Time-triggered Implementations of Mixed-Criticality Automotive Software

Dip Goswami[1], Martin Lukasiewycz[2], Reinhard Schneider[3] and Samarjit Chakraborty[3]
[1]Alexander von Humboldt Research Fellow, TU Munich, Germany
[2]TUM CREATE, Singapore
[3]Institute for Real-Time Computer Systems, TU Munich, Germany
(dip.goswami@tum.de, martin.lukasiewycz@tum-create.edu.sg, reinhard.schneider@rcs.ei.tum.de and samarjit@tum.de)

***Abstract*—We present an automatic schedule synthesis framework for applications that are mapped onto distributed time-triggered automotive platforms where multiple Electronic Control Units (ECUs) are synchronized over a FlexRay bus. We classify applications into two categories (i) safety-critical control applications with stability and performance constraints, and (ii) time-critical applications with only deadline constraints. Our proposed framework can handle such mixed constraints arising from timing, control stability, and performance requirements. In particular, we synthesize schedules that optimize control performance and respects the timing requirements of the real-time applications. An Integer Linear Programming (ILP) problem is formulated by modeling the ECU and bus schedules as a set of constraints for optimizing both linear or quadratic control performance functions.**

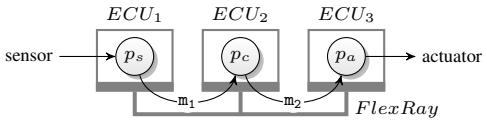
I. Introduction

Mixed critical systems such as automotive software consist of a mix of safety-critical control applications with performance and stability constraints and time-critical applications with only deadline constraints. The constraints associated with the control applications (e.g., engine control, brake-by-wire, vehicle dynamics control) are crucial for ensuring the safety of the vehicle. Timing constraints associated with the real-time, e.g., driver-assistance applications, help improve usability and driving comfort. Both classes of applications co-exist, interact and share common resources like electronic control units (ECUs), and buses. Scheduling and platform design to ensure such a mix of timing, control performance and stability constraints is a challenging problem, especially for architectures with multiple ECUs and buses with complex protocols like FlexRay [1].

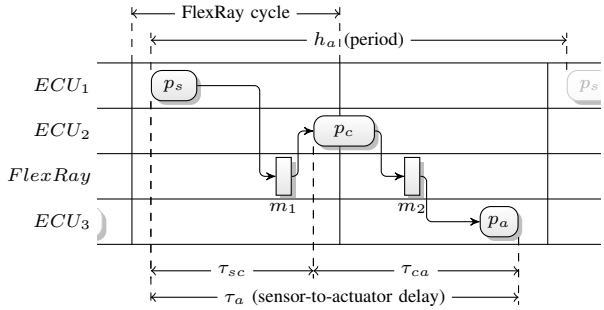
Real-time applications only have deadline constraints and predefined sampling rates. However, both stability and performance of control applications depend on the choice of sampling periods, which in turn determine the deadlines imposed by them. In this work, we consider the problem of implementing such a mix of applications on a time-triggered platform with eCos-based [2] ECUs connected via FlexRay-based buses. We proposed a co-design scheme that finds schedules for both the ECUs and the buses such that all the above-mentioned constraints are satisfied.

Related Work The problem of jointly scheduling applications with a mix of real-time and control performance constraints has not been sufficiently addressed in the literature. However, there has been a flurry of recent work on platform/schedule synthesis for control applications such that control functionalities are robust and they provide the desired performance [3], [4], [5], [6], [7], [8], [9], [10]. While these approaches have made excellent contributions in the area of control/scheduler co-design, they may be further extended along the following lines, which is what we attempt to do in this paper: (i) accurately model the relation between control performance and stability on one hand and sampling period and feedback delay (because of a distributed implementation) on the other hand, (ii) model the low-level details of the platform, i.e., ECU operating systems and bus arbitration policies, (iii) respect constraints from other concurrently running (e.g., real-time) applications, and (iv) automatically synthesize schedules for real-life distributed systems with a large number of design parameters.

978-3-9810801-8-6/DATE12/©2012 EDAA



(a) FlexRay-based automotive control system.



(b) FlexRay-based scheduling.

Fig. 1. (a) Distributed implementation of a control application on a FlexRay-based network and (b) a scheduling example for the processes and messages.

A. Problem Statement

We focus on automatic synthesis of a schedule S for a set of applications $\mathcal{A}$ implemented on a FlexRay based time-triggered platform. The applications consist of sets of data-dependent processes P and messages M. The processes are implemented on a set of ECUs R that use an eCos-based non-preemptive time-triggered scheduling. Two processes $p, \tilde{p} \in P$ communicate via the FlexRay bus by sending a message $m \in M$. The schedule S is determined by the start times of all processes and slot/cycle assignment of the messages. Applications are periodic with h_a being the sampling period

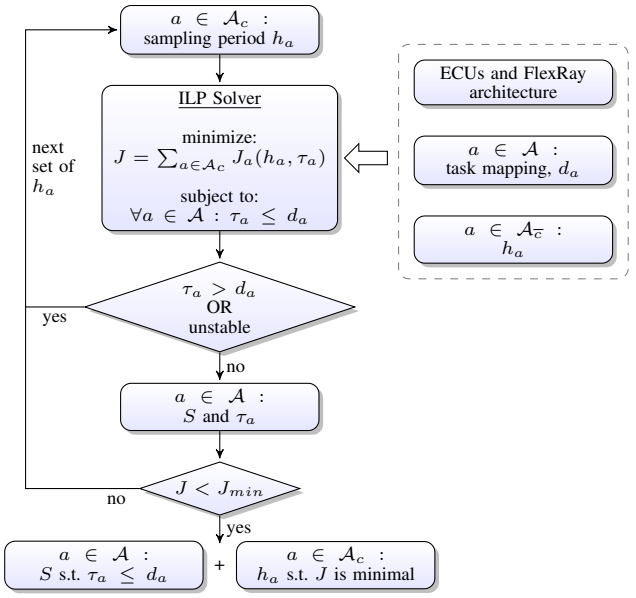


Fig. 2. Overview of our proposed optimization flow.

of application $a \in \mathcal{A}$. The applications are classified into two categories: control and real-time. Let $\mathcal{A}_c$ and $\mathcal{A}_{\overline{c}}$ be the sets of all control and real-time applications, respectively. The set of all applications $\mathcal{A} = \mathcal{A}_c \cup \mathcal{A}_{\overline{c}}$. The overall architecture and the associated timing diagram are illustrated in Fig. 1(a) and (b), respectively.

The stability (constraint) and performance (optimization objective) of a control application depend on (sampling) period h_a and sensor-to-actuator delay $\tau_a < h_a$. Hence, the performance of control application $a \in \mathcal{A}_c$ is denoted by $J_a(h_a, \tau_a)$. Each application $a \in \mathcal{A}$ is specified by its maximum end-to-end delay d_a.

The overall goal is to find:

1. Sampling periods h_a for each $a \in \mathcal{A}_c$
2. Schedule S for all $a \in \mathcal{A}$

Such that:

1. The control applications are stable and the following performance function is minimized:

$$J = \sum_{a \in \mathcal{A}_c} J_a(h_a, \tau_a) \tag{1}$$

2. The end-to-end delay $\tau_a < d_a$ for all $a \in \mathcal{A}$

B. Overview of Our Scheme

An overview of our co-design framework is shown in Fig. 2. In Section II, we introduce the feedback control model that is specific to the time-triggered implementation platform. The platform consists of a set of ECUs with eCos-based non-preemptive time-triggered scheduling, and a FlexRay bus for communication among the ECUs. We model the platform by a set of constraints imposed by eCos and FlexRay. Section III presents the formulation of various constraints imposed by eCos and FlexRay. We consider a given architecture, i.e., the number of ECUs and topology are predefined. Moreover, the task partitioning and their mapping to the ECUs are given for all applications $a \in \mathcal{A}$. The real-time applications $a \in \mathcal{A}_{\overline{c}}$ are specified by the sampling period h_a and the maximum end-to-end delay d_a.

Based on all the constraints coming from the given architecture (described in Section III), task mapping and timing requirements of the applications, we formulate an Integer Linear Programing (ILP) problem and invoke an ILP solver. The objective of the ILP formulation is to find a schedule S for applications $a \in \mathcal{A}$ such that $\tau_a < d_a$ and cost function (1) is minimal for a particular set of h_a for $a \in \mathcal{A}_c$. It is notable that the ILP formulation needs a linear or quadratic cost function and hence, we use an approximated version of the cost function (1) to cast the ILP problem. The approximated cost function is computed for the control applications based on the range of operating points of the designer's interest. In Section IV-C, we describe the approximation of the cost function (1). We repeat the above for every allowable sampling period h_a until we get S for all $a \in \mathcal{A}$ with $\tau_a < d_a$ and h_a such that (1) is globally minimized for all $a \in \mathcal{A}_c$. Note that the allowable set of h_a is restricted by the constraints coming from FlexRay protocol and range of operating points of the control applications. Finally, S is the schedule for both processes at the ECUs and messages at the bus for the optimal implementation of control applications with sampling periods h_a, $a \in \mathcal{A}_c$. It may be noted that we have kept the design space of sampling periods out of the ILP formulation to obtain a Pareto-optimal front between the load coming the control applications and the control performance. Section IV presents the overall control/scheduling co-design framework and the experimental results are shown in Section V.

II. Distributed Feedback Control System

A feedback control system aims to achieve the desired behavior of a dynamical system by applying appropriate inputs to the system. In a dynamical system, the relation between inputs and outputs is modeled by a set of differential equations, called the *state-space model*,

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{2}$$

where $x(t) \in R^n$ is the system *state* and $u(t) \in R$ is the *control input* to the system. $A \in R^{n \times n}$ and $B \in R^{n \times 1}$ are the system and input matrices, respectively. *State-feedback control* essentially implies the design of $u(t)$ as a function of the states $x(t)$ (feedback signals) so as to meet certain high-level design requirements.

A. Modeling feedback control over network

In general, a feedback control loop performs mainly three operations:

- measure the states $x(t)$ (*measure*),
- compute input signal $u(t)$ (*compute*) and,
- apply the computed $u(t)$ to the plant (2) (*actuate*).

Performing these operations in a continuous fashion in any implementation platform requires infinite computation power. Hence, in a digital implementation platform of such feedback loop, these operations are performed only at discrete-time intervals (sampling instants). In view of the above operations, the control tasks are broadly classified into three categories: sensor task p_s, controller task p_c and actuator task p_a. At

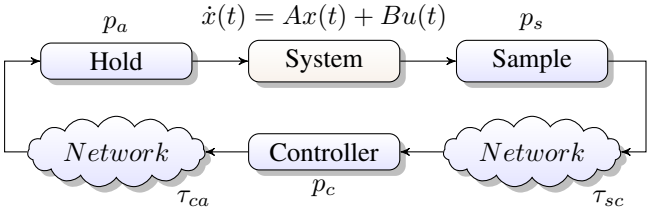


Fig. 3. Control over network.

the sampling instants, p_s reads the continuous states $x(t)$ or *measure*, p_c computes $u(t)$ or *compute* and sends the output to the task p_a or *actuate*.

Fig. 3 shows the overall implementation of such feedback loop over the network. p_s *samples* the states $x(t)$ periodically with sampling period h_a at t_k and it is sent over the communication bus to p_c with a delay τ_{sc}. Next, p_c computes the control input $u(t)$ and in this work, we consider state-feedback controllers of the form $u(t) = Kx(t)$ where K is the state-feedback gain which needs to be designed. $u(t)$ is sent over the communication bus to p_a with a delay of τ_{ca}. p_a applies $u(t)$ to the system at time $t = (t_k + \tau_{sc} + \tau_{ca})$. Further, p_a *holds* $u(t)$ until the next update comes, i.e.,

$$u(t) = Kx(t_k), \quad t_k \leq t \leq t_{k+1} \tag{3}$$

The total delay $\tau_a = (\tau_{sc} + \tau_{ca})$ between the sampling of the system states and the receipt of the corresponding control value is called *sensor-to-actuator* delay. It may be noted that the response time of p_s is included in τ_{sc} and similarly, the response times of p_c and p_a are included in τ_{ca} (Fig. 1(b)).

Given the input signal (3) and the sensor-to-actuator delay being $\tau_a < h_a$, the continuous-time system (2) becomes a *sampled-data* system [11],

$$x[k+1] = A_d x[k] + B_0(\tau_a)u[k] + B_1(\tau_a)u[k-1], \tag{4}$$

where

$$A_d = e^{Ah_a}, B_0(\tau_a) = \int_0^{h_a - \tau_a} e^{At}dt \cdot B, B_1(\tau_a) = \int_0^{\tau_a} e^{At}dt \cdot B$$

Putting (3) in (4), we get the following *closed-loop system*,

$$x[k+1] = A_d x[k] + B_0(\tau_a)Kx[k] + B_1(\tau_a)Kx[k-1]. \tag{5}$$

In (5), we assume that $u[-1] = 0$ for $k = 0$. Next, we define new system states $z[k] = [\ x[k] \quad x[k-1]\ ]'$ and we get,

$$z[k+1] = A_{cl}(h_a, \tau_a)z[k], \tag{6}$$

where

$$A_{cl}(h_a, \tau_a) = \begin{bmatrix} 0 & I \\ B_1(\tau_a)K & A_d + B_0(\tau_a)K \end{bmatrix}, \tag{7}$$

where I is the unity matrix. Stability of the overall closed-loop system is governed by the properties of $A_{cl}(h_a, \tau_a)$ and for stability, the absolute value of maximum eigenvalue of $A_{cl}(h_a, \tau_a)$ should be less than unity, i.e.,

$$|\lambda_{max}(A_{cl}(h_a, \tau_a))| < 1. \tag{8}$$

The closed-loop system might become unstable when τ_a is long and fails to meet (8).

III. Time-triggered Implementation

For determining a schedule S that considers all platform constraints, we propose an ILP formulation. First, the scheduling policy of the ECUs is an eCos-based operating systems without preemption. The static segment of a FlexRay bus is used for the bus system. Finally, the end-to-end timing constraints are defined.

A. eCos-based Task Scheduling

The tasks on each ECU (including sensors and actuators) have to be scheduled in compliance with the used operating system scheduler. Here, the processes $P_r = \{p | target(r) = p, p \in P\}$ are scheduled for each ECU $r \in R$ separately where $target(p)$ determines the resource on which the task is executed. Each task is associated with its period h_p and worst-case execution time e_p. For each ECU, the schedule S is determined by the offset/start time of each task.

The formulation for the eCos operating system requires the following variables:

- $s_p \in \mathbb{R}_{[0,h_p]}$ - start time of task $p \in P$
- $f_p \in \mathbb{R}_{[0,h_p]}$ - finish time of task $p \in P$
- $of_p \in \{0, 1\}$ - offset for finish time $p \in P$
- $y^{i,j}_{p,\tilde{p}} \in \{0, 1\}$ - 1 if job i of task $p \in P$ finished before job j of task $\tilde{p} \in P$

The constraints are determined as follows:

$\forall p \in P_r$:

$$f_p + h_p \cdot of_p = s_p + e_p \tag{9}$$

$\forall p, \tilde{p} \in P_r, p \neq \tilde{p}, i = \{0, .., \frac{2 \cdot H_r}{h_p} - 1\}, j = \{0, .., \frac{2 \cdot H_r}{h_{\tilde{p}}} - 1\}$:

$$i \cdot h_p + s_p + e_p \leq j \cdot h_{\tilde{p}} + s_{\tilde{p}} + 2 \cdot H_r \cdot (1 - y^{i,j}_{p,\tilde{p}}) \tag{10}$$

$$j \cdot h_{\tilde{p}} + s_{\tilde{p}} + e_{\tilde{p}} \leq i \cdot h_p + s_p + 2 \cdot H_r \cdot y^{i,j}_{p,\tilde{p}} \tag{11}$$

Constraint (9) determines the finish time of each task. In case the process finishes in the next cycle, the finish time is smaller than the start time. In this case, the variable of_p becomes 1 to fulfill constraint (9). The constraints (10) and (11) ensure that two tasks never preempt each other within two hyper-periods ($2 \cdot H_r$ with $H_r = \underset{p \in P_r}{lcm}(h_p)$). The variable $y^{i,j}_{p,\tilde{p}}$ is used for switching, i.e., one of the constraints (10) or (11) is trivially satisfied depending on $y^{i,j}_{p,\tilde{p}}$.

B. FlexRay Scheduling

The FlexRay bus is used as a central communication bus in the proposed architecture. This bus system enables a synchronization of the ECUs and the establishment of a fully time-triggered system. For message scheduling, we assume the static segment of FlexRay is used. The static segment of the FlexRay is organized in n_{fx} static slots with each slot having a duration of e_{fx} and available payload of l_{fx} in bytes. The static segment is transmitted in the beginning of every FlexRay cycle which has a duration of h_{fx}. A message that is transmitted on the FlexRay bus, is defined by the repetition that is deduced from the period, the execution time that equals the slot duration, and the length in bytes.

The constants used are as follows:

- h_{fx} - cycle duration of the FlexRay bus
- n_{fx} - number of static slots
- e_{fx} - duration of a static slot (it holds $n_{fx} \cdot e_{fx} \leq h_{fx}$)
- l_{fx} - capacity of one FlexRay static slot in bytes
- h_m - period of message $m \in M$

- $r_m = \frac{h_m}{h_{fx}}$ - repetition of message $m \in M$
- $e_m = e_{fx}$ - transmission time of message $m \in M$
- l_m - length of a message $m \in M$ in bytes

For each message, a slot and base cycle has to be determined. From this value, the assignments of slots to ECUs is deduced implicitly. For the schedule S, the following values have to be determined for each message:

- s, b - slot id and base cycle for each message $m \in M$

The slot and base is encoded in the binary variable $[s, b]_m$. Independent of the FlexRay version, the following variables are required:

- $s_m \in \mathbb{R}_{[0,h_m]}$ - start time of message $m \in M$
- $f_m \in \mathbb{R}_{[0,h_m]}$ - finish time of message $m \in M$
- $[s, b]_m \in \{0, 1\}$ - slot and base cycle pair for each message $m \in M$ with $s \in \{1, .., n_{fx}\}$, $b \in \{0, .., r_m - 1\}$
- $s_r \in \{0, 1\}$ - becomes 1 if slot s is assigned to ECU r and 0 otherwise

These constraints are defined as follows:

$\forall m \in M$:

$$\sum_{s \in \{1,..,n_{fx}\}} \sum_{b \in \{0,..,r_m - 1\}} [s, b]_m = 1 \tag{12}$$

$$s_m = \sum_{s \in \{1,..,n_{fx}\}} \sum_{b \in \{0,..,r_m - 1\}} ((s-1) \cdot e_{fx} + b \cdot h_{fx}) \cdot [s, b]_m \tag{13}$$

$$f_m = s_m + e_{fx} \tag{14}$$

$\forall s \in \{1, .., n_{fx}\}, b \in \{0, .., \frac{lcm_{m \in M}(h_m)}{h_{fx}} - 1\}$:

$$\sum_{m \in M} l_m \cdot [s, b\%r_m]_m \leq l_{fx} \tag{15}$$

$\forall s \in \{1, .., n_{fx}\}$:

$$\sum_{r \in R} s_r \leq 1 \tag{16}$$

$\forall m \in M, s \in \{1, .., n_{fx}\}, b \in \{0, .., r_m - 1\}, r = target(m)$:

$$[s, b]_m \leq s_r \tag{17}$$

Constraint (12) states that each message is scheduled in exactly one slot at a specific base cycle. Constraint (13) determines the start time of a message transmission based on the slot and base cycle. Constraint (14) defines the finish time of a message transmission depending on the duration of the transmission of a static slot. Constraint (15) ensures that the capacity of a slot is not exceeded. Constraint (16) states that a slot can be assigned to a most one ECU. Constraint (17) ensures that if a message is send in a specific slot, the slot is assigned to the sending ECU.

C. Sensor-to-Actuator Delay

Control applications have strict end-to-end timing constraints. In this case, it becomes necessary to define constraints that restrict the latencies of these applications. Here, $d_a \in \mathbb{R}$ defines the maximal tolerated delay of an application $a \in \mathcal{A}$. Additionally, Π_a determines all paths of a function from the source processes (sensors) to the sink processes (actuators).

The end-to-end delay is determined in an additive manner as follows. Along the paths the delay is determined by the sum of the execution times of all tasks and messages as well as the waiting time between the data-dependent tasks and messages, respectively. To constrain the end-to-end delay, the following variables are required:

- $\tau_a \in \mathbb{R}_{[0,d_a]}$ - the maximal delay of a function $a \in \mathcal{A}$
- $w_{p,\tilde{p}} \in \mathbb{R}_{[0,h_p]}$ - waiting time between the finish of $p \in P \cup M$ and start of $\tilde{p} \in P \cup M$
- $ow_{p,\tilde{p}} \in \{0, 1\}$ - 0 if $p \in P \cup M$ starts before $\tilde{p} \in P \cup M$, 1 otherwise

The constraints are defined as follows:

$\forall a \in \mathcal{A}, \pi \in \Pi_a, (p, \tilde{p}) \in \pi$:

$$w_{p,\tilde{p}} = s_{\tilde{p}} - f_p + h_p \cdot ow_{p,\tilde{p}} \tag{18}$$

$\forall a \in \mathcal{A}, \pi \in \Pi_a$:

$$\tau_a \geq \sum_{t \in \pi \cap P} e_p + \sum_{t \in \pi \cap M} e_{fx} + \sum_{(p,\tilde{p}) \in \pi} w_{p,\tilde{p}} \tag{19}$$

Constraint (18) determines the waiting time between a data-dependent task and message. The binary variable $ow_{p,\tilde{p}}$ ensures that the waiting time is always within the predefined bounds, i.e., not negative. Constraint (19) determines the end-to-end delay of a function. The end-to-end delay is defined as the maximal latency along all paths of a function.

IV. Co-design scheme

The controller gain K in (3) is designed based on the Linear Quadratic Regulator (LQR) for sensor-to-actuator delay $\tau_a = 0$ and sampling period h_a, $a \in \mathcal{A}_c$. This essentially boils down to designing an LQR gain K with A_d and constant B_0 ($B_1(\tau_a) = 0$) in (4). In a distributed platform modeled as described in Section III, the sensor-to-actuator delay τ_a is non-zero and constant. Thus, the resultant closed-loop system behaves as described in Section II and the condition for stability of the control applications is given by (8). In this section, we describe the proposed scheme for optimal implementation of the control applications respecting all the platform constraints.

A. Performance function

It can be noticed from (7) that the closed-loop system depends on (i) the sampling period h_a (ii) the sensor-to-actuator delay τ_a. Based on this observation, we consider the following commonly used quadratic performance function for each control application (for minimization),

$$J_a(h_a, \tau_a) = \sum_{k=0}^{N} \int_{kh_a}^{(k+1)h_a} [u(t)^2 + x(t)'x(t)]dt, \tag{20}$$

where $u(t)$ and $x(t)$ are as per (2) and N is the total number of samples under consideration. In the following paragraph, we explore the dependency among $J_a(h_a, \tau_a)$, h_a and τ_a. We consider the following system for illustration,

$$A = \begin{bmatrix} 0 & 1.0 \\ 20 & 35 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{21}$$

We compute the performance of the system (21) utilizing (20) considering an initial condition $x_1(0) = 0.3$ and $x_2(0) = 0.1$. The open-loop system is highly unstable (as one *pole* is at 35.56) and hence, the system has stringent timing requirements. The FlexRay cycle length $h_{fx} = 5ms$. The possible sampling periods are $h_a \in \{5, 10, 20, 40, 80, 160, 320\}$ (in ms) as per FlexRay 2.1. The system behavior becomes unacceptable for $h_a \geq 40ms$ in terms of systems's response time. Therefore, we restrict our interest to $h_a \in \{5, 10, 20\}$ (in ms). The system with feedback delay τ_a leads to system dynamics

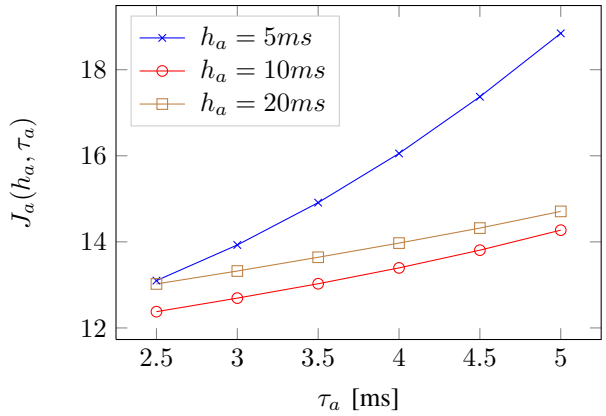


Fig. 4. Performance $J_a(h_a, \tau_a)$ vs. sensor-to-actuator delay τ_a and sampling period h_a.

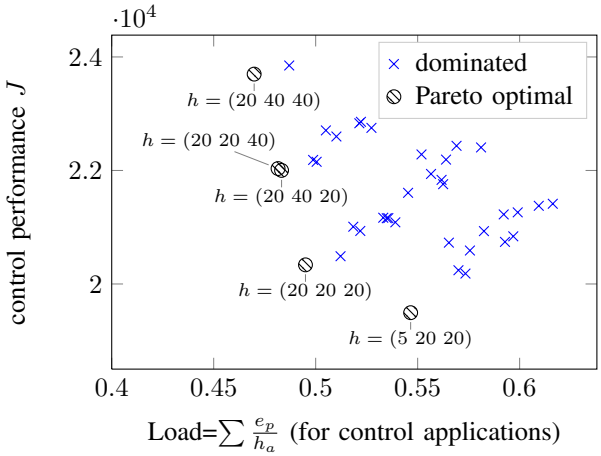


Fig. 6. Performance J vs. execution load resulting from the control applications.

given by (6) which is utilized to compute the performance given by (20). Given an architecture, a feedback signal experiences a minimum τ_a (because of the finite execution times of the processes and the transmission times of the messages). We assume that minimum $\tau_a = 2.5ms$. Let us assume that the performance becomes unacceptable for $\tau_a > 5ms$, i.e., $d_a = 5ms$. Hence, it is sufficient to observe the behavior of the system for a τ_a in the range of $2.5ms$ to $5ms$. Fig. 4 shows the plot of $J_a(h_a, \tau_a)$ vs. τ_a for $h_a = 5ms$, $h_a = 10ms$ and $h_a = 20ms$. It can be noticed from the plots that the performance deteriorates with increasing τ_a (for constant h_a) and the performance degradation is closely linear with τ_a. Based on these observations, we approximate (20) as a function of τ_a which is discussed in the next subsection.

B. Approximated performance function

The performance function (20) is approximated by,

$$\hat{J}_{a,n}(\tau_a) = \sum_{i=0}^{n} \kappa_i \tau_a^i, \tag{22}$$

where n is the order of the approximation polynomial and κ_i are the coefficients. Note that $\hat{J}_{a,n}(\tau_a)$ is computed for every h_a. The accuracy of the approximation improves with increasing order of the polynomial, i.e., higher n.

The accuracy depends on the choice of operating points of the control applications, i.e., sampling periods h_a and sensor-to-actuator delay τ_a. Generally, the sampling period is less than $40ms$ for the safety-critical control applications in the automotive domain. With increasing τ_a, the performance deteriorates and the system might even become unstable failing to meet the stability condition (8). Hence, the choice of its allowable range is restricted by the desired performance constraints and the stability condition. We are especially interested in the approximation accuracy of quadratic and linear approximations as they are utilized as an objective function in the ILP formulation for co-design (discussed in the following subsection). We notice that the quadratic (less than 0.005% error) and linear (less than 0.5% error) approximations provide sufficiently good accuracy for common operating points in the automotive domain, i.e., a sampling period in the range of $[5, 40]$ (in ms) and a sensor-to-actuator delay in the range of $[0, 10]$ (in ms).

C. Co-design algorithm

Our overall goal is to find sampling periods h_a for $a \in \mathcal{A}_c$ such that cost function (1) is minimized and timing requirements of all applications are respected, i.e., $\tau_a < d_a$ for $a \in \mathcal{A}$. For control performance optimization, we formulate an ILP problem which models the platform by the set of constraints described in Section III. The ILP optimizes either linear or quadratic approximations of control performance functions. Hence, we use quadratic and linear approximations described in Section IV-B and use objective function,

$$J = \sum_{a \in \mathcal{A}_c} \hat{J}_{a,n}(\tau_a), \tag{23}$$

where $n = 1$ (for linear approximation) and $n = 2$ (for quadratic approximation). Utilizing linear cost function (23) in the ILP solver, we obtain schedule S for $a \in \mathcal{A}$ that minimizes (1) for a particular h_a and a given range of τ_a (determined from the range of operating points), $a \in \mathcal{A}_c$. For global optimization, we invoke the ILP solver for all possible combinations of sampling periods h_a for $a \in \mathcal{A}_c$ that are allowed by the platform and within the range of operating points. Thus, we obtain the sampling periods h_a for $a \in \mathcal{A}_c$ and schedules S for $a \in \mathcal{A}$ corresponding to the minimum (1) and $\tau_a < d_a$.

V. Experimental Results

In this section, we present the experimental results considering following setup: There are 4 ECUs that are connected via a FlexRay bus. Each ECU is running eCos-based non-preemptive time-triggered operating systems. The FlexRay cycle length is $h_{fx} = 5ms$ and number of static slot $n_{fx} = 20$ with each slot duration $e_{fx} = 0.2ms$. For $h_{fx} = 5ms$, the feasible sampling periods are $h_a \in \{5, 10, 20, 40, 80, 160, 320\}$ (in ms). There are five applications: three control applications - a_1, a_2, a_3 and two real-time applications - a_4, a_5. Table I shows the given architecture, applications, task mappings, d_a and minimum τ_a of these applications. The execution times e_p of all control processes are chosen between $[0.3, 0.8]$ (in ms). For the two real-time applications: $h_{a4} = 10ms$ and $h_{a5} = 5ms$.

We consider three safety-critical automotive control plants: brake-by-wire a_1, engine control a_2 and cruise control a_3

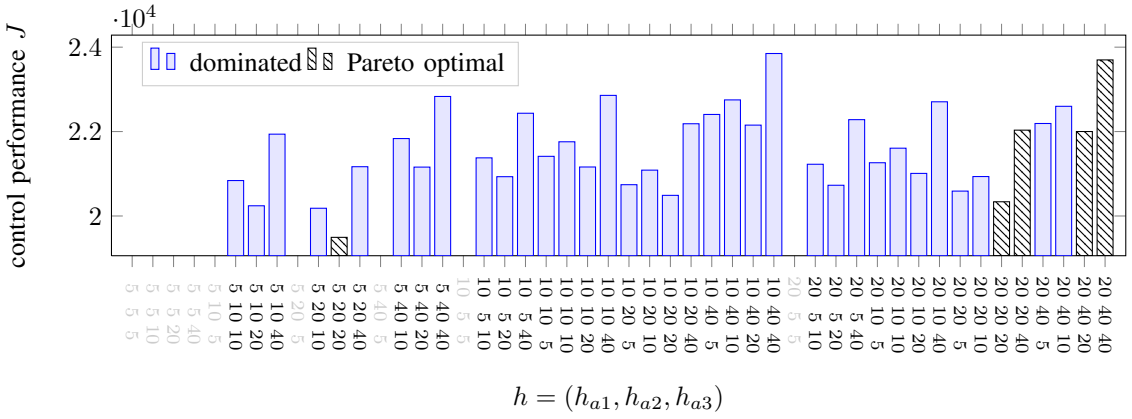


Fig. 5. Performance J for different combinations of the periods h of the applications.

Applications	Paths	Task Mappings	d_a (ms)	min. τ_a (ms)
a_1	$T_1 \rightarrow m_1 \rightarrow T_2 \rightarrow m_2 \rightarrow T_3$	$T_1 : ECU_1$, $T_2 : ECU_2$, $T_3 : ECU_3$	5	1.775
a_2	$T_4 \rightarrow m_3 \rightarrow T_5 \rightarrow m_4 \rightarrow T_6$	$T_4 : ECU_1$, $T_5 : ECU_4$, $T_6 : ECU_2$	10	2.55
a_3	$T_7 \rightarrow m_5 \rightarrow T_8 \rightarrow m_6 \rightarrow T_9$	$T_7 : ECU_4$, $T_8 : ECU_1$, $T_8 : ECU_2$	10	2.275
a_4	$T_{10-13} \rightarrow m_{7-10} \rightarrow T_{14} \rightarrow m_{11} \rightarrow T_{15}$	$T_{10,14} : ECU_1$, $T_{11,15} : ECU_2$, $T_{12} : ECU_3$, $T_{13} : ECU_4$	20	3.7
a_5	$T_{16,17} \rightarrow m_{12,13} \rightarrow T_{18} \rightarrow m_{14} \rightarrow T_{19}$	$T_{16,18} : ECU_1$, $T_{17,19} : ECU_2$	20	3.7

TABLE I
GIVEN ARCHITECTURE, APPLICATIONS AND TASK MAPPINGS OF THE SETUP UNDER CONSIDERATION.

(we skip their details). a_1 has the higher degree of criticality compared to a_2 and a_3. Hence, we choose $h_{a1} \in \{5, 10, 20\}$ while $h_{a2,a3} \in \{5, 10, 20, 40\}$. We used the approximated cost function (23) based on the stated operating points of the applications.

Results and discussions: We implemented the co-design scheme with CPLEX ILP solver [12] on an Intel $i5$ 2.53 GHz with 4 GB RAM. The runtime of the entire optimization process is 4143sec. Fig. 5 shows the performance J for various combinations of sampling periods h_a, $a \in \mathcal{A}_c$. The optimal sampling periods for the control applications are $h = (5, 20, 20)$ (in ms). Due to higher criticality, a_1 needs lower sampling period (i.e., 5ms) for the optimal operation. It may be noted that $h = (5, 20, 20)$, $h = (20, 5, 20)$ and $h = (20, 20, 5)$ do not provide similar performance. Moreover, the lower sampling periods do not essentially guarantee to provide better performance, e.g., $h = (10, 20, 5)$ or $h = (20, 10, 5)$. Hence, the control performance depends on the criticality, the task mapping, the execution demand of the control processes and finally, the sampling period. The ILP finds the sampling periods for optimal periods for a given set of rest of the parameters.

The sampling period indicates how frequently the processes are executed and how many messages are transmitted. Naturally, the execution demand or load from the control applications goes up with lower sampling periods. It is reflected in the schedules with more than two applications having sampling periods 5ms which fail to find feasible solutions (indicated by faded columns in Fig. 5). Fig. 6 shows Pareto-optimal front in the solution space obtained from the ILP between J and load coming from the control applications. The nature of the front indicates that the schedules with lower sampling periods impose higher execution load and provide better control performance, and vice versa. It is also evident from the Pareto-front that the optimal solutions are not obvious and the design space has to be explored.

VI. CONCLUSIONS

The automotive softwares are mix of the safety-critical control applications with stringent stability and performance requirements and the time-critical applications with stringent deadline constraints. This paper presents a co-design framework for automatic schedule synthesis of optimal implementation of such applications with mixed criticality onto a time-triggered platform. The main technical challenge is the formulation of an ILP by integrating the exact models of platform and application constraints with control stability (constraint) and performance (optimized). The framework can be extended to heterogeneous architecture with both preemptive (e.g., OSEK, OSEKTime) and non-preemptive scheduling policies on the ECUs and, time- and event-triggered (e.g., CAN, dynamic segment of FlexRay) arbitration on the buses (possible future work).

REFERENCES

[1] "The FlexRay Communications System Specifications, Ver. 2.1," www.flexray.com.
[2] "eCos," ecos.sourceware.org.
[3] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *IEEE RTSS*, 2008.
[4] A. Quagli, D. Fontanelli, L. Greco, L. Palopoli, and A. Bicchi, "Designing real-time embedded controllers using the anytime computing paradigm," in *IEEE ETFA*, 2009.
[5] S. Samii, P. Eles, Z. Peng, P. Tabuada, and A. Cervin, "Dynamic scheduling and control-quality optimization of self-triggered control applications," in *IEEE RTSS*, 2010.
[6] A. Cervin and P. Alriksson, "Optimal on-line scheduling of multiple control tasks: A case study," in *ECRTS*, 2006.
[7] S. Samii, A. Cervin, P. Eles, and Z. Peng, "Integrated scheduling and synthesis of control applications on distributed embedded systems," in *DATE*, 2009.
[8] R. Castane, P. Mart, M. Velasco, and A. Cervin, "Resource management for control tasks based on the transient dynamics of closed-loop systems," in *ECRTS*, 2006.
[9] F. Zhang, K. Szwaykowska, W. Wolf, and V. J. Mooney, "Task scheduling for control oriented requirements for Cyber-Physical Systems," in *IEEE RTSS*, 2008.
[10] A. Anta and P. Tabuada, "On the benefits of relaxing the periodicity assumption for networked control systems over can," in *IEEE RTSS*, 2009.
[11] A. Y. Bhave and B. H. Krogh, "Performance bounds on state-feedback controller with network delay," in *IEEE CDC*, 2008.
[12] IBM, "ILOG CPLEX," http://www.ibm.com/software/, Version 12.2.