition (10) does not hold anymore and the worst case response time of τ_k can be altered by the permutation.

4. $\tau_i \notin hp(k)$ and $\tau_j = \tau_k$ (one of the two messages that permute their priorities is τ_k): In this case the number of summands of (9) before the permutation is less than the number that we can have after the permutation, which implies that the worst case response time of τ_k can drastically increase. The only situation in which we have guarantees that the permutation will benefit the worst case response time of τ_k is when τ_k is the lowest priority message of Ω_1^+ right before the permutation. In this case, any permutation will result in a reduction of its worst case response time.

□

The situation announced by Proposition 1 was illustrated in Figure 5. There it can be seen that pairs of granted sensor and control messages are permuted at run time. And this actually affects their worst case response time.

Observation 2 *Proposition 1 implies that schedulability of Ω is affected by messages of Ω_1^+ when priority changes occurs between messages of Ω_1^+ . Then, in order to guarantee schedulability while accounting for priority changes, the worst case response time analysis of all k^{th} messages of Ω_1^+ has to be performed considering that $hp(k) = \Omega_1^+ - \{\tau_k\}$, that is, considering that all messages of Ω_1^+ are of higher priority than the message under analysis, τ_k .*

5.3 Case 2 and Case 3

These cases share a common fact: both look at how the worst case response time of messages belonging to lower priority classes (Ω_2 or Ω_3) than messages of Ω_1^+ would be affected with respect to permutations of messages of Ω_1^+ . For this reason the rest of this subsection focuses on Case 2. The study of Case 3 would follow exactly the same argumentation.

Hence, this case studies the change on the worst case response time of message $\tau_k \in \Omega_2$ when a priority change occurs between two messages of Ω_1^+ , which can be alternatively announced as follows. If R_k^{n+1} is the solution of (8) for message $\tau_k \in \Omega_2$ when the priority of $\tau_i \in \Omega_1^+$ is higher than the priority of $\tau_j \in \Omega_1^+$, then R_k^{n+1} is also the solution of (8) for message τ_k when the priority of τ_j is higher than the priority of τ_i .

Proposition 2 *The priority change between any two messages of Ω_1^+ does not alter the worst case response time of any message of Ω_2 .*

Proof.

The worst case response time R_k^{n+1} of $\tau_k \in \Omega_2$ is given by

$$R_k^{n+1} = B^{MAX^+} + \sum_{j \in hp(k)} \left\lceil \frac{W_k^n + t_{bit}}{T_j} \right\rceil C_j + C_k. \quad (11)$$

Knowing that $hp(k)$ is the set of higher priority messages of τ_k , it holds that $\Omega_1^+ \subseteq hp(k)$. Therefore, to alter the order of two factors of the summand in (11) do not affect the result.

□

As it is illustrated in the example shown in Figure 5, permutations between granted messages of different control loops do not affect the worst case response time of messages of Ω_2 . The first general purpose message after 0.002 is not altered by the permutation occurring right before.

Observation 3 *Proposition 2 implies that schedulability of Ω is not affected by messages of Ω_2 when priority changes occurs between messages of Ω_1^+ . Therefore, to check schedulability we only require to evaluate the worst case response time of all messages of Ω_2 in only one scenario of priorities.*

5.4 Case 4

This case studies the change on the worst case response time of message $\tau_k \in \Omega_1^+ \cup \Omega_2$ when messages of Ω_3 are introduced in the bus. This case can be alternatively announced as follows. If R_k^{n+1} is the solution of (8) for message $\tau_k \in \Omega_1^+ \cup \Omega_2$ when considering that $\Omega = \Omega_1^+ \cup \Omega_2$, then R_k^{n+1} is also the solution of (8) for message τ_k when $\Omega = \Omega_1^+ \cup \Omega_2 \cup \Omega_3$.

Proposition 3 *The priority change between any two messages of Ω_3 does not alter the worst case response time of any message of $\Omega_1^+ \cup \Omega_2$. The same happens if new messages of Ω_3 appear in the bus or when messages of Ω_3 terminate.*

Proof.

The worst case response time R_k^{n+1} of any message $\tau_k \in \Omega_1^+ \cup \Omega_2$ is given by

$$R_k^{n+1} = B^{MAX^+} + \sum_{j \in hp(k)} \left\lceil \frac{W_k^n + t_{bit}}{T_j} \right\rceil C_j + C_k. \quad (12)$$

Knowing that $hp(k)$ is the set of higher priority messages of τ_k , R_k^{n+1} is not altered if two messages $\tau_i, \tau_j \in \Omega_3$ change their priorities because $\Omega_3 \notin hp(k)$. The same holds regardless of whether the number of messages in Ω_3 increases or decreases □

As it was illustrated in the example shown in Figure 5, the insertion of best effort messages could delay the transmission of general purpose messages. But their worst-case response time remains the same.

```

Schedulability test
{
1  Schedulable=true
2  For all  $\tau_k$  messages of  $\Omega_1^+$ 
3    Compute  $R_k^{n+1}$  (8) considering  $hp(k) = \Omega_1^+ - \{\tau_k\}$ 
4    If  $R_k^{n+1} > D_k$  then Schedulable=false
5  For all  $\tau_k$  messages of  $\Omega_2$ 
6    Compute  $R_k^{n+1}$  (8) considering  $hp(k) = \Omega_1^+$ 
7    If  $R_k^{n+1} > D_k$  then Schedulable=false
8  return Schedulable
}

```

Figure 6: Schedulability test for the control application profile

Observation 4 *Messages of Ω_3 are not guaranteed in the bus by definition of the application profile because they are considered best effort messages, with the lowest set of priorities. Therefore they have to be discarded in the schedulability analysis.*

6 Schedulability Test

To guarantee that all messages of CAN-based control applications enabled with the presented profile will meet their deadlines, the schedulability test shown in Figure 6 has to be applied. It is built upon observations 1, 2, 3 and 4. Observation 1 is included in lines 3 and 6. Observation 2 is reflected in lines 2-4. Observation 3 is reflected in lines 5-7. The last observation is also reflected in the test, in the sense that since best effort messages (Ω_3) are not guaranteed, they do not need to be analyzed for their worst case response time.

The new schedulability test restricts message set schedulability with respect to the standard schedulability test presented in [33]. In particular, the restriction affects only granted sensor and control messages when accounting for the priority changes. That is, a given specification of timing constraints for these messages may result in a feasible schedule according to [33] but unfeasible according to the presented schedulability test. In any case, priority changes do not affect general purpose message feasibility (recall observation 3).

The apparent limitation given by the feasibility restriction is overcome

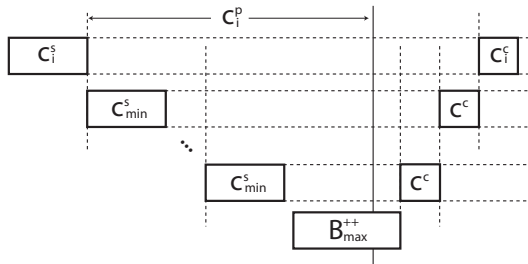


Figure 7: Worst case scenario when the control node computation time is longer than the interframe time.

by the improvement on control performance that is achieved when using the application profile (see [29]). It is left for future work to answer the question of whether it is possible to assess the trade off between the benefits in terms of control performance that can be reached vs. the schedulability restrictions that are imposed.

6.1 Extension

The schedulability analysis and the derived test presented in Sections 5 and 6 assumed that granted sampler messages or best effort sampler messages are always followed by a control message (messages of classes Ω_1^+ and Ω_3^+). The reason for this assumptions is that the time spent by the i^{th} controller node upon reception of any sampler message to both compute the control signal and place the message in a transmitter buffer, namely C_i^P , is less than the inter-frame time, $t_{interframe}$. Therefore, any other message of class Ω_1 , Ω_2 or Ω_3 ready to be transmitted, will not win buss access until the control message has been sent.

This assumption is realistic because at the faster CAN bit rate, the inter-frame time is 3ms. And for a standard microcontroller with a standard clock, e.g. for a PIC18F458 with a 20MHz oscillator, the execution time of a control algorithm is of the order of $10\mu s$. Hence, the controller computation time C_i^P is by far shorter than the interframe time.

However, in some implementations, the assumption may not hold and further analysis is required. In this case, when $C_i^P > t_{interframe}$, the transmission time of the messages belonging to classes Ω_1^+ and Ω_3^+ can no be computed as defined in Section 4.3.

The transmission time C_i of these messages can be longer. First, C_i has to include the time spent for transmitting the sensor message ($C_i^S = C_i^{001}$

or C_i^{011} for granted or best effort messages) and the control message (C_i^{000}). Note that according to the message classes definition, all control messages take the same transmission time, namely C^C , while C_i^S vary. Let C_{min}^S denote the the transmission time of a the shortest sensor message in Ω .

Second, C_i has to include the controller node processing time C_i^P , and all time spent by other messages that can be transmitted within the node processing time and that may delay the transmission of the control message. Figure 7 illustrates this case in the worst scenario. The top boxes in the left and right side represent the i^{th} sensor message and corresponding control message. The controller node processing time C_i^P is also represented. Although the i^{th} control message, C_i^C , is ready for transmission after C_i^P , it is blocked by B^{MAX++} (in the bottom of the subfigure) and by several control messages C^C generated by sensor messages transmitted during C_i^P . The more sensor message that enter during C_i^P , the more control messages C^C that can delay C_i^C . The worst case scenario is given when all the control messages have C_{min}^S . And B^{MAX++} is the maximum among the transmission time of the longest general purpose message or the longest time spent by the messaging of a j^{th} closed loop operation that includes sensor message, processing time and control message. Formally,

$$B^{MAX++} = \max(B^{MAX}, \text{longest}(\tau_j) \in \Omega_1^+ \cup \Omega_3^+) \quad (13)$$

with $C_j = C_j^S + C_j^P + C^C$.

By putting all together, the transmission time C_i of any message in Ω_1^+ and Ω_3^+ is given by

$$C_i = C_i^S + C_i^P + B^{MAX++} + \left\lfloor \frac{C_i^P}{C_{min}^S} \right\rfloor C^C + C_i^C \quad (14)$$

where $\lfloor a/b \rfloor$ is the notation for the *floor* function which returns the biggest integer smaller than or equal to a/b .

By updating the schedulability analysis and schedulability test in Sections 5 and 6 using (14), the case of having control nodes with a longer processing time than the interframe time is also covered.

7 Conclusions

This paper has presented a complete schedulability analysis of an application profile for networked embedded control systems that allows to implement an optimal bandwidth allocation policy among networked control

loops using the bitwise arbitration of CAN. The analysis has been carried out considering all possible interactions between CAN messages encoded using the application profile. The analysis has also considered current standard microcontrollers processing times, and has covered all possible hardware limitations.

This work is being extended by including different networked architectures, by analyzing whether multiple-input/multiple-output networked control systems can also be implemented with the application profile, and by assessing whether other existing approaches to control performance optimization for networked control systems can be similarly treated.

Further work will focus on studying whether alternative approaches to networked control systems such as those based on event-driven control can achieve similar control performance for a given set of networked control loops while reducing resource (bandwidth) utilization.

References

- [1] CAN Specification version 2.0. Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart, 1991.
- [2] Lim, D.; Anbuky, A., "A distributed industrial battery management network," *IEEE Transactions on Industrial Electronics*, vol.51, no.6, pp. 1181-1193, Dec. 2004.
- [3] Bertoluzzo, M.; Buja, G.; Menis, R., "Control schemes for steer-by-wire systems," *IEEE Industrial Electronics Magazine*, vol.1, no.1, pp.20-27, Spring 2007.
- [4] Baronti, F.; Lenzi, F.; Roncella, R.; Saletti, R.; Di Tanna, O., "Electronic Control of a Motorcycle Suspension for Preload Self-Adjustment," *IEEE Transactions on Industrial Electronics*, vol.55, no.7, pp.2832-2837, July 2008.
- [5] Hansson, H.A.; Nolte, T.; Norstrom, C.; Punnekkat, S., "Integrating reliability and timing analysis of CAN-based systems," *IEEE Transactions on Industrial Electronics*, vol.49, no.6, pp. 1240-1250, Dec 2002.
- [6] Cena, G.; Valenzano, A., "A protocol for automatic node discovery in CANopen networks," *IEEE Transactions on Industrial Electronics*, vol.50, no.3, pp. 419-430, June 2003.

- [7] J. Ferreira, L. Almeida, A. Fonseca, P. Pedreiras, E. Martins, G. Rodríguez-Navas, J. Rigo, and J. Proenza, "Combining operational flexibility and dependability in FTT-CAN," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 2, pp. 95 – 102, May 2006.
- [8] Buja, G.; Pimentel, J.R.; Zuccollo, A., "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, no.3, pp.225-233, Aug. 2007.
- [9] Gil-Castineira, F.; Gonzalez-Castano, F. J.; Franck, L., "Extending Vehicular CAN Fieldbuses With Delay-Tolerant Networks," *IEEE Transactions on Industrial Electronics*, vol.55, no.9, pp.3307-3314, Sept. 2008.
- [10] G. Buttazzo, "Research trends in real-time computing for embedded systems," *SIGBED Rev.*, vol.3, no. 3, pp. 1-10, Jul. 2006
- [11] D. Seto, J. P. Lehoczky, L. Sha, and K. Shin, "On task schedulability in real-time control systems," *17th IEEE Real-Time Systems Symposium*, Washington, DC, pp. 13-21, December 1996.
- [12] Q.C. Zhao, and D.Z. Zheng, "Stable and Real-Time Scheduling of a Class of Hybrid Dynamic Systems", *Discrete Event Dynamic Systems*, vol. 9, n. 1, pp. 45-64, 1999.
- [13] J. Eker, P. Hagander and K.-E. Årzén, "A Feedback Scheduler for Real-time Control Tasks," *Control Engineering Practice*, vol. 8, n.12, December 2000.
- [14] P. Martí, C. Lin, S. Brandt, M. Velasco and J.M. Fuertes, "Optimal State Feedback Based Resource Allocation for Resource-Constrained Control Tasks", *25th IEEE Real-Time Systems Symposium*, Lisbon, Portugal, December 2004.
- [15] D. Henriksson and A. Cervin, "Optimal On-line Sampling Period Assignment for Real-Time Control Tasks Based on Plant State Information," *44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*, Dec. 2005.
- [16] L. Palopoli, C. Pinello, A. Bicchi and A. L. Sangiovanni-Vincentelli, "Maximizing the Stability Radius of a Set of Systems Under Real-Time Scheduling Constraints", *IEEE Transactions on Automatic Control*, vol. 50, n. 11, pp. 1790- 1795, Nov. 2005

- [17] R. Castañé, P. Martí, M. Velasco, and A. Cervin, “Resource Management for Control Tasks Based on the Transient Dynamics of Closed-Loop Systems”, *18th Euromicro Conference on Real-Time Systems*, July 2006.
- [18] E. Bini and A. Cervin, “Delay-Aware Period Assignment in Control Systems,”, *29th IEEE Real-Time Systems Symposium*, December 2008.
- [19] P. Martí, C. Lin, S.A. Brandt, M. Velasco and Josep M. Fuertes, “Draco: Efficient resource Management for Resource-Constrained Control tasks,”, *IEEE Transactions on Computers*, vol. 58, n. 1, Jan. 2009.
- [20] M. El Mongi Ben Gaid, A. Çela and Y. Hamam, “Optimal Real-Time Scheduling of Control Tasks With State-Feedback Resource Allocation,” *IEEE Transactions on Control Systems Technology*, to appear.
- [21] Y. Tipsuwan and M. Y. Chow, “Control methodologies in networked control systems”, *Control Engineering Practice*, Vol. 11, pp. 1099-1111, 2003
- [22] G.C. Walsh, and Y. Hong, “Scheduling of networked control systems,” *IEEE Control Systems Magazine*, vol. 21, n. 1, pp. 57-65, Feb. 2001
- [23] D. Hristu-Varsakelis, “Feedback control systems as users of a shared network:communication sequences that guarantee stability”, *40th IEEE Conference on Decision and Control*, 2001
- [24] M.S. Branicky, S.M. Phillips, and W. Zhang “Scheduling and Feedback Co-Design for Networked Control Systems”, *IEEE Conference on Decision and Control*, 2002.
- [25] H. Rehbinder, and M. Sanfridson, “Scheduling of a limited communication channel for optimal control”, *Automatica*, Vol. 30, No. 3, pp. 491-500, March 2004.
- [26] M. Velasco, J.M. Fuertes, C. Lin, P. Martí, and S. Brandt, “A Control Approach to Bandwidth Management in Networked Control Systems.”, *30th Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2004.
- [27] M. Ben Gaid, A. Cela and Y. Hamam, “Optimal Integrated Control and Scheduling of Networked Control Systems with Communication Constraints: Application to a Car Suspension System,” *IEEE Transactions on Control Systems Technology*, vol. 14, n. 4 , pp. 776-787, 2006.

- [28] Al-Hammouri, A.T.; Branicky, M.S.; Liberatore, V.; Phillips, S.M., “Decentralized and dynamic bandwidth allocation in networked control systems,” In 20th International Parallel and Distributed Processing Symposium, April 2006.
- [29] M. Velasco, P. Martí, R. Castañé, J. Guardia and J.M. Fuertes, “A CAN Application Profile for Control Optimization in Networked Embedded Systems”, *32th Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2006.
- [30] Martí, P.; Yopez, J.; Velasco, M.; Villa, R.; Fuertes, J.M., “Managing quality-of-control in network-based control systems by controller and message scheduling co-design,” *IEEE Transactions on Industrial Electronics*, vol.51, no.6, pp. 1159-1167, Dec. 2004
- [31] K.M. Zuberi, and K.G. Shin, “Design and Implementation of Efficient Message Scheduling for Controller Area Network”, *IEEE Transactions on Computers*, vol. 49(2), pp. 182-188, Feb. 2000.
- [32] G. Juanole and G. Mouney, “Networked Control Systems: Definition and Analysis of a Hybrid Priority Scheme for the Message Scheduling,” *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2007.
- [33] R.I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, “Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised,” *Real-Time Systems*, vol. 35, n. 3, pp. 239-272, Apr. 2007