A Structured Methodology for Pattern based Adaptive Scheduling in Embedded Control

SUMANA GHOSH, SOURADEEP DUTTA, SOUMYAJIT DEY, and PALLAB DASGUPTA,
Indian Institute of Technology Kharagpur

Software implementation of multiple embedded control loops often share compute resources. The control performance of such implementations have been shown to improve if the sharing of bandwidth between control loops can be dynamically regulated in response to input disturbances. In the absence of a structured methodology for planning such measures, the scheduler may spend too much time in deciding the optimal scheduling pattern. Our work leverages well known results in the domain of network control systems and applies them in the context of bandwidth sharing among controllers. We provide techniques that may be used a priori for computing co-schedulable execution patterns for a given set of control loops such that stability is guaranteed under all possible disturbance scenarios. Additionally, the design of the control loops optimize the average case control performance by adaptive sharing of bandwidth under time varying input disturbances.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: Embedded control, adaptive scheduling, schedulability analysis, control performance

ACM Reference format:
Sumana Ghosh, Souradeep Dutta, Soumyajit Dey, and Pallab Dasgupta. 2017. A Structured Methodology for Pattern based Adaptive Scheduling in Embedded Control. *ACM Trans. Embed. Comput. Syst.* 16, 5s, Article 189 (September 2017), 22 pages.
https://doi.org/10.1145/3126514

1 INTRODUCTION

The rapid growth in the number of features in a modern automobile has led the industry to seek a migration from a federated architecture, where each control task has a dedicated Electronic Control Unit (ECU), to an integrated architecture, where ECUs are shared between multiple control components. This requirement has opened up interesting questions about how well can we share ECUs among control components *without affecting* the control performance of each component in an adverse way.

The traditional solution to this problem, which is widely studied in the literature [4, 5, 7, 22], assumes that each control task executes periodically with pre-defined arrival times and deadlines.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue.
Authors' addresses: S. Ghosh, S. Dutta, S. Dey, and P. Dasgupta, Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India; emails: sumanaghosh@cse.iitkgp.ernet.in, jusouradeep@gmail.com, {soumya, pallab}@cse.iitkgp.ernet.in.

The periodicity is determined from control theoretic analysis and reflects the interval at which the controller needs to inspect the system under control (called the *plant*) in order to guarantee the desired control performance. Given the periodicity and deadlines of each control task mapped to a common ECU, a schedulability analysis is performed to determine the scheduling strategy that guarantees the completion of all control tasks are completed on time.

Research performed in [7, 15, 22] shows that controllers mapped to the same ECU can trade-off each other's bandwidth in response to the input disturbances to achieve a better overall control performance. In the new paradigm, a controller may increase the rate at which it samples the plant when it senses a disturbance above normal margins, and return to the original sampling rate only after it has rejected the disturbance. In order to accommodate the higher sampling rate, the other controllers sharing the same ECU temporarily work with a lower sampling rate – sometimes at the expense of marginal degradation in their control performance. It has been shown that an objective function representing a combined control performance criterion can be optimized by dynamically regulating the sampling rates of the controllers [4, 7].

Typically for accommodating the non-idealities of the platform, the sampling rates are chosen with sufficient margin, so that failure to complete the loop in one or more sampling windows may not degrade the control performance in many state of the plant [3]. Particularly under marginal disturbances, the controller can often skip one or more loop executions and allow another controller to utilize the residual bandwidth. We formalize this aspect to develop a method for adaptive bandwidth sharing among controllers.

At the heart of our approach is a theory which defines the patterns of skipping loop executions that do not compromise the stability guarantee. For multiple controllers sharing an ECU, we present a design methodology for adaptive sharing of bandwidth between controllers in which the pattern of regulation is guided by the choice of *compatible* loop execution patterns, where *compatibility* is with respect to schedulability. Adaptive control and related scheduling techniques have also been recommended in upcoming automotive standards like Adaptive AUTOSAR [11] which supports runtime configuration management and dynamic scheduling of embedded software. Our theory can be leveraged in various ways for designing software implementable adaptive control strategies. The approach followed in this paper is as follows:

(1) We consider the design of a set of n plant-control loops.
(2) For each plant we are given a *disturbance threshold*. Disturbances below this threshold are called *nominal disturbances* and those above are called *high disturbances*.
(3) For each plant, separate control performance requirements are formally specified for high and nominal disturbances, where the control performance guarantee for high disturbance is typically more stringent than the nominal one (for example, a sharper damping curve). We refer to these modes as the *nominal mode* and the *extremal mode*. For nominal disturbance levels we have an additional mode of the controller called the *marginal mode*, which has a relaxed control performance requirement. Intuitively, When all the controllers are experiencing nominal disturbances, they will stay in their nominal modes, which are chosen in a way that the overall control cost is minimized.

    When one or more controllers experience high disturbances, we switch them to their respective extremal modes, so that they can eliminate such disturbances quickly. In some cases this is schedulable without disturbing the other controllers from their nominal mode of operation. In the other cases we relegate some of the controllers to their marginal modes and use the residual bandwidth to elevate the controllers experiencing high disturbances to their extremal modes.

(4) We present novel synthesis algorithms which compute the set of co-schedulable loop execution patterns that guarantee the specified control requirements for the various modes along with stable switching sequences which provide admissible ways for effecting safe simultaneous mode switching for all control loops.
(5) Further, we use additional cost metrics to choose between the admissible loop execution patterns so as to optimize the overall control performance.

Essentially our goal is to develop an offline approach for choosing the loop execution patterns for the extremal, nominal and marginal modes of the controllers, and design an adaptive control strategy that orchestrates the switching between such pre-defined patterns at runtime in response to disturbances. We report gains in control performance in all experiments as compared to the non-adaptive control scheduling. Interesting trends are also reported and discussed in the paper.

The paper is organized as follows. Section 2 formalizes the notion of recurrent loop execution patterns. Section 3 develops the platform based formalism for adaptive sharing of ECU bandwidth among controllers. Section 4 describes the algorithmic framework. Section 5 presents the illustration of the key concepts together with the experimental evidence of the gain in control performance with the proposed approach. Section 6 discusses related work in this field. Section 7 summarizes the conclusions from the paper.

2 FORMALIZING LOOP EXECUTION PATTERNS

Let $P = (A_p, B_p, C_p)$ be a linear, discrete time invariant plant defined as follows.

$$\begin{aligned} x_p[t+1] &= A_p x_p[t] + B_p u[t] \\ y[t] &= C_p x_p[t] \end{aligned}$$

Following usual notations, the plant state at the t-th time instant is given by the vector $x_p[t]$. Similarly, $y[t]$ defines the output and $u[t]$ defines the control input at the t-th time instant. The matrices A_p, B_p and C_p describe the transition matrix, the input map, and the output map for the plant model respectively. A stabilizing controller $\Gamma = (A_c, B_c, C_c)$ for P senses the plant output y and decides the control action by adjusting the control variables in u. In usual convention, the feedback control law is represented as a linear time invariant (LTI) system of the following form [1]:

$$\begin{aligned} x_c[t+1] &= A_c x_c[t] + B_c y[t] \\ u[t] &= C_c x_c[t] \end{aligned}$$

Here, $x_c[t]$ represents the state of the controller, and A_c, B_c and C_c are the state transition matrix, the input map, and the output map for Γ respectively. With $x = [x_p', x_c']'$, the dynamics of the resulting closed loop $\langle P, \Gamma \rangle$ is as follows:

$$x[t+1] = \begin{bmatrix} A_p & B_p C_c \\ B_c C_p & A_c \end{bmatrix} x[t] \tag{1}$$

If A_1 represents the closed loop dynamic matrix for the normal notion of periodic loop executions, then we have the system dynamics at each sampling time instant t as : $x[t+1] = A_1 x[t]$.

Suppose the loop execution is skipped in some sampling period $[t, t+1]$. This means that between t and $t+1$, the control variables will not change and consequently the control input to the plant will not change. Formally we have, $x_c[t+1] = x_c[t]$ and $u[t+1] = C_c x_c[t+1] = C_c x_c[t]$. The restriction, $x_c[t+1] = x_c[t]$, replaces the transition matrix (A_c) and the input map (B_c) of the controller by the *identity matrix* and *null matrix* respectively, that is, $A_c = I_c$ and $B_c = O$. The situation where the control actuation values remain constant but the plant continues to evolve can be viewed as application of the closed loop dynamic matrix:

$$A_0 = \begin{bmatrix} A_p & B_p C_c \\ O & I_c \end{bmatrix}$$

which is derived from A_1 by substituting $A_c = I_c$ and $B_c = O$.

A switching signal for the loop is a sequence defined over $\{A_1, A_0\}$. We consider such infinite switching signal for control loops to be specified as infinitely repeating finite length (control) *loop execution patterns*. Formally, given a control loop $\langle P_i, \Gamma_i \rangle$, let, σ be a switching signal of the form, $\sigma : \mathbb{N} \mapsto \{A_{0,i}, A_{1,i}\}$, that is, $\sigma(t) = A_{1,i}$ if the control loop is executed at sampling window $[t, t+1]$ and $\sigma(t) = A_{0,i}$ if it is skipped. A loop execution pattern is a finite l-length string $s \in \{A_{0,i}, A_{1,i}\}^*$ such that $\sigma = s^\omega$. For a given loop execution pattern s with the associated infinite length switching signal σ, the closed loop represents a switched system as defined as:

$$x[t+1] = \sigma(t) x[t]$$

In general, when the loop index (i) is inconsequential, or clear from the context, we shall abstract it out for simplicity and also express loop execution patterns as members of $\{0,1\}^*$. For example, according to the loop execution pattern $s = 1^2 0^2 10$, we have, $\forall t \in \mathbb{N},\ x[t+6] = A_0 A_1 A_0 A_0 A_1 A_1 x[t]$ such that the associated infinite length switching signal is given by $\sigma = (A_0 A_1 A_0 A_0 A_1 A_1)^\omega$.

We develop additional methodology for switching between such loop execution patterns so as to adaptively improve the quality of control for a set of controllers sharing an ECU. Moreover, as opposed to controllers that switch between different sampling rates, we use a single sampling rate in all modes of operation of the system, but variable loop execution patterns. This aspect separates our work from existing body of literatures [5–7, 22] that studies multi-rate controllers.

3 PATTERN BASED ADAPTIVE SCHEDULING

The goal of improving control performance through adaptive bandwidth sharing may be attempted in various ways. In our present line of research, driven by practical considerations, we study a structured mechanism for adaptive sharing of bandwidth in the face of input disturbances. Specifically, we begin by defining for each plant, what constitutes a *high* level of disturbance. Intuitively, our goal is to achieve a sharper damping curve for *high* disturbances, and for that we prepare the controller to give the plant more attention (if needed) by elevating it to a different loop execution pattern. In general our methodology could be extended to multiple discrete disturbance thresholds, but at this stage we have experimented with only two levels of disturbances for each plant, namely *high* and *nominal*, and therefore we confine our discussion to only two disturbance levels per plant. A controller will therefore operate under three different modes, each represented by a set of loop execution patterns.

(1) *Nominal Mode:* A controller operates in this mode by default.
(2) *Extremal Mode:* A controller moves to this mode when the respective plant experiences high disturbance.
(3) *Marginal Mode:* A controller moves to this mode when it gives up a fraction of its ECU bandwidth to allow some other controller(s) to move to their extremal modes. This mode is allowed for a controller only at times when its plant is not experiencing high disturbance.

In the proposed methodology, the modes are formally characterized in terms of performance requirement captured in terms of (l, ϵ)-exponential stability [24].

Definition 3.1 (Exponential Stability Criterion). A dynamical system (as represented by Equation (1)) is said to be (l, ϵ)-exponentially stable, with $l \in \mathbb{N}$ and $\epsilon \in (0, 1]$, if $\frac{||x(t+l)||}{||x(t)||} < \epsilon$ for every

$t \in \mathbb{N}$ and $x(t) \in \mathbb{R}^n$, where $||.||$ represents the euclidean norm (2-norm). This implies that the error is reduced by at least a factor of ϵ in every l sampling periods.

Given the settling time criteria for different modes and the input disturbance levels, we compute the corresponding (l, ϵ)-exponential stability constraints. In summary, the inputs to our problem are as follows. A set of n control loops where for each plant, P_i, we are given,

(1) $\langle \Gamma_i, h_i, A_{1,i}, A_{0,i} \rangle$, where the controller Γ_i is designed using a sampling period h_i and $A_{1,i}/A_{0,i}$ are the closed system dynamics in the sampling window where the control loop is executed/skipped.
(2) Worst case execution time (WCET) value w_i for the loop.
(3) A disturbance threshold, η_i.
(4) The exponential damping requirement in the nominal mode represented by the pair, $(l_i, \epsilon_{nom,i})$ which is derived from the specified *desirable settling time*.
(5) The exponential damping requirement in the extremal mode represented by the pair, $(l_i, \epsilon_{ext,i})$ which is derived from the specified *desirable settling time*.
(6) The exponential damping requirement in the marginal mode represented by the pair, $(l_i, \epsilon_{mrg,i})$ which is derived from the specified *marginal settling time*.

In this offline design phase, our goal is to design an ECU-wide strategy for regulating the loop execution patterns of the controllers so as to minimize the control cost, subject to the following constraints:

(1) The patterns are chosen on the basis of the the combined control cost and are subject to the settling time constraints.
(2) Controllers that experience high disturbance must be moved to their extremal modes.
(3) Any controller that does not experience high disturbance must remain in its nominal mode unless compelled to move to its marginal mode to release some of its ECU bandwidth.

The global configurations of the system of controllers sharing an ECU are as follows:

(1) The *settled configuration*. In this configuration, none of the plants are experiencing high disturbance, and therefore, all controllers are in their nominal modes of operation.
(2) The *perturbed configurations*. Given the set $\{P_1, \ldots, P_n\}$ of plants, at any point of time there can be $n_o \leq n$ number of plants experiencing high disturbance. A configuration in which a *non-empty* subset $M \subseteq \{P_1, \ldots, P_n\}$ of plants are experiencing high disturbance is called a perturbed configuration M. For any plant $P_i \in M$, the stabilizing controller should operate in extremal mode.

It may so happen that some configurations are not schedulable, that is, even after moving the remaining controllers to their marginal loop execution patterns we do not have sufficient ECU bandwidth to elevate the controllers experiencing high disturbance to their extremal loop execution patterns. Our analysis will find such infeasible configurations, and the policy of the system will be to remain in its previous configuration in such scenarios.

4 THE ALGORITHMIC FRAMEWORK

In this section, we discuss the algorithmic aspects involved in the different offline computational steps of our adaptive scheduling strategy.

4.1 Obtaining (l, ϵ)-criterion from Settling Time

As evident from earlier discussions, the operational modes for each controller are characterized using (l, ϵ) stability criteria. We derive the (l, ϵ)-pairs for various disturbance levels based on the settling time constraints. A settling time criterion requires the controller to reject all disturbances of level up to d within a time ST. Let the desired operating region of plant P be given by the sphere $\sqrt{x^T \cdot x} \leq \chi$ for some system norm χ. The perturbed system satisfies the settling time criteria within the sphere given by $\sqrt{x^T \cdot x} \leq \chi + d$. Given these as inputs, we compute,

(1) the number of sampling periods $L = \lceil \frac{ST}{h} \rceil$, needed to settle within ST (h is sampling period of P).
(2) the damping factor $f = \frac{\chi}{\chi + d}$.

The pair (L, f) is an exponential stability specification for P for the given disturbance level and settling time requirement. However, for a given (L, f) stability criterion, one should always consider a stricter criterion $(\frac{L}{m}, f^{\frac{1}{m}})$, for some $m > 1$, with a smaller number of sampling periods such that the satisfaction of $(\frac{L}{m}, f^{\frac{1}{m}})$ naturally implies the satisfaction of (L, f). We set $l = \frac{L}{m}$ and $\epsilon = f^{\frac{1}{m}}$ and consider this stricter (l, ϵ)-exponential criterion as our performance requirement. We derive the (l, ϵ)-pairs for the modes of the i-th control loop, from the following inputs:

(1) The *desirable settling time*, $ST_{dsr,i}$, and the *marginal settling time*, $ST_{mrg,i}$, where $ST_{mrg,i} \geq ST_{dsr,i}$.
(2) The disturbance threshold, $\eta_i < \chi_i + d_{max,i}$ to switch to the extremal mode, where $d_{max,i}$ is a design parameter representing the maximum level of disturbance for which the desirable settling time is guaranteed.

In our design we choose the (l, ϵ)-pairs for guaranteeing the respective settling times in the following ranges of disturbance,

(1) $(\chi_i, \eta_i]$: under which we guarantee desirable settling time for the nominal mode, and marginal settling time for the marginal mode.
(2) $(\eta_i, \chi_i + d_{max,i}]$: under which desirable settling time is guaranteed for the extremal mode.

Thus, for a plant P_i,

(1) a controller in nominal mode and extremal mode satisfy $(l_i, \epsilon_{nom,i})$ and $(l_i, \epsilon_{ext,i})$ respectively while ensuring that the settling time $< ST_{dsr,i}$ for the disturbance levels within the respective ranges $(\chi_i, \eta_i]$ and $(\eta_i, \chi_i + d_{max,i}]$.
(2) in marginal mode the controller satisfies $(l_i, \epsilon_{mrg,i})$ stability criterion and ensures settling time $< ST_{mrg,i}$ for the disturbances below η_i.

From the settling time criteria, the computation of the (l, ϵ)-pairs for the i-th loop are as follows. First we compute $L_1 = \lceil ST_{dsr,i}/h_i \rceil$ and $L_2 = \lceil ST_{mrg,i}/h_i \rceil$. Next we choose m_1 and m_2, in such a way so that $l_i = \lceil \frac{L_1}{m_1} \rceil = \lceil \frac{L_2}{m_2} \rceil$. Then we set $\epsilon_{ext,i} = (\frac{\chi_i}{\chi_i + d_{max,i}})^{\frac{1}{m_1}}$, $\epsilon_{nom,i} = (\frac{\chi_i}{\eta_i})^{\frac{1}{m_1}}$ and, $\epsilon_{mrg,i} = (\frac{\chi_i}{\eta_i})^{\frac{1}{m_2}}$. It may also be noted that $\epsilon_{mrg,i} \geq \epsilon_{nom,i} \geq \epsilon_{ext,i}$ since smaller settling times and higher disturbance levels both necessitate higher damping factors (that is, smaller values of ϵ).

4.2 Stability under Loop Skipping

Given a (l, ϵ) performance criterion for any mode (nominal, extremal or marginal) of a system, we can compute the corresponding minimum decay rate (α) to be satisfied at that mode. Following the standard exponential stability definition [14], we set $\alpha = \frac{\log 1/\epsilon}{l}$. The loop skipping behavior of loop execution patterns provide a well defined *drop rate* for the corresponding switching signal. The stability of periodic control loops in the presence of drops has been studied in [3, 25] which provides the following sufficient condition on the rate of loop skipping while guaranteeing exponential stability in terms of a minimum decay rate.

Theorem 4.1. *[25] For a control loop with the associated closed loop matrix A_1 being Schur stable and r being the rate of successful execution of the loop over an infinite horizon, if there exists a Lyapunov function $V(x(t)) = x'(t)Px(t)$ and scalars α_0, α_1 such that*

$$\alpha_1^r \alpha_0^{1-r} > \alpha \tag{2}$$

$$A_1^T P A_1 \leq \alpha_1^{-2} P \tag{3}$$

$$A_0^T P A_0 \leq \alpha_0^{-2} P \tag{4}$$

then the system remains exponentially stable with a decay rate greater than α.

The bound on the execution rate r can be found if the following results hold.

(1) if A_0 is marginally stable, then the closed loop is exponentially stable for $0 < r \leq 1$.
(2) if A_0 is unstable, the closed loop is exponentially stable for $\frac{2\log_e(\alpha)+\log_e(\gamma_0)}{\log_e(\gamma_0)-\log_e(\gamma_1)} < r \leq 1$, where $\gamma_1 = \alpha_1^{-2} = \lambda_{max}^2(A_1)$, $\gamma_0 = \alpha_0^{-2} = \lambda_{max}^2(A_0)$, $\gamma_1 < 1$, $\gamma_0 > \gamma_1$ and $\lambda_{max}(A_j)$ is the maximum eigenvalue of A_j, $j = 0, 1$.

Given a set of n control loops, the above theorem provides the maximum possible rate of loop skipping $\langle(1 - r_1), \ldots, (1 - r_n)\rangle$ which may be allowed in each control loop while respecting some associated stability constraint. As an example, for any loop execution pattern of length $l = 50$ and for $r = 0.9$, the above conditions imply that in any 50 consecutive samples, the controller is allowed to miss at most $50 \times (1 - 0.9) = 5$ computation to remain exponential stable. For a given mode (nominal, extremal or marginal), once we have computed the required minimum decay rate α as discussed earlier, we can compute the corresponding minimum successful execution rate r^{min} at that mode by solving the above mentioned Linear Matrix Inequalities (Equations (2)–(4)) using α as deduced from the given (l, ϵ) criteria, and α_1, α_0 as deduced from A_1, A_0 respectively. We will continue designating this computation using a procedure, `Compute_Min_rates`$(A_0, A_1, (l, \epsilon))$.

4.3 Control Cost of a Loop Execution Pattern

For patterns satisfying a given decay constraint, we employ the notion of linear quadratic cost given by the following cost function as a measure of Quality of Control (QoC).

$$J = \int_0^\infty \left(x^T[t]Qx[t] + u^T[t]Ru[t]\right) dt$$

In the cost function, Q and R are quadratic weight matrices representing the relative importance of the deviation of state valuation $x[t]$ and control effort $u[t]$ respectively. Standard Linear Quadratic Regulator (LQR) based control design minimizes J providing a least cost optimal controller subject to perfectly periodic execution. However, in our case, we consider recurrent patterns instead of perfectly periodic execution. Though the controller designed is optimal for infinite horizon, for control execution with drops, the QoC becomes dependent on the relative locations of loop skipping [20]. The cost of different loop execution patterns satisfying the same rate/decay constraint may potentially be different. Among loop execution patterns satifying the same rate constraint, it can be shown following [20] that the pattern with most uniform drop exhibits the best QoC in terms of LQR cost. We use the above observation as a guiding principle inside our algorithm for schedulable pattern synthesis. The definition of uniformity follows uniform distribution of 0s in

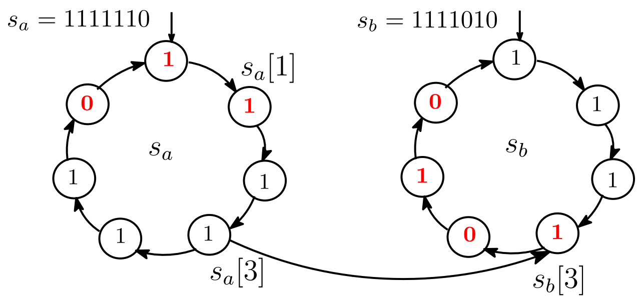


Fig. 1. Stable Transition and Bridges.

binary words known as upper mechanical word [8]. Formally, a binary word is upper mechanical iff there exists $0 \leq \beta \leq 1$ such that the n^{th} letter of the word is given by, $\lceil (n+1).\beta \rceil - \lceil n.\beta \rceil, \forall n \geq 0$ where β represents the fraction of 1-s in the sequence. Hence, among l length loop execution patterns with execution rate r, the optimal pattern s_{opt} is given by

$$\left\lceil (n+1).\frac{q}{l} \right\rceil - \left\lceil n.\frac{q}{l} \right\rceil, \quad \forall n \geq 0, \quad where \quad q = \lceil r \times l \rceil$$

4.4 Stable Transition between Patterns

It is important to guarantee stability when a controller switches between two loop execution patterns. For this purpose, we introduce the notion of *bridges* between recurrent loop execution patterns. Intuitively, a bridge from one loop execution pattern s_a to another pattern s_b is a transition from $s_a[i]$ to $s_b[j]$, such that every l-length subsequence containing this transition must respect the execution rates obtained from the (l, ϵ)-stability criterion of s_a or s_b. The following example illustrates the notion.

Example 4.2. We consider two loop execution patterns, $s_a[0 \cdots 6] = 1111110$ and $s_b[0 \cdots 6] = 1111010$, which follow the execution rates $r_a = 0.85$ and $r_b = 0.71$ obtained by $(7, 0.25)$ and $(7, 0.4)$ exponential stability criteria respectively. The transition from $s_a[1]$ to $s_b[3]$ is not a bridge because the intermediate pattern $s[0..6] = 0111010$ (marked by red color in Figure 1) obtained by appending the substring starting from $s_a[6]$, ending at $s_a[1]$ with the substring starting from $s_b[3]$, ending at $s_b[6]$, has execution rate of 0.57 which is much less than r_a and r_b. On the other hand, the transition from $s_a[3]$ to $s_b[3]$ is a bridge from s_a to s_b since all 7-length patterns using this bridge have execution rates atleast of $min(r_a, r_b)$.

Definition 4.3 (Bridge Between Loop Execution Patterns). Given a recurrent loop execution pattern $s_a[0..l-1]$ satisfying the successful execution rate r_a, and a recurrent loop execution pattern $s_b[0..l-1]$ satisfying the successful execution rate r_b, the pair $(s_a[i], s_b[j])$ is a bridge from s_a to s_b, iff every l-length string, s, obtained by appending a substring from s_a^ω ending at $s_a[i]$ with a substring from s_b^ω starting at $s_b[j]$ satisfy the rate $min(r_a, r_b)$.

For multiple controllers sharing the same ECU, the notion of loop execution patterns is lifted to compatible combinations of loop execution patterns, and the notion of bridges is lifted to compatible combination of bridges.

Definition 4.4 (Compatible Loop Execution Patterns). Given n control loops, a compatible combination of loop execution patterns is an ordered set of patterns $\bar{s} \triangleq \{s_1, \ldots, s_n\}$ which are schedulable in a shared ECU.

Schedulability analysis for patterns is presented in next section. For each configuration of the controllers, we require a compatible set of loop execution patterns. For moving from one configuration to another, we require bridges between the configurations.

Definition 4.5 (Bridge Between Global Configurations). Let $\bar{s}_1$ and $\bar{s}_2$ be the set of compatible loop execution patterns corresponding to two controller configurations. A bridge from one configuration of the controllers to another is an ordered set of bridges between the i-th loop execution pattern in $\bar{s}_1$ and the i-th loop execution pattern in $\bar{s}_2$, $1 \leq i \leq n$.

4.5 Pattern Based Scheduler Automaton

This section outlines the steps in constructing the scheduler automaton that orchestrates the switching of the controllers from one set of loop execution patterns to another in response to disturbances in one or more controllers. Each state of the automaton represents a selected combination of loop execution patterns for n controllers in a specific configuration of the system (that is, corresponding to a disturbance scenario). A transition of the automaton from one state to another is defined by the bridges between two combinations of n loop execution patterns for these two states. In general there may be more than one bridge between two combinations of loop execution patterns, hence a transition of the scheduler automaton is represented by a *set* of bridges. The typical operation of the adaptive scheduler is as follows:

(1) It executes the combination of loop execution patterns corresponding to the settled configuration by default.
(2) When a subset of the plants start experiencing high disturbance, the scheduler selects the next available bridge from the settled configuration to the combination of loop execution patterns corresponding to the new perturbed configuration.
(3) When these plants return to nominal disturbance levels, the scheduler selects the next available bridge to return to the settled configuration.

In general, a different combination of plants may start experiencing high disturbance before the controller has returned to the settled configuration from its previous perturbed configuration. This is particularly significant when a plant which experiences high disturbance in the new perturbed scenario was operating in marginal mode in the previous scenario. Currently our scheduling strategy handles such scenarios in the following way. The scheduler uses the next available bridge to return to the settled configuration (temporarily), and then uses the next available bridge to the configuration which corresponds to the new disturbance scenario. By moving to the new perturbed configuration, it attempts to improve the performance under the changed disturbance scenario.

We shall now elaborate the design steps further. To begin with, for each controller Γ_i corresponding to its three operational modes, we compute the exponential stability criterion $(l_i, \epsilon_{nom,i})$, $(l_i, \epsilon_{ext,i})$ and $(l_i, \epsilon_{mrg,i})$, where $\forall i,\ \epsilon_{mrg,i} \geq \epsilon_{nom,i} \geq \epsilon_{ext,i}$ from the given settling time criteria as outlined in Section 4.1. Next we calculate the following sets of minimum successful execution rates for the patterns in extremal mode, nominal mode and marginal mode respectively to guarantee the exponential stability as described in Section 4.2.

$$r^{min}_{ext,i} = \mathtt{Compute\_Min\_rates}(\mathsf{A}_{0,i}, \mathsf{A}_{1,i}, (\mathtt{l}_i, \epsilon_{\mathsf{ext},i}))$$
$$r^{min}_{nom,i} = \mathtt{Compute\_Min\_rates}(\mathsf{A}_{0,i}, \mathsf{A}_{1,i}, (\mathtt{l}_i, \epsilon_{\mathsf{nom},i}))$$
$$r^{min}_{mrg,i} = \mathtt{Compute\_Min\_rates}(\mathsf{A}_{0,i}, \mathsf{A}_{1,i}, (\mathtt{l}_i, \epsilon_{\mathsf{mrg},i}))$$

For a configuration M, we have a minimum execution rate vector $\boldsymbol{r^{min}_M}$ having the component chosen from above rates for the loops in accordance with their mode of operation in M. For example, in settled configuration, we have $\boldsymbol{r^{min}_{stl}}$=$[r^{min}_{nom,1}, \ldots, r^{min}_{nom,n}]$. For any perturbed configuration $M_i \subseteq \{P_1, \ldots, P_n\}$, we have $\boldsymbol{r^{min}_{M_i}}[j] = r^{min}_{ext,j}$ if $P_j \in M_i$ and $\boldsymbol{r^{min}_{M_i}}[j] = r^{min}_{mrg,j}$ otherwise.

The above rates as computed for a configuration provide the allowable lower bounds on loop executions. We also maintain the maximum possible execution rate for each control loop satisfying its respective lower bound. This however, is constrained by

(1) co-schedulability of all loops given a choice of individual rates under a limited ECU bandwidth.
(2) QoC of the pattern selected for each loop depending on its uniformity.

An algorithm addressing this computational problem of choosing loop execution patterns for a given configuration without compromising the overall QoC is presented in Algorithm 1.

ALGORITHM 1: *Find_Schedulable_Patterns*()

```
Input: Minimum execution rates of a configuration $M$: $\boldsymbol{r^{min}_M}$=$[r^{min}_1, \ldots, r^{min}_n]$, Loop priorities:
    $\mathscr{P} = \{p_1, \ldots, p_n\}$, % bandwidth available: $BW$, Sampling period: $H = \{h_1, \ldots, h_n\}$, WCET:
    $W = \{w_1, \ldots, w_n\}$
Output: Schedulable combination of loop execution pattern for $M$: $\xi_M$
1  $\boldsymbol{r^{max}} = [1, 1, \ldots, 1]$;
2  while $\boldsymbol{r^{max}} \geq \boldsymbol{r^{min}_M}$ do
3      Compute $\langle\delta_1, \ldots, \delta_n\rangle$ such that $\frac{\delta_1}{r^{min}_1} : \ldots : \frac{\delta_n}{r^{min}_n} = p_1 : \ldots : p_n$ and
       $BW - 2\varepsilon \leq \sum_{i=1}^{n}(r^{min}_i + \delta_i)\frac{w_i}{h_i} \leq BW - \varepsilon$ ;     // $\forall i$, find the increment $\delta_i$ over $r^{min}_i$
4      $\boldsymbol{r^{max}}$=$[(r^{min}_1 + \delta_1), \ldots, (r^{min}_n + \delta_n)]$;
5      Find $s_{opt} = \{s_1, s_2, \ldots, s_n\}$, where $s_i$ is an uniform pattern with $\beta_i = \lceil l_i \times \boldsymbol{r^{max}}[i]\rceil / l_i$, $\forall i \in \{1, \ldots, n\}$;
6      if $s_{opt}$ is EDF_Schedulable then
7          $\xi_M = s_{opt}$ ;     // Uniform patterns are schedulable
8          break;
9      end
       // Search For Schedulable Combination of Patterns
10     else
11         $\bar{s}$=Find_Best_Match($W$, $H$, $\boldsymbol{r^{max}}$);
12         if $\bar{s} \neq \Lambda$ then
13             $\xi_M = \bar{s}$ ;     // Schedulable combinations of patterns are found
14             break;
15         end
16         else $BW = BW - \varepsilon$ ;     // Relax the bandwidth utilization
17     end
18 end
   // Pattern is not found for Settled Configuration
19 if $\exists i$ $\boldsymbol{r^{max}}[i] < r^{min}_i$ then
20     $\xi_M = \Lambda$ ;     // Loops are not schedulable at all
21 end
22 return $\xi_M$;
```

The algorithm assumes as input the vector $\boldsymbol{r^{min}_M}$ comprising minimum execution rates for a configuration M, the set $\mathscr{P} = \{p_1, p_2, \ldots, p_n\}$ of loop priorities (with $\sum_i^n p_i = 1$) provided as a design parameter depicting the relative importance of the different control loops along with sampling periods $H = \{h_1, \ldots, h_n\}$, available percentage bandwidth BW and WCETs $W = \{w_1, \ldots, w_n\}$ of the loops. The algorithm executes an iterative behavior (line 2) everytime doing a rate selection for each control loop while respecting the minimum rate constraint $\boldsymbol{r}_{\boldsymbol{M}}^{\boldsymbol{min}}$. It terminates (line 18) on computing the best possible execution rates (stored in $\boldsymbol{r}^{\boldsymbol{max}}$) for the loops while optimizing bandwidth utilization and control cost.

Lines 3–4: In each iteration of the `while` loop, we estimate $\boldsymbol{r}^{\boldsymbol{max}}$ by considering an increment δ_i over the minimum execution rate r_i^{min} (i.e, $\boldsymbol{r}^{\boldsymbol{max}}[i] = r_i^{min} + \delta_i$) of i-th control loop. The selection of the offset value δ_i w.r.t. r_i^{min} is done using the relative priority information while maintaining the ECU bandwidth constraint, as discussed next. The overall bandwidth constraint for all the loops is checked by EDF-schedulability [17] analysis. Let τ be the of n periodic control tasks with implicit-deadlines (i.e. deadline is equal to period) having w_i, h_i as the WCET and period of i-th control loop respectively. A necessary and sufficient condition for τ to be schedulable on a unit-capacity processor, is $U(\tau) \leq 1$, where $U(\tau)$ is the total bandwidth utilization of all the tasks, defined as $U(\tau) = \sum_{i=1}^{n} \frac{w_i}{h_i}$ [2]. In general, with $0 \leq BW \leq 1$ as the available percentage ECU bandwidth, all the control loops are schedulable under their respective minimum execution rates if $\sum_{i=1}^{n} r_i^{min} \frac{w_i}{h_i} \leq BW$. In our attempt of best possible utilization of available bandwidth, we compute an increment δ_i over r_i^{min} for each loop such that $\frac{\delta_1}{r_1^{min}} : \cdots : \frac{\delta_n}{r_n^{min}} = p_1 : \cdots : p_n$ (i.e., in accordance with their relative priority) so that the total bandwidth requirement, $\sum_{i=1}^{n}(r_i^{min} + \delta_i)\frac{w_i}{h_i}$, meets the percentage utilization BW, with a relaxation of $[2\varepsilon, \varepsilon]$, where $0 < \varepsilon < 1$ is a small constant, i.e., $BW - 2\varepsilon \leq \sum_{i=1}^{n}(r_i^{min} + \delta_i)\frac{w_i}{h_i} \leq BW - \varepsilon$. Let $\frac{\delta_i}{\delta_j} = \frac{p_i \times r_i^{min}}{p_j \times r_j^{min}} = k_{j,i}$. Hence the relative increments to be computed become $\{k_{n,1}\delta_n,\ k_{n,2}\delta_n, \ldots, k_{n,n}\delta_n\}$ with $k_{n,n} = 1$. Let $U = \sum_{i=1}^{n} r_i^{min} \frac{w_i}{h_i}$. Incorporating the new quantities, we get $BW - 2\varepsilon \leq \sum_{i=1}^{n} k_{n,i}\delta_n \frac{w_i}{h_i} + U \leq BW - \varepsilon \Rightarrow \delta_n \in [\frac{BW-2\varepsilon-U}{\sum_{i=1}^{n} k_{n,i}\frac{w_i}{h_i}}, \ \frac{BW-\varepsilon-U}{\sum_{i=1}^{n} k_{n,i}\frac{w_i}{h_i}}]$. Maximizing δ_n, we choose, $\delta_n = \frac{BW-\varepsilon-U}{\sum_{i=1}^{n} k_{n,i}\frac{w_i}{h_i}}$ and subsequently find $\delta_i, i = 1, \ldots n-1$ taking constant time for each. Thus this operation of finding the relative bandwidth increments for n control loops is linear in n.

Lines 5–9: For the configuration M, once a suitable value of $\boldsymbol{r}^{\boldsymbol{max}}$ is decided as discussed above, we generate the collection, $\mathbf{s_{opt}}$, of uniform loop execution patterns (the upper mechanical words, see Section 4.3) for n control loops. The uniform patterns are generated with the respective rate choices of $\boldsymbol{r}^{\boldsymbol{max}}$ by setting $\beta_i = \lceil l_i \times \boldsymbol{r}^{\boldsymbol{max}}[i] \rceil / l_i$ for the i-th control loop. This operation is linear in n. If $\mathbf{s_{opt}}$ is found schedulable, it is set as the compatible loop execution pattern for that configuration M and the algorithm terminates there (line 8).

Lines 10–22: Otherwise, if the set of cost optimal uniform patterns $\mathbf{s_{opt}}$ is not schedulable, we execute the procedure `Find_Best_Match` (discussed later) to generate the set $\bar{s}$ of compatible combination of loop execution patterns $\{s_1, s_2, \ldots, s_n\}$ that *minimally* deviates from $\mathbf{s_{opt}}$. In case, when even such a choice is not found, more relaxation in available bandwidth utilization constraint is performed by decrementing BW by an amount of ε. $\boldsymbol{r}^{\boldsymbol{max}}$ is recomputed (in line 3-4) based on this new value of BW and the process iterates. In the next iteration, with smaller increments of δ_i for each control loop, there will be lesser number of control tasks to schedule for all loops inside the actual available bandwidth (original value of BW). Hence the probability of finding a set of compatible loop execution patterns will increase. The iterations are tried out until the minimum execution rates are violated for at least one control loop (i.e., $\exists\ i$ such that $\boldsymbol{r}^{\boldsymbol{max}}[\mathrm{i}] < r_i^{min}$). The number of iterations is upper bounded by $\frac{BW-U}{\varepsilon}$.

Schedulability: The EDF schedulability of the n control loops corresponding to the set, $\{s_1, \ldots, s_n\}$, of loop execution patterns can be determined by preparing the periodic task set from the loop execution patterns and then performing the standard schedulability test over this task set. We create a set, Z_k, of recurrent tasks for each k-th control loop as follows.

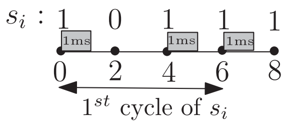
Let l_k denotes the length of s_k, w_k denotes the WCET of loop execution for Γ_k and, h_k denotes the sampling period for Γ_k. For each j, $0 \leq j < l_k$, if $s_k[j]$ is 1, we add the recurrent set of instantiations of $s_k[j]$ into Z_k. Formally these are represented by the set of tasks: $\{\langle k, w_k, v_i, v_i + h_k\rangle \mid v_i = (j + i \times l_k) \times h_k,\ i = 0, 1, \ldots\}$ where w_k is the loop execution time for Γ_k, v_i represents the time at which the $(i+1)^{th}$ instance of $s_k[j]$ is issued, and $v_i + h_k$ is the deadline for that instance. It may be observed that the sequence of tasks issued by the pattern s_k is exactly defined by Z_k. Therefore, the loop execution patterns, $s_1, \ldots, s_n$ for n control loops, are schedulable, iff the set of tasks in $Z = \bigcup_{1 \leq k \leq n} Z_k$ are schedulable. Task set generation from a pattern is linear in pattern length.

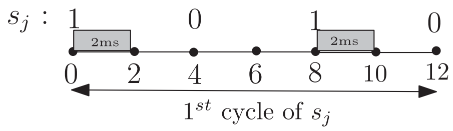
Find_Best_Match: This procedure is called when the cost optimal uniform patterns for all control loops s_{opt} is found to be unschedulable (i.e. line 6 returns false). The procedure searches for a set of compatible loop execution patterns which are minimally nonuniform in terms of distribution of drops. The procedure is based on the definition of a cost function which serves as a measure of the uniformity of a given pattern s satisfying execution rate r. For the pattern s of length l, the drop rate, $(1 - r)$, implies a total of $q = \lceil (1 - r) \times l \rceil$ number of allowable drops such that on average there should be one drop in every sequence of $m = \lceil l/q \rceil$ consecutive loop executions. The non-uniformity of drops in an l length pattern s with execution rate r, is defined point wise as follows. Let $penalty(i, s) = min(0, j - 1)$ if the m length cyclic subsequence starting from $s[i]$ has j number of drops where m is as defined above. Note that in a perfectly uniform pattern, there will be exactly one drop in an m length subsequence thus incurring zero penalty. Hence the overall non-uniformity of s is given by,

$$COST(s) = \sum_{i=1}^{|s|} penalty(i, s)$$

Incurring zero penalty in all subsequences, the cost function evaluates to zero for the most uniform pattern (with rate r) and all its cyclic shifts. For defining the EDF-schedulability constraints in the parlance of the optimization problem our approach is as follows.

Following [2], we define bandwidth requirement, $BR(s, t_1, t_2)$, of the control tasks initiated according to the loop execution pattern s, within the finite time interval $[t_1, t_2)$, as the cumulative execution requirement of those tasks set. For example, let, $s_i = 1011, s_j = 1010, w_i = 1ms, w_j = 2ms, h_i = 2ms, h_j = 4ms$, and both the loops start at $t = 0$. For the time interval $[0, 4)$, $BR(s_i, [0, 4)) = 1$ and $BR(s_j, [0, 4)) = 2$, while $BR(s_i, [4, 8))$=1+1=2 and $BR(s_j, [4, 8))$=0. Therefore





within the time interval $[t_1, t_2)$, the total bandwidth requirement of all the control tasks initiated according to $\{s_1, s_2 \ldots, s_n\}$ is,

$$BR(s_1, \ldots, s_n, [t_1, t_2)) = \sum_{i=1}^{n} BR(s_i, [t_1, t_2))$$

For above example, $BR(s_1, s_2, [0, 4)) = 1 + 2 = 3ms$. A sufficient feasibility condition for schedulability of n control loops getting executed according to the loop execution patterns $\{s_1, s_2, \ldots, s_n\}$ is,

$$BR(s_1, \ldots, s_n, [t_1, t_2)) \leq (t_2 - t_1), \quad \forall t_1, t_2 \leq t_B,\ t_1 < t_2$$

where $t_B = lcm(|s_1| \times h_1, |s_2| \times h_2, \ldots, |s_n| \times h_n)$.

Let $\mathcal{N}_1(s)$ denote the number of 1s in the pattern s. Given the length $\{l_1, \ldots, l_n\}$ and the execution rates $\{r_1, \ldots, r_n\}$ for the n loop execution patterns, the problem of finding the schedulable combination of patterns $\bar{s} = \{s_1, s_2, \ldots, s_n\}$ with maximum possible uniformity, has the following Integer Linear Programming (ILP) formulation,

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n} COST(s_i) \\ \text{subject to} \quad & \mathcal{N}_1(s_i) = \lceil l_i \times r_i \rceil, \quad \forall i = 1, 2, \ldots, n \\ & BR(s_1, \ldots, s_n, [t_1, t_2)) \leq (t_2 - t_1), \forall t_1, t_2 \leq t_B,\ t_1 < t_2 \end{aligned}$$

The function `Find_Best_Match()` formulates and solves this ILP instance and returns the maximum possible uniform and schedulable combination if any feasible solution exists, else it returns NULL. The ILP instance contains $O(n)$ number of constraints of type $\mathcal{N}_1(s_i) = \lceil l_i \times r_i \rceil$ and $O(t_B^2)$ constraints of type $BR(s_1, \ldots, s_n, [t_1, t_2)) \leq (t_2 - t_1)$.

The methodology for synthesizing a table driven scheduler automaton which safely orchestrates mode based scheduling of all the control loops is presented in Algorithm 2. The procedure makes use of Algorithm 1 for computing the compatible loop execution patterns for each configuration.

ALGORITHM 2: *Generate Pattern Based Schedule*

Input: For all controllers, sampling period: $H = \{h_1, \ldots, h_n\}$, minimum execution rates at modes: $\{r_{ext,i}^{min}, r_{nom,i}^{min}, r_{mrg,i}^{min}\}_{i=1}^{n}$,WCET: $W = \{w_1, \ldots, w_n\}$, length of the patterns: $\{l_1, \ldots, l_n\}$, loop priorities: $\mathscr{P} = \{p_1, \ldots, p_n\}$,% bandwidth utilization: BW

Output: Table driven Scheduler Automaton $\mathbb{T}$

```
1  $\boldsymbol{r_{ext,min}} = [r_{ext,1}^{min}, \ldots, r_{ext,n}^{min}]$, $\boldsymbol{r_{nom,min}} = [r_{nom,1}^{min}, \ldots, r_{nom,n}^{min}]$, $\boldsymbol{r_{mrg,min}} = [r_{mrg,1}^{min}, \ldots, r_{mrg,n}^{min}]$;
   // Find Optimal pattern for Settled Configuration
2  $\xi_{stl}$=Find_Schedulable_Patterns($\boldsymbol{r_{nom,min}}$, $\mathscr{P}$, $BW$, $H$, $W$);
   // Pattern is not found for Settled Configuration
3  if $\xi_{stl} = \Lambda$ then
4      $\mathbb{T} = \phi$ ;                                  // The controllers are not schedulable
5      return $\mathbb{T}$;
6  end
   // Find schedulable pattern for each Perturbed Configuration $M_i$
7  for each $M_i \subseteq \{P_1, P_2, \ldots, P_n\}$ do
8      Compute $\boldsymbol{r_{M_i}^{min}}$ ;        // set minimal execution rates according to mode definitions
9      $\xi_{M_i}$=Find_Schedulable_Patterns($\boldsymbol{r_{M_i}^{min}}$, $\mathscr{P}$, $BW$, $H$, $W$);
       // Pattern is not found for this Perturbed Configuration
10     if $\xi_{M_i} = \Lambda$ then
11         $\mathbb{T}[i] = \phi$ ;
12     end
       // Find compatible bridges between $\xi_{M_i}$ and $\xi_{stl}$
13     else
14         $[List_1, List_2]$ = Find_Bridge($\xi_{stl}$, $\xi_{M_i}$) ;   // Find bridges between combinations of patterns
15         $In_{\xi_{stl},\xi_{M_i}} = List_1$, $Out_{\xi_{stl},\xi_{M_i}} = List_2$;
           // Store into the table $\mathbb{T}$
16         $cell(\xi_{stl}, \xi_{M_i}) = (\xi_{stl}, \xi_{M_i}, In_{\xi_{stl},\xi_{M_i}}, Out_{\xi_{stl},\xi_{M_i}})$;
17         $\mathbb{T}[i] = [cell(\xi_{stl}, \xi_{M_i})]$ ;
18     end
19 end
20 return $\mathbb{T}$;
```

Table 1. System Parameters

Automotive Cases Parameters						
ID	System	h_i(ms)	w_i (ms)	$ST_{dsr,i}$ (sec)	$ST_{mrg,i}$ (sec)	l_i
1	CC	40	25	1.2	2.4	15
2	SC	20	6	0.75	1.4	10
3	MS	100	20	2	3.5	7
Double Integrator Parameters						
1	DI	10	5	0.4	0.7	14
2	DI	20	12.5	0.5	0.8	7

Line 1: The vectors $\boldsymbol{r_{ext,min}}$, $\boldsymbol{r_{nom,min}}$ and $\boldsymbol{r_{mrg,min}}$ are initialized with the minimum execution rates of n loops at extemal, nominal and marginal modes respectively. **Line 2–6:** We first compute the set of compatible loop execution patterns for the settled configuration and if found store it in ξ_{stl}. **Line 7–12:** Next we compute the compatible loop execution patterns ξ_{M_i} for each perturbed configuration, $M_i \subseteq \{P_1, P_2, \ldots, P_n\}$. For this, first we set the minimal execution rates in the rate vector $\boldsymbol{r^{min}_{M_i}}$ according to the mode definitions of M_i as described in third paragraph of Section 4.5. Then Algorithm 1 is invoked for finding the compatible loop execution patterns for M_i. **Line 13–19:** The procedure `Find_Bridge(`ξ_i, ξ_j`)` in line 14, is used to find the bridges between two compatible combinations of loop execution patterns, ξ_i and ξ_j. It returns the list, $List_1$, containing all possible bridges from ξ_i to ξ_j and the list, $List_2$, containing the bridges from ξ_j to ξ_i.

The output of Algorithm 2 is a table structure, $\mathbb{T}$, defined as follows. For two compatible combinations of loop execution patterns, ξ_i and ξ_j, for two configurations, we define two list structures, In_{ξ_i,ξ_j} and Out_{ξ_j,ξ_i}. The list In_{ξ_i,ξ_j} is used to store the possible bridges from ξ_i to ξ_j, and the list Out_{ξ_j,ξ_i} is used to store the possible bridges from ξ_j to ξ_i. We also define a cell structure as $cell(\xi_i, \xi_j) \triangleq (\xi_i, \xi_j, In_{\xi_i,\xi_j}, Out_{\xi_j,\xi_i})$ to store the compatible loop execution patterns ξ_i and ξ_j together with their bridging information. Finally, we define our scheduler automaton as a one dimensional list of $N = 2^n - 1$ cells,

$$\mathbb{T}[1, \ldots N] \triangleq [cell(\xi_{stl}, \xi_{M_1}), cell(\xi_{stl}, \xi_{M_2}), \ldots, cell(\xi_{stl}, \xi_{M_N})]$$

The `for` loop in line 7 iterates $2^n - 1$ times, where each iteration is dominated by the complexity of `Find_Schedulable_Patterns`. The procedure `Find_Bridge` takes $O(nl)$ times to find the bridges between the patterns in ξ_{stl} and ξ_{M_i}, where l is the pattern length.

5 CASE STUDIES AND RESULTS

In this section we illustrate our results with an automotive case study comprising a cruise control system (CC), a car suspension control system (SC), and a DC motor speed control (MS) system. The plant models for CC, SC and MS are taken from [21]. We also provide results on a standard control theoretic case study consists of two independent instances of double integrator circuits. Our choice of the various plant parameters are shown in Table 1. Given these parameter choices, we find suitable values of the constants (m_1, m_2) using the method outlined in Section 4.1. In each case study, we synthesize LQR based controllers and consider scheduling the control loops on single ECUs. All the simulations have been performed using MATLAB version R2015b, running on 64-bit Ubuntu 14.04 in a 3.10 GHz Intel Core-i5 machine with 4 GB of memory. The ILP problem has been solved using the MATLAB built-in function `intlinprog`.

5.1 Case Study 1: Automotive Examples

The linearized third order CC system regulates the vehicle speed at a reference level by adjusting the engine throttle angle, the control input u. The dynamic matrices are as follows,

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6.0476 & -5.2856 & -0.238 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 0 \\ 2.4767 \end{bmatrix}, \ C = [1 \ \ 0 \ \ 0]$$

The SC system has four state variables representing the position, velocity of the car and position, velocity of the suspension mass respectively. The control input is the force applied to the body by the suspension system. The dynamic matrices for this system are,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -8 & -4 & 8 & 4 \\ 0 & 0 & 0 & 1 \\ 80 & 40 & -160 & -60 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 80 \\ 20 \\ -1120 \end{bmatrix}, \ C = [1 \ \ 0 \ \ 0 \ \ 0]$$

The second order MS system controls the rotational speed of the motor by adjusting the motor terminal voltage. The state variables represent the rotational speed and the armature current respectively. The dynamic matrices are,

$$A = \begin{bmatrix} -10 & 1 \\ -0.02 & -2 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \ C = [1 \ \ 0]$$

For CC, the desired speed limit is considered as 30 km/hr, i.e., $\chi_1 = 30$. We set the disturbance threshold η_1 to the speed limit of 50 km/hr. The damping factors, $\epsilon_{nom,1}$ and $\epsilon_{mrg,1}$ are determined for the nominal range of disturbance, $(\chi_1, \eta_1]$, using the method outlined in Section 4.1. Similarly, the damping factor, $\epsilon_{ext,1}$ is computed for the extremal disturbance range $(\eta_1, \chi_1 + d_{max,1}]$, where $d_{max,1}$ is the maximum disturbance level for which we guarantee the desired settling time $ST_{dsr,1}$. The value of $d_{max,1}$ is chosen such that $\chi_1 + d_{max,1} = 160$km/hr. For each of $(l_1, \epsilon_{ext,1})$, $(l_1, \epsilon_{nom,1})$ and $(l_1, \epsilon_{mrg,1})$ we calculate the corresponding minimum execution rates $r^{min}_{ext,1} = 0.66$, $r^{min}_{nom,1} = 0.55$ and $r^{min}_{mrg,1} = 0.5$ for extremal, nominal and marginal mode respectively using the method discussed in Section 4.2. Similarly, for SC, we find the minimum execution rates for all these three modes depending on the reference value of the car position as $\chi_2 = 0.02$ m, disturbance threshold of $\eta_2 = 0.5$ m and $d_{max,1} = 4.98$. For MS, we calculate the minimum execution rates for the modes corresponding to the reference value of rotational speed as $\chi_3 = 0.5$ rad/sec, disturbance threshold as $\eta_3 = 1$ rad/sec and $d_{max,1} = 4.5$.

With these design criteria, we apply Algorithm 2 to generate the table driven scheduler automation. It first computes the optimal compatible loop execution patterns ξ_{stl} for the settled configuration and then ξ_i for the different perturbed configurations $M_i \subseteq \{P_1, P_2, P_3\}$. The execution time and other relevant statistical outputs obtained from Algorithm 2 are reported in Table 2. First row of Table 2 shows the output for settled configuration (null set denoting absence of disturbance in all plants) while rest of the rows reflect the same for different perturbed configurations. Column 2 shows the minimum execution rates, $\boldsymbol{r}^{\boldsymbol{min}}_{\boldsymbol{M}_1}$, of the plants, needed for that configuration M. For all these cases, the maximum possible execution rates, $\boldsymbol{r}^{\boldsymbol{max}}$ as derived using Algorithm 2 is shown in Column 3. The % bandwidth utilization obtained by executing the control loops according to $\boldsymbol{r}^{\boldsymbol{max}}$ is reported in Column 4. Since the complexity of Algorithm 2 is mainly dominated by Algorithm 1 for generating the loop execution patterns, we report the time taken by Algorithm 1 separately for each configuration in Column 5. Note that, for the perturbed configuration $M = \{P_1\}$, i.e., when CC experiences the disturbance, the maximum execution rate, $\boldsymbol{r}^{\boldsymbol{max}}[1]$, for it has been increased

Table 2. Results for Various Configurations - Case Study 1

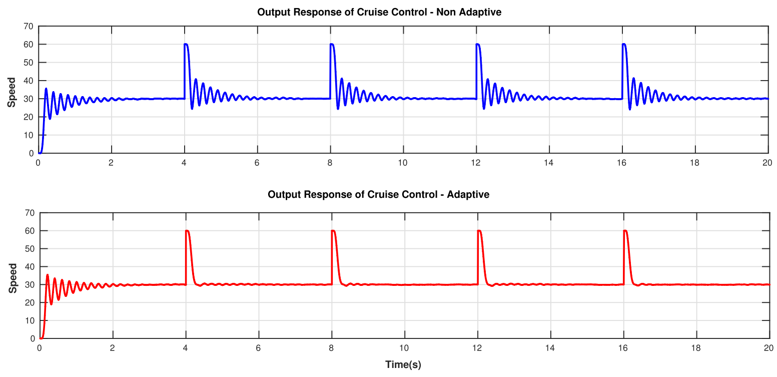
M	r^{min}_{M}	r^{max}	Max. Util.	Time (sec)
Λ	[0.55 0.6, 0.53]	[0.6, 0.64, 0.55]	80.6%	5.4
$\{P_1\}$	[0.66 0.53, 0.5]	[0.77, 0.6, 0.53]	89.5%	3.15
$\{P_2\}$	[0.5, 0.69, 0.5]	[0.5, 0.7, 0.5]	76.6%	5.33
$\{P_3\}$	[0.5, 0.532, 0.6]	[0.6, 0.61, 0.71]	82.3%	5.99
$\{P_1, P_3\}$	[0.66, 0.532, 0.62]	[0.8,0.58,0.70]	93.3%	2.13
$\{P_2, P_3\}$	[0.5, 0.69, 0.6]	[0.51, 0.7, 0.63]	79.4%	3.42
$\{P_1, P_2\}$	[0.66 , 0.69, 0.5]	Not Schedulable	–	4.38
$\{P_1, P_2, P_3\}$	[0.66 , 0.69, 0.6]	Not Schedulable	–	4.02

to 0.77 (see Row-2,Col-3) from the value it had in settled configuration, i.e, 0.6 (see Row-1,Col-3). For the perturbed configurations $\{P_1, P_2\}$ and $\{P_1, P_2, P_3\}$ no schedulable solution is found.

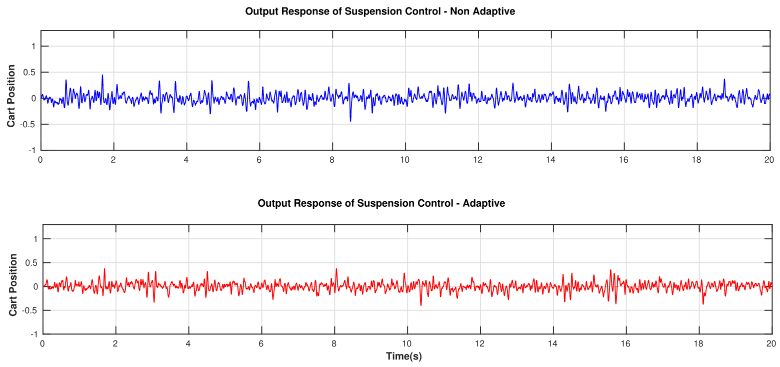
The performance of the adaptive schedules obtained by Algorithm 2 is described next. Here we mainly discuss the situation when disturbance comes into the system CC only. We run a simulation of 20 sec. We generate five periodic disturbance signals, essentially spikes arriving with period of 2 *sec* having peak amplitudes value $\{60, 70, 85, 95, 110\}$ km/hr respectively. Each of these disturbance signals perturbs x_1 (vehicle speed) from its desired value $\chi_1 = 30$. Superimposed with it, we consider the Gaussian state noise with a covariance $R = 500 \times (BB')$. Figure 2 shows the output responses of the plants in the presence and absence of adaptive sharing of bandwidths, when CC is affected by the disturbance signal with peak amplitude of 60. The output response curves of all the plants in the non-adaptive case when all controllers execute the same loop execution patterns regardless of the disturbances, are marked by blue color. The response plots in red are for the adaptive case. The benefit of the adaptive approach can be appreciated by comparing the damping curve of the plants, specially for cruise control, in situations when it gets affected by the disturbance signal as shown in Figure 2(a). In this case, Algorithm 2 finds the optimal cost uniform pattern (s_{opt} in line 6 of Algorithm 1) to be schedulable thus resulting in nice damping behavior for the extremal mode of CC. For SC and MS only Gaussian state noise have been injected without any disturbance spikes. Since SC is a dynamically faster plant w.r.t. MS we find the response to be more noisy. When CC shifts to the extremal mode, SC and MS shift to marginal mode schedules as dictated by the table driven scheduler. Note that for both MS and SC, the adaptive case performs almost similar to non-adaptive one (see Figure 2(b) and Figure 2(c)) since the marginal mode patterns for them generated by Algorithm 2 have execution rates r^{max} very near to that of respective nominal modes.

The choice of the disturbance threshold separating nominal and high disturbance levels is an important design parameter. A high threshold restricts frequent transitions to the extremal modes, thereby curtailing the potential benefits of adaptive scheduling. A low threshold may have the side effect of relegating some controllers to their marginal modes, possibly leading to overall degradation in control performance. In order to analyze the effect of such thresholds on overall utilization and control performance, we have synthesized the table driven schedulers assuming four different choices of the disturbance threshold η_1, namely, the speed limit of $\{50, 70, 80, 90\}$ km/hr. In all cases, we run the simulation for 20 sec and calculate the percentage gain for the adaptive setting over and above the non-adaptive case using $\%Gain = \frac{C_{NA} - C_A}{C_{NA}}$ where C_{NA} and C_A are the quadratic costs of the non-adaptive and adaptive scheduling schemes computed as follows.

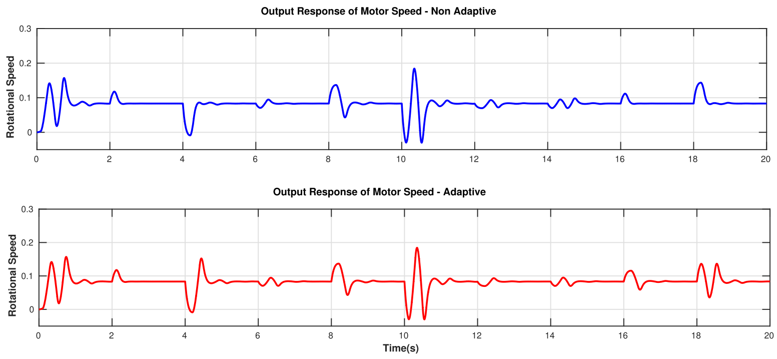
Note that in the settled configuration our adaptive and non-adaptive set of schedules are exactly same. The benefit of adaptive scheduling is manifested in the perturbed configuration. It is therefore well justified to evaluate the control cost over the windows of time when high disturbance is



(a) Cruise Control System



(b) Suspension Control System



(c) Motor Speed System

Fig. 2. Adaptive vs. Non-Adaptive for $M_1 = \{P_1\}$.

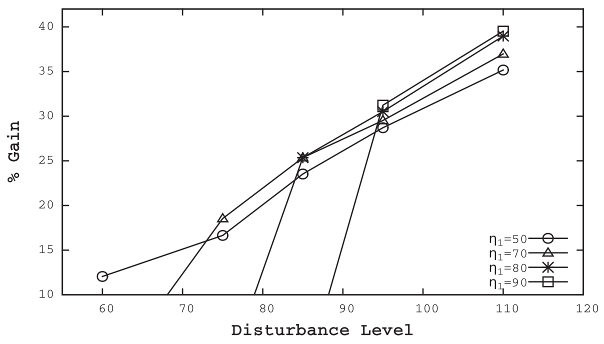


Fig. 3. Improvement in Control Cost.

experienced in one or more plants until the time when the system returns to the settled configuration. In our setup, such windows are observed to be of length 50 consecutive sampling periods. Inside the simulation horizon, we compute C_{NA} as well as C_A by accumulating the quadratic cost for such windows over 50 sampling periods starting from the time instances when disturbance is sensed. Figure 3 shows the percentage gain in quadratic cost when we choose the adaptive approach over the non-adaptive one. Each such curve shows the percentage gains under the five different disturbance signals having peak amplitudes $\{60, 70, 85, 95, 110\}$ as discussed earlier.

The key observations from these curves are:

(1) Lower disturbance thresholds allow more adaptive scheduling and increase the gain in control performance.
(2) For a given choice of the disturbance threshold the percentage gain increases with amplitude of disturbance encountered, but this gain seems to saturate for higher amplitude disturbances.

We have considered the sampling periods for CC, SC and MS to be 40 ms, 20 ms and 100 ms respectively. If we consider our scheme for adaptive scheduling but in a multi rate settings like [5, 7] without the generalization of recurrent patterns, then the choice of sampling periods for the nominal modes become 60 ms, 30 ms and 100 ms respectively. This is because we are now considering only 'all 1' patterns due to which the former choice of periods remain unschedulable inside identical bandwidth budget. Faced with a disturbance scenario, the controller for CC jumps to the sampling period of 40 ms (extremal mode), whereas for compensation SC and MS relinquishes their bandwidth by changing their periods to 50 ms and 120 ms respectively. With $\eta = 50$, we run the simulation and compute the control cost under afore mentioned five disturbance signals (as in Figure 3) and identical simulation settings. Across different disturbance scenarios, the *%Gain* in terms of control cost for our scheme is 20% to 35% w.r.t. the multi-rate setting. This is because, in our scheme, the choice of sampling periods remain same across modes and thus it less increases the overall convex quadratic cost [5, 7] as compared to the multi rate setting.

5.2 Case Study 2: Double Integrator Circuits

We consider two independent instances of double integrator plants with the following discrete time dynamics,

$$x_p[t+1] = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} x_p[t] + \begin{bmatrix} -1 \\ 0.5 \end{bmatrix} u[t]$$
$$y[t] = [0 \quad 1]\ x_p[t]$$

Table 3. Results for Various Configurations - Case Study 2

M	r_M^{min}	r^{max}	Max. Util.	Time (sec)
Λ	$[0.587 \quad 0.573]$	$[0.642,\ 0.70]$	75.9%	4.67
$\{P_1\}$	$[0.692 \quad 0.502]$	$[0.77,\ 0.57]$	78.4%	3.15
$\{P_2\}$	$[0.519,\ 0.687]$	$[0.56,\ 0.849]$	82.6%	3.06
$\{P_1, P_2\}$	$[0.692\ ,\ 0.687]$	Not Schedulable	–	5.14

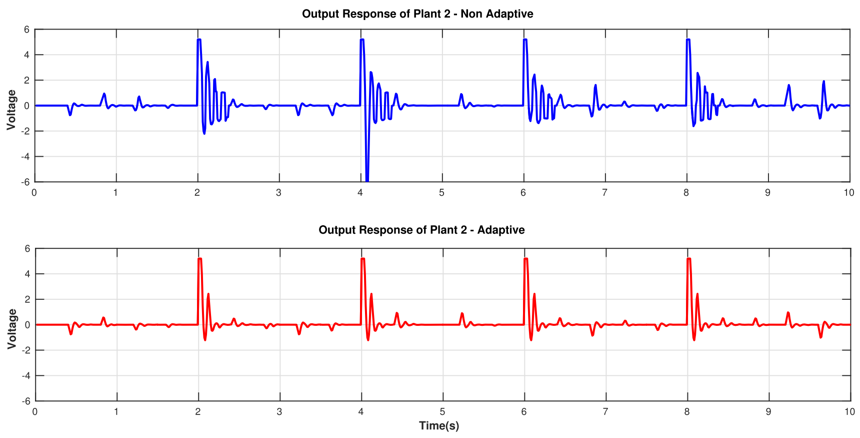
For these double integrator plants, the desired operating regions are considered as spheres with radius $\chi_1 = \chi_2 = 0.05V$. We set the disturbance threshold η_i to $0.5V$, $d_{max,i}$ to $5.45V$, $i = 1, 2$ and based on it we compute the minimum execution rates for all the operating modes. Using this design criteria, Algorithm 2 generates the scheduler automaton and others relevant data which are reported in Table 3. To evaluate the performance of the adaptive approach we run a simulation of 10 sec where we consider five periodic disturbance signals with period of 2 sec, and peak amplitudes of $\{1.2V,\ 2.2V,\ 3.2V,\ 4.2V,\ 5.2V\}$ respectively together with a Gaussian state noise having a covariance $R = 0.005 * (BB')$. Figure 4 shows the output responses of the plants for the perturbed configuration $M_2 = \{P_2\}$. The benefit of the adaptive approach is clearly understandable specially by comparing the damping curves of second plant. For this system, we computed the control cost for the six different disturbance signals and disturbance thresholds $\{0.5V, 1V, 1.5V, 2V, 2.5V, 3V\}$. When compared to non-adaptive scheduling (similar to case study 1), the percentage gain was found to be the range [16-50]%.

6 RELATED WORK AND DISCUSSIONS

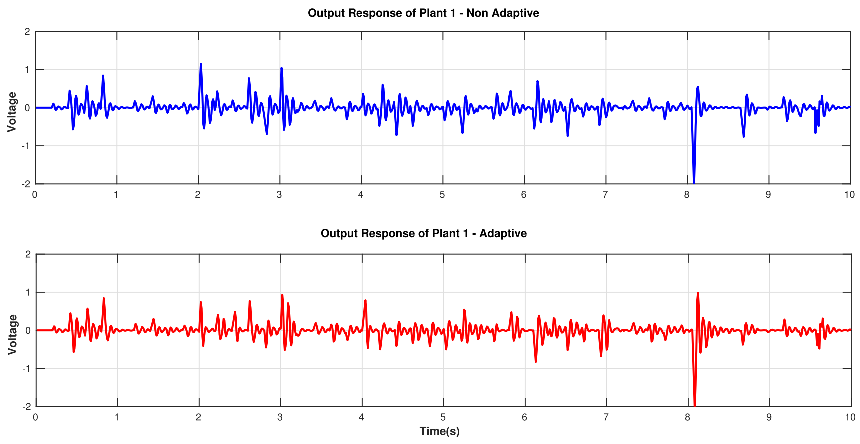
Adaptive control design and application of adaptive control techniques to control architecture co-design and scheduling is a well established field of research. The notion of adaptiveness is primarily implemented using two approaches. One approach is to adaptively alter the sampling rate of controllers in order to improve upon control performance and robustness metrics as demanded by the environment [5–7, 22]. The second approach emphasizes on striving to improve performance metrics by adaptively altering the control law so that any extra bandwidth available at the runtime can be used for computing a better control actuation [9, 10, 13].

The approach of [6, 7] is to activate an outer feedback scheduling loop after a pre-specified interval (the feedback scheduling period), observe the current plant state, estimate the noise, and assign sampling rates to each control loop so that the total control cost is optimized over the feedback scheduling period. While the objective is similar in the present work, i.e. to retain control performance under plant disturbances, our solution method does not change sampling periods as done in [6, 7]. Instead, for the disturbed plants, our offline scheduler switches to loop execution patterns which reject the disturbance more efficiently. Our method has two benefits, a) one need not assume that a control loop can choose from a continuum of sampling periods, which is often not realistic due to restrictions imposed by underlying execution platform, b) the cost of online optimal sampling period computation is done away with thus trading off cost optimality in favor of offline design of low-overhead schedulers.

Another direction of adaptive regulation of sampling rates proposed in [1, 24] where the authors show that for switched systems, the set of infinite schedules that guarantee (l, ϵ)-exponential stability are ω-regular languages defined over the alphabet of closed loop dynamic matrices. The work reported in [24] also outlines a method for construction of finite state automaton of such languages. However, from a practical perspective, it is not easy to use the generalized automaton prescribed in [24] for several reasons: (1) The size of the automaton grows exponentially with l.



(a) Output of Plant-2



(b) Output of Plant-1

Fig. 4. Adaptive vs. Non-Adaptive (Double Integrator).

For large values of l, the automaton will occupy a significant volume of space which is typically not healthy for an embedded scheduler, (2) though all runs guarantee stability, the control cost (as explained later) is not the same in all runs. In fact, we aim to improve the quality of control by switching to a more appropriate loop execution pattern in the face of a disturbance, (3) for adaptive sharing of ECU bandwidth between controllers, we need to examine the trade-off between control cost and the number of loop executions and choose the appropriate runs.

An earlier effort by [18], addresses the problem of dynamic sampling period assignment for handling the reduction of bandwidth due to transient overloads in real time systems. Essentially, they build on the idea of design time schedulable period assignment proposed in [12]. Design methods for control implementations which are robust against irregularities like packet dropout have been reported in [16, 19, 23]. Our idea of loop execution patterns is a formalization of finite length recurrent actuation patterns where we apply it in the context of offline scheduler synthesis for switching among such patterns and the switching is triggered by disturbance levels while ensuring schedulability and stability.

7 CONCLUSION AND FUTURE WORK

Adaptive sharing of ECU bandwidth among controllers is an ambitious as well as desirable objective of control. The benefits of such sharing has been demonstrated by many researchers, but the gap between the control theoretic proofs of concept and the intricacies of mapping the approach to an execution environment has been prominent. Since the approach also marries the ECU bandwidth with the choice of sampling modes, the proposed approach can also be used for choosing platform parameters, such as ECU speed. This direction of research may also yield optimizations in terms of platform dependent attributes such as power. Also as a matter of fact, extending the notion of perturbed global configurations as defined in the present work to a soft real-time setting and admitting deadline misses in a stochastic manner can be an interesting extension of the present work.

REFERENCES

[1] Rajeev Alur and Gera Weiss. 2008. Regular specifications of resource requirements for embedded control software. In *Proc. RTAS.* 159–168.

[2] Sanjoy Baruah and Joël Goossens. 2004. Scheduling real-time tasks: Algorithms and complexity. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* 3 (2004).

[3] Michael S. Branicky, Stephen M. Phillips, and Wei Zhang. 2002. Scheduling and feedback co-design for networked control systems. In *Proc. CDC*, Vol. 2. 1211–1217.

[4] Rosa Castané et al. 2006. Resource management for control tasks based on the transient dynamics of closed-loop systems. In *Proc. ECRTS.* 10–pp.

[5] Anton Cervin, Johan Eker, Bo Bernhardsson, and Karl-Erik Årzén. 2002. Feedback feedforward scheduling of control tasks. *Real-Time Systems* 23, 1–2 (2002), 25–53.

[6] Anton Cervin, Manel Velasco, et al. 2009. Optimal on-line sampling period assignment. *Dept. Autom. Control, Tech. Univ. Catalonia, Barcelona, Spain, Tech. Rep. ESAII-RR-09-04* (2009).

[7] Anton Cervin, Manel Velasco, Pau Martí, and Antonio Camacho. 2011. Optimal online sampling period assignment: theory and experiments. *IEEE Trans. on Control Systems Technology* 19, 4 (2011), 902–910.

[8] Christian Choffrut and Juhani Karhumäki. 1997. Combinatorics of words, Handbook of formal languages. (1997).

[9] Daniele Fontanelli, Luca Greco, and Luigi Palopoli. 2013. Soft real-time scheduling for embedded control systems. *Automatica* 49, 8 (2013), 2330–2338.

[10] Daniele Fontantelli, Luigi Palopoli, and Luca Greco. 2013. Optimal CPU allocation to a set of control tasks with soft real time execution constraints. In *Proc. Hybrid Systems: Computation and Control.* 233–242.

[11] Simon Fürst and AUTOSAR Spokesperson. 2015. Autosar the next generation–the adaptive platform. *CARS@ EDCC2015* (2015).

[12] MEM Ben Gaid, Arben Cela, Yskandar Hamam, and Cosmin Ionete. 2006. Optimal scheduling of control tasks with state feedback resource allocation. In *2006 American Control Conference.* 6–pp.

[13] Vijay Gupta. 2010. On a control algorithm for time-varying processor availability. In *Proc. HSCC.* 81–90.

[14] Arash Hassibi et al. 1999. Control of asynchronous dynamical systems with rate constraints on events. In *Proc. CDC*, Vol. 2. 1345–1351.

[15] Dan Henriksson and Anton Cervin. 2005. Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In *Proc. CDC.* 4469–4474.

[16] Qiang Ling and Michael D. Lemmon. 2002. Robust performance of soft real-time networked control systems with data dropouts. In *Proc. CDC*, Vol. 2. 1225–1230.

[17] Chung Laung Liu and James W. Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM* 20, 1 (1973), 46–61.

[18] Patrizia Marti et al. 2009. Draco: Efficient resource management for resource-constrained control tasks. *IEEE Trans. Comput.* 58, 1 (2009), 90–105.

[19] Johan Nilsson, Bo Bernhardsson, et al. 1996. Analysis of real-time control systems with time delays. In *Proc. CDC*, Vol. 3. 3173–3172.

[20] Jia Ning, Song YeQiong, and Simonot-Lion Francoise. 2007. Graceful degradation of the quality of control through data drop policy. In *Proc. ECC.* 4324–4331.

[21] Debayan Roy et al. 2016. Multi-objective co-optimization of FlexRay-based distributed control systems. In *Proc. RTAS.* 1–12.

[22] Danbing Seto, John P. Lehoczky, Lui Sha, and Kang G. Shin. 1996. On task schedulability in real-time control systems. In *Proc. RTSS*. 13–21.

[23] Damoon Soudbakhsh, Linh T.X. Phan, et al. 2013. Co-design of control and platform with dropped signals. In *Proc. ICCPS*. 129–140.

[24] Gera Weiss and Rajeev Alur. 2007. Automata based interfaces for control and scheduling. In *Proc. HSCC*. 601–613.

[25] Wei Zhang, Michael S. Branicky, and Stephen M. Phillips. 2001. Stability of networked control systems. *IEEE Control Systems* 21, 1 (2001), 84–99.

Received April 2017; revised June 2017; accepted June 2017