Exploiting System Dynamics for Resource-Efficient Automotive CPS Design

Leslie Maldonado*, Wanli Chang†, Debayan Roy‡, Anuradha Annaswamy*, Dip Goswami§, Samarjit Chakraborty‡
*Massachusetts Institute of Technology, USA †University of York, UK
‡Technical University of Munich, Germany §Eindhoven University of Technology, The Netherlands
†wanli.chang@york.ac.uk

***Abstract*—Automotive embedded systems are safety-critical, while being highly cost-sensitive at the same time. The former requires resource dimensioning that accounts for the worst case, even if such a case occurs infrequently, while this is in conflict with the latter requirement. In order to manage both of these aspects at the same time, one research direction being explored is to dynamically assign a mixture of resources based on needs and priorities of different tasks. Along this direction, in this paper we show that by properly modeling the physical dynamics of the systems that an automotive control software interacts with, it is possible to better save resources while still guaranteeing safety properties. Towards this, we focus on a distributed controller implementation that uses an automotive FlexRay bus. Our approach combines techniques from timing/schedulability analysis and control theory and shows the significance of synergistically combining the cyber component and physical processes in the cyber-physical systems (CPS) design paradigm.**

***Index Terms*—cyber-physical systems, resource efficiency, automotive systems, physical dynamics**

I. Introduction

Modern automotive systems deploy a large number of safety-critical functions and many of them are feedback control functions. These functions closely interact with the *physical* environment using sensors and actuators, and the computation of the control signals is performed on the Electrical/Electronic (E/E) architecture. Design of such cyber-physical systems (CPS) requires guarantee on functional and timing behavior in all scenarios including the worst case, even if it may rarely occur. This leads to significant resource (computation or communication) over-dimensioning — a particular concern for cost-sensitive domains like the automotive.

One method that has been extensively investigated to tackle these two conflicting requirements is to *dynamically* allocate resources according to the needs and priorities of tasks. That is, an application in a more urgent need to complete a task may temporarily get a higher resource allocation, subject to its priority. When implemented properly, this makes it possible to save resources while still satisfying safety requirements.

In the context of CPS, the timing behavior and resource requirements of applications are governed by the dynamics of the physical processes that are being controlled. *The main message of this paper is that by appropriately modeling and analyzing such dynamics, a better resource utilization is possible.* This essentially requires a synergistic study on both the cyber component and physical processes. In particular, in this paper we describe a novel interplay between control theory and timing/schedulability analysis.

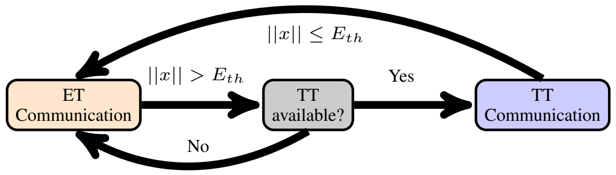

Figure 1. The dynamic resource allocation scheme

Setup under study: We focus on a specific automotive setup. It consists of a distributed implementation of controllers, where control signals are exchanged over a FlexRay communication bus. Hybrid communication protocols like FlexRay (and TTCAN) implement both time-triggered (TT) and event-triggered (ET) communication. TT communication offers deterministic timing behavior and hence TT slots are considered to be a more *valuable* resource. Therefore, a resource-efficient design requires a reduced usage of TT slots. Instead of assigning a dedicated TT slot to each control signal/application, reduction in the usage of TT slots is achieved under this setup in the following manner.

To start with, a control signal is communicated using ET communication. Now, as illustrated in Figure 1, when the norm of the state vector $\|x\|$ of the corresponding control application is larger than a predefined threshold E_{th} (i.e., the system is in the *transient* state), which could be caused by an external disturbance, the control signal associated with the application may request access to TT communication. As long as $\|x\|$ is smaller than or equal to E_{th}, the system is in the *steady* state and the ET communication for the control signal suffices. If a control signal is communicated using a TT slot, it experiences more deterministic transmission time, which can be exploited by the controller to reject the disturbance more promptly. Once the disturbance is rejected, the control signal relinquishes the TT slot and again starts using ET communication. When multiple applications share one TT slot and request access to it simultaneously, the ones with lower priorities (e.g., determined by the deadlines) have to continue to use ET communication while waiting for the higher-priority one to release the TT slot.

Under this dynamic resource allocation scheme, we define two parameters in disturbance rejection: (i) the *wait* time k_{wait} — the time a control application spends in ET communication after a disturbance occurs, while waiting for access to a TT slot; (ii) the *dwell* time k_{dw} — the time for which a control application's signal uses a TT slot to return to the steady state after getting access to the slot. No preemption is allowed.

TT slot allocation — how many TT slots? The challenge is to compute the minimum number of TT slots required, while ensuring that all the applications meet their performance requirements, i.e., disturbances are all rejected within the specified deadlines. To address this problem, we need to estimate the maximum time $\hat{\xi}_i$ (known as the worst-case response time or the worst-case settling time) a control application C_i needs (when sharing a TT slot with other applications) to bring the system back from the transient to steady state after an external disturbance has occurred. With this knowledge, we will be able to determine if a certain TT slot allocation (how TT slots are shared by applications) is schedulable (i.e., whether all the deadlines are guaranteed to be met), and further minimize the number of TT slots required.

Our contributions in this work are mainly twofold. First, since the disturbance is also rejected when using ET communication — albeit less effectively than when using TT communication — intuitively, the longer k_{wait} is, the shorter k_{dw} needs to be (because "more" disturbance is rejected while the system used ET communication). That is, k_{dw} decreases *monotonically* as k_{wait} increases. The exact relation between k_{wait} and its corresponding k_{dw} can be modeled by properly analyzing the system's switching dynamics, and will be used in determining the maximum wait time as well as the worst-case response time. In this paper we show that this intuitive monotonic assumption is not true, and that an accurate characterization of the non-monotonic relation between k_{wait} and k_{dw} can be used to reduce the number of TT slots needed.

Second, we prove the existence of and compute the maximum wait time for an application C_i, when it shares a TT slot with other applications. The worst-case response time can then be easily computed from the maximum wait time with the derived relation between k_{wait} and k_{dw}. This enables a schedulability analysis that determines whether an application is schedulable on one TT slot together with certain other applications and further minimization of the TT slots.

This work shows the importance of accurately modeling the system dynamics (i.e., the *physical* aspect of CPS) for better design of the computation/communication platform (i.e., the *cyber* part of CPS) on which the control algorithms run. It may be noted that while there has been a lot of work on how characteristics of the cyber component of CPS influences the design of control algorithms — e.g., by accounting for delays and numerical precisions of the computational platforms when designing control algorithms — the other direction, as we study in this paper has been much less explored.

Related work: Communication-aware design of distributed embedded control systems has been extensively investigated since [1]. Our work follows the dynamic resource allocation scheme in [2], exploiting the hybrid communication protocol. There have been efforts in [3] to compute the minimum number of slots required to guarantee performance even in the worst case. However, the fundamental issue is that the system dynamics has not been properly analyzed and modeled. This leads to either over-provisioning of resources or violation of control performance requirements.

Worst-case response time analysis of real-time tasks has been studied on different architectures [4], [5]. The schedulability analysis problem formulation in this paper, i.e., analyzing the worst-case settling time for switching control over hybrid communication, is motivated from the CAN response time analysis (pure ET communication) for fixed-priority non-preemptive policies in [6]. Unlike the iterative approach taken in [6] that provides no knowledge on whether there is a bound and what the bound is, we prove the existence of and compute the bound. Such a bound on the worst-case response time is particularly relevant for performance analysis of safety-critical control applications.

II. Problem Setting

We consider a distributed setting where multiple control applications $\{C_1, C_2, \cdots, C_n\}$ share a communication bus. Each application C_i is implemented using three tasks: sensing ($T_{s,i}$), control ($T_{c,i}$) and actuation ($T_{a,i}$). Both $T_{s,i}$ and $T_{c,i}$ are mapped on one electronic control unit (ECU) while $T_{a,i}$ is mapped on another, due to the spatial distribution of sensors and actuators. The control input calculated by $T_{c,i}$ is communicated to $T_{a,i}$ over the shared bus.

A. Hybrid communication protocol

In this work, we consider a provision of both TT and ET communication, such as FlexRay. Each time cycle in FlexRay is composed of a static and a dynamic segment. The static segment exhibits TT communication and comprises a number of TDMA (time division multiple access)-like slots of equal length Ψ. A message assigned to a static slot is transmitted within the corresponding time window. Thus, the start and end of a message transmission are precisely known. However, if no data arrives at the beginning of the slot, the entire slot of length Ψ goes unused. The dynamic segment implements ET communication and is partitioned into a number of mini-slots of equal length ψ, where typically $\psi \ll \Psi$. A message assigned to the dynamic segment may consume more than one mini-slot. Thus, the timing of a message depends on other preceding messages. When there is no data to be transmitted, only one mini-slot (ψ time units) is wasted. This results in time-varying transmission delay while the worst case may still be determined [7].

B. Mathematical modeling of control systems

For each control application C_i, we consider a discrete-time linear time-invariant (LTI) plant model given by

$$\begin{aligned} x_i[k+1] &= \Phi_i x_i[k] + \Gamma_{0,i} u_i[k] + \Gamma_{1,i} u_i[k-1], \\ y_i[k] &= C_i x_i[k], \end{aligned} \tag{1}$$

where x_i, u_i and y_i represent respectively the plant states, the control input, and the system output. Φ_i, $\Gamma_{0,i}$, $\Gamma_{1,i}$ and C_i are system matrices. Within a control loop, there are three operations: the sensors read the plant states, the controller computes the control input, and the actuator applies the control input. The sampling period between two consecutive time instants $t_i[k]$ and $t_i[k+1]$ is constant and denoted as h_i. The sensor-to-actuator delay is d_i. At $t_i[k] + d_i$, a new control input $u_i[k]$ (computed from $x_i[k]$) is applied to the plant and held constant till the next input at $t_i[k+1]+d_i$. In our setting, $d_i \leq h_i$. It is noted that k is a non-negative integer. To simplify the notation, we also use k to refer to the time $t[k]$ (and k_i for $t_i[k]$) later in the paper.

The control loop in our problem setting can be closed over either TT or ET communication. For TT communication, negligible delay $d_i \simeq 0$ can often be achieved by configuring task and message schedules[1]. For ET communication, due to the non-determinism, we must consider the worst case and a finite delay d_i. Individual state-feedback controllers can be designed to stabilize the system from a disturbance for ET and TT communication, respectively. The gains can be computed using optimal control principles [9], [10]. As discussed in Section I and illustrated in Figure 1, an application switches once from ET to TT communication in the process of rejecting a disturbance, due to non-preemption, unless it never gets access to the TT slot. Therefore, switching stability is ensured as long as both the switching systems are stable.

C. Dynamic resource allocation over hybrid communication

For each application C_i, the control requirement is to stabilize the system within a deadline (or desired response time) ξ_i^d after a disturbance has occurred. We consider independent periodic or sporadic disturbances with minimum inter-arrival time r_i, and, $\xi_i^d \leq r_i$. Under this assumption, a disturbance is expected to be rejected before another arrives.

We let ξ_i be the response time, i.e., the time taken to stabilize a system after a disturbance has occurred. If only the TT communication is used to close the control loop, the response time is denoted as ξ_i^{TT}. If only the ET communication is used, the response time is denoted as ξ_i^{ET}, and

$$\xi_i = k_{dw,i} + k_{wait,i}. \tag{2}$$

There is often $\xi_i^{TT} < \xi_i^d < \xi_i^{ET}$, and $\xi_i \leq \xi_i^d$ must be satisfied.

This work employs a dynamic resource allocation scheme over hybrid communication, and uses mixed TT and ET communication for closing the control loop, as illustrated in Figure 1. The main question is how to allocate the TT slots to the applications such that all the performance requirements are satisfied with the minimum number of TT slots. To answer this question, two topics must be investigated. First, for a given TT slot and applications sharing it, how to determine whether the deadline of any application can be met even in the worst case? In such a schedulability analysis, the maximum wait time and the worst-case response time need to be computed. Second, how to model the relation between the wait time and dwell time? It is used in determining the maximum wait time and computing the worst-case response time from the maximum wait time. Both issues will be addressed in the next sections.

[1]When the control loop involves heavy tasks like image processing, the delay coming from task execution will not be negligible. The reported method in this paper can still be applied, but the memory hierarchy also needs to be considered in the control systems design [8].

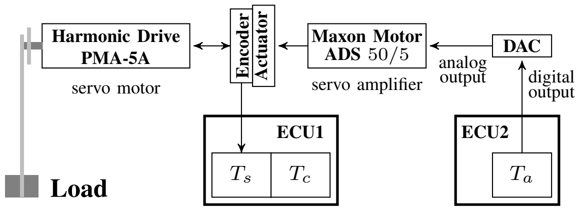

Figure 2. Servo motor position control system

III. Non-monotonic Relation between Dwell Time and Wait Time

In the previous works [3], it was assumed that the relation between the dwell time k_{dw} and the wait time k_{wait} is monotonic. That is, as k_{wait} increases, k_{dw} always decreases. However, this is not true for many systems. We assume A_1 and A_2 to be the closed-loop system matrices considering ET and TT communication, respectively. They can be derived using the plant dynamics (1) and corresponding controllers discussed in Section II-B. The closed-loop dynamics switches from A_1 to A_2 after a time interval k_{wait}. Given an initial condition x_0, the state trajectory before switching is given by

$$x_1[k] = A_1^k x_0, \tag{3}$$

and after switching it becomes

$$x_2[k_{wait}, k] = A_2^k x_1[k_{wait}] = A_2^k A_1^{k_{wait}} x_0, \tag{4}$$

where $x_2[k_{wait}, k]$ is the state of the system after evolving for k_{wait} samples with the dynamics A_1 and then k samples with A_2, from the initial condition x_0.

Even if A_1 is made asymptotically stable (eigenvalues of A_1 are less than unity) via a proper controller design for the ET communication, $\|x_1[k]\|$ does not necessarily monotonically decrease before switching. In fact, very often, $\|x_1[k]\|$ increases in the beginning and then decreases. Therefore, it may take more time for the norm of the state vector to go below E_{th} when the switching occurs later.

We have conducted an experiment to characterize the above non-monotonic behavior in a real-life setup, as illustrated in Figure 2. The shaft of the servo motor (Harmonic Drive) is attached to a rigid stick with 300g of weight at the end. The position of the motor shaft is measured by digital quadrature encoders attached to the motor shaft. The motor provides a desired amount of torque (computed by the control algorithm) using a digital-to-analog converter (DAC) via a servo amplifier (Maxon Motor).

The sampling period is chosen as $h = 20$ms. The sensor-to-actuator delay is 0.7ms when the message is transmitted over the TT slot. The maximum delay when using the ET communication is 20ms. The control objective is to keep the rigid load upright, i.e., both the angular position and the angular velocity should be zero. The disturbance is that the rigid load is moved by $45°$ from the upright position with zero angular velocity. The threshold E_{th} is set to be 0.1. Two state-feedback controllers are designed for the ET and TT communication, respectively.

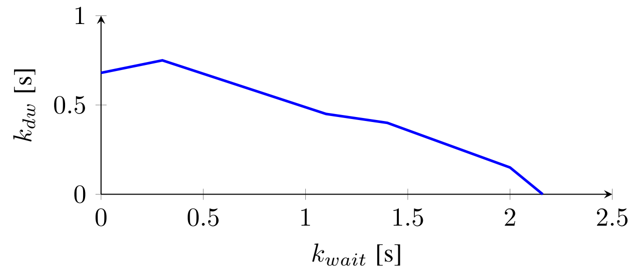

Figure 3. Experimental relation between the dwell time and the wait time

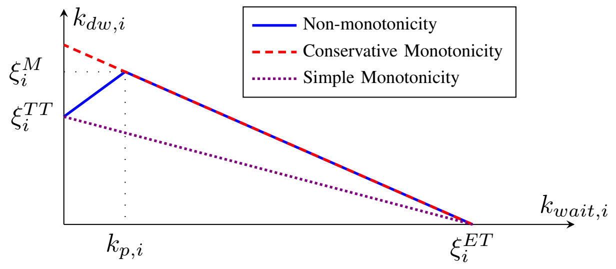

Figure 4. Approximated relation between the dwell time and the wait time

The relation between k_{wait} and k_{dw} for this application is shown in Figure 3. The response times (settling times) to bring the system back to the steady state with pure TT and ET communication are respectively $\xi_i^{TT} = 0.68$s and $\xi_i^{ET} = 2.16$s. It can be clearly observed that the timing behavior is split into two distinct regions: (i) the phase from $k_{wait} = 0$s to $k_{wait} = 0.3$s with positive gradient and (ii) the phase $k_{wait} > 0.3$s with negative gradient.

The non-monotonic relation between k_{dw} and k_{wait} can be approximately depicted using two piecewise linear curves, as shown in Figure 4. The maximum possible dwell time is denoted as ξ_i^M and the corresponding wait time is $k_{p,i}$. This is a more accurate approximation than previous works that assume monotonicity. It is noted that the actual curve in Figure 3 must be entirely below the model with two piecewise linear curves in Figure 4. That is, for any wait time, the corresponding modeled dwell time used for schedulability analysis in the next section must be longer than or equal to the actual dwell time. Otherwise, deadlines may be violated.

The total response time ξ_i is equal to $k_{wait,i} + k_{dw,i}$, as discussed before. Typically, due to the difference in response times of TT and ET communication, the gradient of the second part is between 0 and -1. Therefore, as the wait time $k_{dw,i}$ increases, the total response time ξ_i also increases.

Previous works have assumed a monotonic relation between the dwell time $k_{dw,i}$ and the wait time $k_{wait,i}$. The simple monotonic relation can be constructed by computing the response times of only TT communication ξ_i^{TT} and only ET communication ξ_i^{TT}, as shown in Figure 4. If the schedulability analysis and the allocation of TT slots are based on this simple monotonicity, the deadline may be violated, since the actual response time is longer (except at the two ends). This is clearly unacceptable. A conservative monotonic relation between the dwell time and the wait time can be constructed as shown in Figure 4, where for any wait time, the corresponding dwell time is longer than the actual dwell time. Thus, the deadline guarantees made based on this relation are valid. However, it leads to resource over-provisioning, since the actual response time is much shorter especially in the first phase where the dwell time increases. In this work, we compare the non-monotonic relation with the conservative monotonic relation on communication resource dimensioning. It is noted that the relation between the dwell time and the wait time may be modeled with three or more piecewise linear curves, to be closer to the actual behavior.

IV. Schedulability Analysis

As discussed before, when multiple applications are contending for a single TT slot, the non-preemptive scheduling is deployed. Priorities are determined according to the deadlines (desired response times) $\{\xi_{d,i}\}$. Smaller the value of $\xi_{d,i}$ is, higher is the priority of C_i to get access to the TT slot.

The schedulability of an application C_i is guaranteed if for every possible wait time $k_{wait,i}$, $\xi_i \leq \xi_i^d$. It has been discussed in Section III that the total response time ξ_i increases as the wait time $k_{wait,i}$ increases. Therefore, the maximum possible wait time, $\hat{k}_{wait,i}$, will lead to the worst-case response time, $\hat{\xi}_i$. The following situation is analyzed to derive $\hat{k}_{wait,i}$. When C_i requests for the TT slot, the lower-priority application with the maximum dwell time has just taken the slot (note non-preemption), and as C_i waits for the slot, all higher-priority applications request for it as many times as possible according to the minimum inter-arrival times of the external disturbances. It is assumed that every application interfering with C_i in the same TT slot requires its maximum possible dwell time, ξ_i^M. This leads to a conservative and safe schedulability analysis.

Assuming that n applications are sorted in the order of decreasing priority such that if $j < i$, C_j has a higher priority,

$$\hat{k}_{wait,i} = \max_{i<k\leq n} \xi_k^M + \sum_{j=1}^{i-1} \left\lceil \frac{\hat{k}_{wait,i}}{r_j} \right\rceil \xi_j^M. \tag{5}$$

The worst-case response time $\hat{\xi}_i$ can be computed as a sum of the above maximum wait time and its corresponding dwell time according to the modeled relation between the wait time and the dwell time, as discussed in Section III. If $\hat{\xi}_i$ is greater than ξ_i^d, then it implies that C_i is not schedulable on the shared TT slot. If it is less than or equal to ξ_i^d, then C_i can meet its deadline and thus is schedulable.

We note that (5) is essentially a difference equation, which can be expressed as

$$k_{wait}(l+1) = f(k_{wait}(l)), \tag{6}$$

where

$$f(k_{wait}(l)) = a + \sum_{j=1}^{i-1} \left\lceil \frac{k_{wait}(l)}{r_j} \right\rceil \xi_j^M, \tag{7}$$

and

$$a = \max_{i<k\leq n} \xi_k^M > 0. \tag{8}$$

If $k_{wait}(1) > k_{wait}(0)$, referring to (7),

$$f(k_{wait}(1)) \geq f(k_{wait}(0)), \tag{9}$$

since the ceiling function has the property,

$$x_1 > x_2 \to \lceil x_1 \rceil \geq \lceil x_2 \rceil. \tag{10}$$

According to (6),

$$k_{wait}(2) = f(k_{wait}(1)) \geq f(k_{wait}(0)) = k_{wait}(1). \tag{11}$$

By induction, we have

$$\forall l \in \mathbb{N}_0, \;\; k_{wait}(l+1) \geq k_{wait}(l). \tag{12}$$

With similar derivation, if $k_{wait}(1) < k_{wait}(0)$,

$$\forall l \in \mathbb{N}_0, \;\; k_{wait}(l+1) \leq k_{wait}(l). \tag{13}$$

If $k_{wait}(1) = k_{wait}(0)$

$$\forall l \in \mathbb{N}_0, \;\; k_{wait}(l+1) = k_{wait}(l). \tag{14}$$

Following the above findings, if we can bound $k_{wait}(l)$ as l approaches infinity, then the non-linear difference equation (6) has a fixed point and converges to the fixed point. The right hand side of (7) can be bounded using the following property of the ceiling function:

$$x \leq \lceil x \rceil < x + 1. \tag{15}$$

Linear combinations of $\lceil x \rceil$ are used to get lower and upper bounds of $f(k_{wait}(l))$ as

$$a + \sum_{j=1}^{i-1} \frac{\xi_j^M}{r_j} k_{wait}(l) \leq f(k_{wait}(l)) \tag{16}$$

$$f(k_{wait}(l)) < a + \sum_{j=1}^{i-1} \frac{\xi_j^M}{r_j} k_{wait}(l) + \sum_{j=1}^{i-1} \xi_j^M, \tag{17}$$

which can be simplified as

$$a + m k_{wait}(l) \leq f(k_{wait}(l)) < a' + m k_{wait}(l), \tag{18}$$

where

$$a' = a + \sum_{j=1}^{i-1} \xi_j^M, \;\; m = \sum_{j=1}^{i-1} \frac{\xi_j^M}{r_j} < 1. \tag{19}$$

It is noted that m is the sum of the TT slot utilizations for all the higher-priority applications in the worst case that we are considering. If $m \geq 1$, then C_i is not schedulable on this TT slot together with the other applications.

Starting from $k_{wait}(0)$ and computing the upper bound,

$$\begin{aligned}
k_{wait}(1) &= f(k_{wait}(0)) < a' + m k_{wait}(0),\\
k_{wait}(2) &= f(k_{wait}(1)) < a' + m k_{wait}(1) = a' + \\
&\quad m f(k_{wait}(0)) < a' + m a' + m^2 k_{wait}(0),\\
k_{wait}(3) &= f(k_{wait}(2)) < a' + m k_{wait}(2) = a' + \\
&\quad m f(k_{wait}(1)) < a' + m a' + m^2 k_{wait}(1)\\
&\quad = a' + m a' + m^2 f(k_{wait}(0))\\
&\quad < a' + m a' + m^2 a' + m^3 k_{wait}(0),\\
&\vdots\\
k_{wait}(l) &= f(k_{wait}(l-1)) < a' + m a' + \\
&\quad m^2 a' + \cdots + m^{l-1} a' + m^l k_{wait}(0).
\end{aligned}$$

Since $m < 1$,

$$\lim_{l \to \infty} k_{wait}(l) < \frac{a'}{1-m}. \tag{20}$$

Similarly, the lower bound of $k_{wait}(l)$ as l approaches infinity can be computed as

$$\lim_{l \to \infty} k_{wait}(l) \geq \frac{a}{1-m}. \tag{21}$$

Now we can conclude that the fixed point of (6) is reached as n approaches infinity and we have proven the existence of the maximum wait time $\hat{k}_{wait}$. It is bounded by (20), which can be used to compute the worst-case response time $\hat{\xi}$, as discussed at the beginning of this section.

A simple algorithm is deployed to allocate n control applications to m TT slots ($m < n$). It begins with the slot S_1 and tries to map the control applications one by one (starting from C_1) to S_1. This continues as long as these applications are schedulable on S_1 as per the above schedulability analysis. As discussed before, C_i is schedulable on a particular slot S_j if it can meet its deadline ξ_i^d. After adding a new application C_i into S_1, if one of the applications allocated to S_1 is not schedulable (note that C_i influences the schedulability of previously allocated applications), a new TT slot S_2 is added and C_i is moved to S_2. This process continues until all applications are assigned to TT slots. The total number of necessary TT slots, which is assumed to be less than or equal to m, is then returned. Finding the optimal slot allocation is NP-hard and thus we use a heuristic.

V. Case Study

As a case study, we consider 6 control applications sharing a common FlexRay bus. The sampling period for each of them is assumed as 0.02s. A FlexRay cycle is 5ms with 10 static slots in the 2ms TT segment while the rest is ET. The simulation is performed using MATLAB/Simulink and TrueTime [11]. The timing parameters required for schedulability analysis and slot allocation can thus be obtained and are listed in Table I. The relation between the dwell time $k_{dw,i}$ and the wait time $k_{wait,i}$ for every application can then be constructed. The maximum dwell time for the monotonic case is denoted as $\xi_i'^M$. We conduct the slot allocation considering both the non-monotonic and monotonic relations. Correspondingly, we compare the resource dimensionings.

First, we allocate applications to the TT slots based on the non-monotonicity. Starting from C_3 with the shortest deadline ξ_3^d and thus the highest priority, the maximum wait time $\hat{k}_{wait,3}$ is 0, since there are no other applications sharing the TT slot S_1 yet. Therefore, the worst-case response time $\hat{\xi}_3$ is equal to $\xi_3^{TT} = 0.39$ and less than $\xi_3^d = 2$, which means that C_3 is schedulable on S_1. Then we add C_6 with the second shortest desired response time and thus the second highest priority into S_1. For C_6, according to (20), the maximum wait time $\hat{k}_{wait,6} = 0.669$, which is used to compute the worst-case response time $\hat{\xi}_6 = 1.589$ based on Figure 4.

Since $\hat{\xi}_6 < \xi_6^d = 6$, C_6 is schedulable on S_1, when together with C_3. It is noted that the schedulability of C_3 could change after C_6 is added, and thus has to be analyzed again. According

Table I
TIMING PARAMETERS FOR APPLICATIONS [IN SEC]

Application C_i	r_i	ξ_i^d	ξ_i^{TT}	ξ_i^{ET}	ξ_i^M	$k_{p,i}$	$\xi_i'^M$
C_1	200	9.5	1.68	11.62	5.30	2.27	6.59
C_2	20	6.25	2.58	8.59	2.95	1.34	3.50
C_3	15	2	0.39	3.97	0.64	0.69	0.77
C_4	200	7.5	2.50	10.40	4.03	1.92	4.94
C_5	20	8.5	2.75	10.63	4.58	1.97	5.62
C_6	6	6	0.71	7.94	0.92	0.67	1.01

to (20), the maximum wait time $\hat{k}_{wait,3} = \xi_6^M = 0.92$, which is used to compute the worst-case response time $\hat{\xi}_3 = 1.515$ based on Figure 4. Since $\hat{\xi}_3 < \xi_3^d$, C_3 is schedulable on S_1, when together with C_6. Then we add C_2 into S_1. In this scenario, $\hat{\xi}_3 > \xi_3^d$. Therefore, C_3 is not schedulable and C_2 is added to a new TT slot S_2. Following the same method, we assign C_2 and C_4 to S_2. C_5 and C_1 share the TT slot S_3.

The responses of all six applications are shown in Figure 5, assuming that all disturbances occur at $t = 0$. The blue region represents the period when the control input is transmitted in the TT segment, while the orange region indicates the ET communication. The horizontal dashed red line is the threshold, below which the system is in the steady state. The vertical black line denotes the desired response time. It is evident that all the control applications are able to meet their deadlines and thus achieve satisfactory control performances.

Now we derive the number of TT slots required in the monotonic case. C_3 and C_6 can still share S_1. C_2 cannot be added to S_1, and thus is allocated to S_2. When C_4 is added to S_2, according to (20), the maximum wait time $\hat{k}'_{wait,2} = \xi_4'^M = 4.94$, which is used to compute the worst-case response time $\hat{\xi}'_2 = 6.426 > \xi_2^d$. Therefore, C_2 is not schedulable when together with C_4. C_4 is then allocated to S_3. Following the same approach, C_5 is allocated to S_4 and C_1 is allocated to S_5. It can be seen that when assuming the conservative monotonic relation between the dwell time and the wait time, 5 TT slots are required for all the applications to be schedulable. The amount of communication resources required in the monotonic case is 67% more than the non-monotonic case, which is a significant number.

VI. CONCLUDING REMARKS

This paper explores a major challenge in CPS, i.e., efficient resource dimensioning while ensuring safety. In particular, we consider that a mixture of resources are dynamically allocated to multiple applications. For such a setting, we accurately characterize the impact of switching from a lower to a higher quality resource as well as the interference from other applications. The method reported in this paper is not restricted to the automotive FlexRay, but can be generally applied to other types of hybrid communication (such as wired and wireless communication), and other embedded control systems with limited resources, such as in the robotic domain.

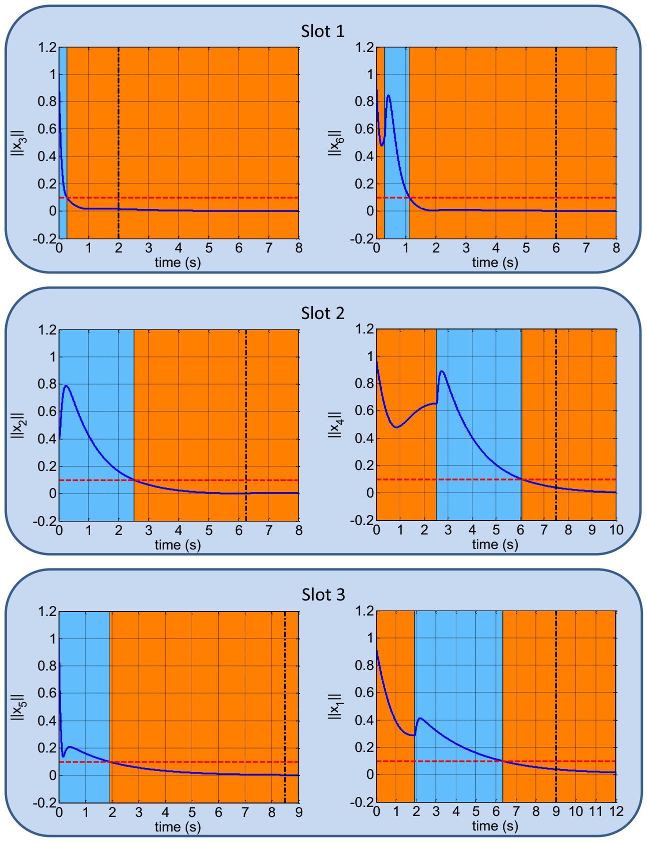

Figure 5. Responses of all six applications with disturbances at time 0

REFERENCES

[1] W. Zhang, M. Branicky, and S. Phillips, "Stability of networked control systems," *IEEE Control Systems*, vol. 21, no. 1, pp. 84–99, 2001.

[2] D. Goswami, R. Schneider, and S. Chakraborty, "Re-engineering cyber-physical control applications for hybrid communication protocols," in *DATE*, 2011.

[3] A. Masrur, D. Goswami, S. Chakraborty, J. Chen, A. Annaswamy, and A. Banerjee, "Timing analysis of cyber-physical applications for hybrid communication protocols," in *DATE*, 2012.

[4] H. Zeng, M. D. Natale, A. Ghosal, and A. Sangiovanni-Vincentelli, "Scheduling optimization of time-triggered systems communicating over the FlexRay static segment," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 1, pp. 1–17, 2011.

[5] J. Yao, X. Xu, and X. Liu, "Mixcps: Mixed time/event-triggered architecture of cyber-physical systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 923–937, 2016.

[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, no. 3, pp. 239–272, 2007.

[7] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing analysis of the FlexRay communication protocol," *Real-Time Systems*, vol. 39, no. 1, pp. 205–235, 2008.

[8] W. Chang, D. Goswami, S. Chakraborty, L. Ju, C. Xue, and S. Andalam, "Memory-aware embedded control systems design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 4, pp. 586–599, 2017.

[9] W. Chang, D. Goswami, S. Chakraborty, and A. Hamann, "Os-aware automotive controller design using non-uniform sampling," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 4, 2018, article 26.

[10] W. Chang and S. Chakraborty, "Resource-aware automotive control systems design: A cyber-physical systems approach," *Foundations and Trends in Electronic Design Automation*, vol. 10, no. 4, pp. 249–369, 2016.

[11] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. Årzén, "How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime," *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 16–30, 2003.