Co-design of Control and Platform with Dropped Signals*

Damoon Soudbakhsh[1] Linh T.X. Phan[2] Oleg Sokolsky[2]
Insup Lee[2] Anuradha Annaswamy[1]
[1] *Department of Mechanical Engineering, Massachusetts Institute of Technology*
email: {damoon, aanna}@mit.edu
[2] *Department of Computer and Information Sciences, University of Pennsylvania*
{linhphan,sokolsky,lee}@cis.upenn.edu

ABSTRACT

This paper examines a co-design of control and platform in the presence of dropped signals. In a cyber-physical system, due to increasing complexities such as the simultaneous control of several applications, limited resources, and complex platform architectures, some of the signals transmitted may often be dropped. In this paper, we address the challenges that arise both from the control design and the platform design point of view. A dynamic model is proposed that accommodates these drops, and a suitable switching control design is proposed. A Multiple Lyapunov function based approach is used to guarantee the stability of the system with the switching controller. We then present a method for optimizing the amount of platform resource required to ensure stability of the control systems via a buffer control mechanism that exploits the ability to drop signals of the control system and an associated analysis of the drop bound. The results are demonstrated using a case study of a co-designed lane keeping control system in the presence of dropped signals.

1 Introduction

Implementation issues in controller design have lately received a lot of attention in the context of cyber-physical systems design [28]. Embedded control systems typically consist of several control loops, with different parts of each control application being mapped onto different processors that communicate over one or more communication buses. With increasing complexity in the embedded systems, the gap between high-level control models and their actual implementations inevitably widen. Control engineers are typically concerned with analyzing and simulating controllers based on well-defined models of both the plant and the controller being designed. During this process, the control design process does not include implementation aspects such as the computational time or delays that enter the picture due to shared resources and schedules with varying protocols. Once the control design is complete, has been analyzed and simulated, it is the task of the embedded systems engineer to come up with software implementations (e.g., in C) of the different control blocks (e.g., from MATLAB specifications) and implement the software on a hardware architecture or platform. Here too, the constraints and needs of the control systems are often neglected while carrying out the software design and implementations.

The main problem that can be attributed to the gap between the control models and their implication is the delay between the signal sensed at the plant output and the signal sent to the control actuator. In many of the typical approaches used on the control side, a worst case of end-to-end sample delays is assumed and suitable control laws are designed based on statistical properties or upper-bounds of delays. This assumption introduces a strict constraint on the platform design, where the platform needs to guarantee that all samples must meet the delay bounds. However, the actual delay that a control system experiences on a platform varies with time and can be significantly different from and smaller than the worst-case delay. As a result, designing the platform under the worst-case assumption often wastes a lot of resource.

In this paper, we propose a co-design of control and platform that allows samples to be dropped if their end-to-end delays are more than a delay threshold. Based on both matrix analysis and multiple Lyapunov functions, the control algorithm ensures stability in the presence of a certain number of drops which may occur at any time within the time window. A platform analysis method based on a buffer control mechanism is proposed that ensures that for a given platform, the maximum number of dropped samples does not exceed that is allowed by the control design. As validated using academic examples and numerical studies of a lane-keeping control system of an automotive system, this tight coupling between the platform and control system not only helps optimize the resource use efficiency but also enables more flexibility in the system design.

Related Work: There exist several efforts that aim to close the gap between control and platform in the embedded system domain, for instance, the development of the synchronous language paradigm [4], time-triggered languages like Giotto [12] and other related formalisms such as PTIDES [32]. Efforts in the control systems domain in this direction have been the focus of Networked Control Systems (NCS) [14, 31], whose approach has been the study of distributed controllers communicating over a network. Research in NCS has addressed problems such as delays suffered by control samples, dropped samples and jitter, and their impact on control design, stability, and performance. There have also been attempts to close this gap through a systematic co-design of controllers and the platform [7, 15, 16, 19, 23, 26, 27, 29]. Quite often some of these co-designs lead to an inherent switched system whose stability is analyzed using tools such as Average Dwell time approach [13], and common quadratic Lyapunov functions [1, 18]. Research in NCS in the context of dropped signals has been carried out in [1, 15, 17, 30]. The common approach is to use a sample and hold block and an-

*This work was supported in part by the National Science Foundation grants ECCS-1135815 and ECCS-1135630.

ICCPS2013 '13 Philadelphia, PA USA

alyze system stability [14, 17, 30]. Very few of these papers however have dealt with co-designs. Using a drop frequency metric based on maximum number of drops after each successful signal was presented in [1, 15, 30]. However, these studies result in conservative analysis since each successful sample should compensate for a number of drops.

Several task models within the real-time systems community have also been designed to enable data drops. For instance, the $(m;k)$-firm task model in [11] enables up to $k-m$ jobs to miss their deadlines for any k consecutive jobs, which is a form of dropping. Several analysis methods have been proposed for the $(m;k)$ task model (e.g., [20–22]); however, they focus on the scheduling of a given set of $(m;k)$ tasks instead of analyzing the maximum bound on the number of drops.

Computation of the maximum number of drops in the absence of buffer control was considered in [15]. In theory, the approach in [15] can be extended to capture the buffer control mechanism. However, it may not scale for large systems, since it is based on automata verification, which requires explorations of a complete state space. Our analysis provides a closed-form solution based on a purely algebraic analysis, which can be done efficiently. In addition, our buffer control mechanism can also enable more resource savings, especially for settings where the worst-case delays are much larger than that of the threshold delay.

Contributions: In §3 we present a dynamic model that not only accommodates the underlying dynamics of the plant that is being controlled but also the possibility that the control input used may be subjected to drops; using this model we provide analytical bounds to guarantee the closed-loop stability in presence of multiple drops (§4). We then present a multiple Lyapunov functions approach to prove the stability of the resulting switching controller without and with drops, which further improves the above analysis bounds (§5). Based on the improved bounds, we propose a buffer control mechanism and an associated platform analysis technique to improve the resource required by the control system while ensuring the control stability (§7). Finally, we demonstrate the utility of our co-design method using a case study of a lane keeping system (§8). Our evaluation results show that our method can help save the platform resource needs by an order of magnitude, and it enables a larger design solution space compared to a baseline approach.

2 Definitions

We define Lyapunov Like Functions (LLF), switching systems, and related stability properties, which are used throughout this paper.

The system of interest in this study is a switching system with $\bar{n}$ modes

$$x[k+1] = f_i(x[k]), \quad i = 1, 2, \cdots, \bar{n} \tag{1}$$

We denote the j^{th} instant at which the system switches to mode i as t_i^j.Lyapunov Like Functions (LLF) and the stability result of (1) using LLF are stated below [9]:

DEFINITION 1. *Consider system* (1) *over the time interval* $T_i = (t_i^{j_i^+}, t_\ell^{j_\ell^-})$ *over which* f_i *is active. A positive definite function* $V_i(x[t])$, *with* $V_i(x_0) = 0$ *and* $V_i(x) \succ 0$ *for* $x \neq x_0$, *is called an LLF for system* (1), *equilibrium point* x_0, *and* T_i *if it remains bounded over* T_i, *i.e.,*

$$V_i(x[t_i^{j_i^+}]) \leq h(V_i(x[t_\ell^{j_\ell^-}])) \tag{2}$$

with h being a continuous function, $h(\cdot) : \Re^+ \to \Re^+$ *and* $h(0) = 0$.

THEOREM 1. *If there exist (i) LLFs* V_i*s for* $i = 1, 2, \cdots, \bar{n}$ *of system* (1) *over all intervals* T_i*'s where* f_i *is active, and (ii) their corresponding starting values at active switching times are monotonically non-increasing, i.e.,*

$$V_i(x[t_i^{(j+1)^+}]) \leq V_i(x[t_i^{j+}]), \tag{3}$$

then the system is stable.

PROOF. The proof can be found in [6]. □

REMARK 1. *We note that condition (i) is satisfied for* $h(x) = x$, *when* V_i *is a Lyapunov function for system i. Hence the switching system in* (1) *is stable if Lyapunov functions, rather than LLFs, exist for subsets of the* $\bar{n}$ *systems in* (1), *and Theorem 1 is satisfied.*

The following well known matrix inequality is useful:

LEMMA 2. *Consider a linear map A with spectral radius* $r(A) = \sup|\lambda|, \lambda \in \sigma(A)$, *where* $\sigma(A)$ *is a set containing eigenvalues of A. Then for any* $\rho > r(A), \quad \exists a > 1$ *such that* $\|A^k\| \leq a\rho^k, \forall k \geq 1$.

The following class of switched systems is of interest in this paper:

$$x[k+1] = \begin{cases} A_1 x[k] & k_1 \leq k \leq k_1 + i \\ A_2 x[k] & k_1 + i \leq k \leq k_1 + i + j \end{cases} \tag{4}$$

where A_1 is stable and A_1 and A_2 satisfy the following inequalities:

$$||A_1^k|| \leq a_1 \lambda_1^k, \qquad a_1 > 1, \qquad 0 \leq \lambda_1 < 1 \tag{5}$$

$$||A_2^k|| \leq a_2 \lambda_2^k, \qquad a_2 > 1, \qquad \lambda_2 \geq 0 \tag{6}$$

Throughout this article, we adopt the following to show symmetric matrices

$$\begin{bmatrix} A & * \\ B & D \end{bmatrix} \quad \text{for} \quad \begin{bmatrix} A & B^T \\ B & D \end{bmatrix} \tag{7}$$

3 Statement of the Problem

The specific plant to be controlled is assumed to be linear time-invariant with the following state-space form:

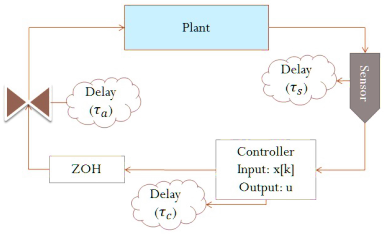


Figure 1: Schematic overview of the control of a plant using a DES.

$$\dot{x}(t) = A_c x(t) + B_c u(t) \tag{8}$$

where, $x(t) \in \Re^p$ and $u(t) \in \Re^q$ are states and inputs, respectively. The goal is to design $u(t)$ so that $x(t)$ tends to zero asymptotically with the closed-loop system remaining bounded. We assume that the plant is periodically sampled with a fixed period T, and define $\tau = \tau_a + \tau_s + \tau_c$ (Fig. 1), where τ_s, τ_c, and τ_a are the processing times for the sensor task, the control computation, and the computed output to be communicated to the actuator. We define τ_{th} as a threshold value for τ and assume that $\tau_{th} < T$. If the delay τ exceeds τ_{th}, we assume that the control computation may arrive too late for it to be effective in controlling the plant. If $\tau < \tau_{th}$, we deem the information useful and use it at time τ_{th}. Denoting $A \stackrel{\text{def}}{=} e^{A_c T}$, $B_1 \stackrel{\text{def}}{=} (\int_0^{T-\tau_{th}} e^{A_c v} dv) B_c$, and $B_2 \stackrel{\text{def}}{=} (\int_{T-\tau_{th}}^{T} e^{A_c v} dv) B_c$ leads to the following discrete-time plant-model [3]

$$x[k+1] = Ax[k] + B_1 u[k] + B_2 u[k-1] \tag{9}$$

when $\tau < \tau_{th}$. This is assumed to be the nominal case, and (9) can be used to design the requisite controller. Suppose a controller of the form

$$u[k] = Kx[k] \tag{10}$$

is used, we obtain the closed-loop system

$$X[k+1] = \begin{bmatrix} A+B_1K & B_2K \\ I & 0 \end{bmatrix} X[k] \stackrel{\text{def}}{=} A_n X[k] \tag{11}$$

where $X[k] = [x[k]^T, x[k-1]^T]^T$. In this paper, it is assumed that a K exists such that A_n is stable. A less restrictive approach when this assumption does not hold can be found in [2]. We refer to this case, when $\tau < \tau_{th}$, as the nominal case. It is quite possible that $\tau > \tau_{th}$, which may be because of the lack of availability of the processor or due to large communication lag between processors, and is assumed to occur infrequently. In such a case, as the information available to the controller is significantly delayed, we simply assume that the message is dropped and use the previous input $u[k-1]$, i.e., employ the zero-order-hold algorithm. If starting at any time k, for m consecutive instants, the delay τ continues to be larger than τ_{th} with $\tau < \tau_{th}$ at $k-1$, then it follows that

$$u[k+j] = u[k-1], \qquad j = 0, 1, \ldots, m-1 \tag{12}$$

That is, the signal has m drops during which no new control input is computed, but rather the old input is used.

In summary, if starting at time k_1,

$$\begin{cases} if \quad \tau < \tau_{th} \quad \text{for} \quad k_1 < k \le k_1 + i & \text{(13a)} \\ \text{if} \quad \tau > \tau_{th} \quad \text{for} \quad k_1 + i \le k \le k_1 + i + j & \text{(13b)} \end{cases}$$

then the corresponding control input is chosen as

$$u[k] = \begin{cases} Kx[k] \quad \text{for} \quad k_1 \le k < k_1 + i & \text{(14a)} \\ u[k_1 + i] \quad \text{for} \quad k_1 + i + 1 \le k \le k_1 + i + j & \text{(14b)} \end{cases}$$

If (14a) is used, the closed-loop system corresponds to (11). If (14b) is used, the closed-loop system becomes more complex and is derived below.

Suppose the drops occur starting at $k = k_1 + i_1$. If $j = 1$ in (14b), then

$$x[k+1] = Ax[k] + Bu[k-1] \tag{15}$$

where $B \stackrel{\text{def}}{=} B_1 + B_2$, and from (14b),

$$u[k-1] = Kx[k-1]. \tag{16}$$

Therefore, with $X[k] = [x[k]^T, x[k-1]^T]^T$, the closed-loop dynamics becomes

$$X[k+1] = \begin{bmatrix} A^2 + AB_1K + BK & AB_2K \\ I & 0 \end{bmatrix} X[k-1] \stackrel{\text{def}}{=} A_m^{(1)} X[k] \tag{17}$$

For a general j number of drops, we have that

$$u[k] = u[k-j] = Kx[k-j].$$

Therefore the underlying dynamics is given by

$$\begin{aligned} X[k+1] = & \begin{bmatrix} A^{j+1} + A^j B_1 K + \sum_{l=0}^{j-1} A^l BK & A^j B_2 K \\ I & 0 \end{bmatrix} X[k-j] \\ \stackrel{\text{def}}{=} & A_m^{(j)} X[k-j] \end{aligned} \tag{18}$$

It is interesting to note that $j = 0$ corresponds to the nominal case, with $A_m^{(0)}$ coinciding with A_n in (18).

In summary, suppose that starting at k, the signal was dropped for the next j_ℓ instants, and i_ℓ instants where it was not dropped, for $\ell = 1, 2, \ldots, p$, over $N = \sum_{\ell=1}^{p} (i_\ell + j_\ell + 1)$ samples. Let m be defined as

$$m \stackrel{\text{def}}{=} \sum_{\ell=1}^{p} j_\ell \qquad n \stackrel{\text{def}}{=} \sum_{\ell=1}^{p} (i_\ell + 1) \tag{19}$$

Then the evolution of the switched system over a time window $[k, k+N]$, $N = m+n$, is given by

$$X[k+N] = A_n^{i_p} A_m^{(j_p)} \cdots A_n^{i_2} A_m^{(j_2)} A_n^{i_1} A_m^{(j_1)} X[k] \tag{20}$$

We note that (20) is a valid description of the underlying system for all $k \ge 0$ that is subjected to $m < m_0$ drops over N samples.

REMARK 2. *We note that in* (20)*, over any interval* $[k, k+N]$*, the sequence* $j_1, i_1, \cdots, j_p, i_p$ *can vary, with* p *varying as well, with the only constraint that* $m \le m_0$ *and the* i*'s such that* $n = N - m \ge n_0$.

We analyze the stability of (20) in the next section.

4 Stability of the Switched System with a maximum of m_0 Drops

We address the stability of (20) in this section using a common Lyapunov function and the matrix inequality in Lemma 2.

THEOREM 3. *System* (20) *with finite drops* $m \le m_0$ *is stable if there are* $n \ge n_0$ *successful signals in any interval of* N *samples.*

PROOF. Noting $\|A_1 A_2\| \le \|A_1\| \cdot \|A_2\|$, we get the following

$$\|X[k+N]\| \le \|A_n^{i_p}\| \cdot \|A_m^{(j_p)}\| \cdots \|A_n^{i_1}\| \cdot \|A_m^{(j_1)}\| \cdot \|X[k]\| \tag{21}$$

The system is stable if $j_1 \ne 0$ and

$$\sum_{\ell=1}^{p} \log(\left\|A_n^{i_\ell}\right\|) + \sum_{\ell=1}^{p} \log(\|A_m^{(j_\ell)}\|) < 0 \tag{22}$$

On the other hand, if $j_1 = 0$, the system is stable if

$$\sum_{\ell=1}^{p} \log(\left\|A_n^{i_\ell}\right\|) + \sum_{\ell=1}^{p-1} \log(\|A_m^{(j_\ell)}\|) < 0 \tag{23}$$

Using $\|A_n^i\| \le a_n \lambda_n^i$ and $\|A_m^{(j)}\| \le \|\bar{A}^j\| \le \bar{a}\bar{\lambda}^j$, it follows that the system is stable if

$$p \log a_n + (n-p) \log \lambda_n + (p-1) \log \bar{a} + m \log \bar{\lambda} < 0 \tag{24}$$

That is, $\|X[k+N]\| \le \|X[k]\|$ if

$$n \ge \frac{m_0 \log(\bar{a}\bar{\lambda}) + (m_0 + 1) \log(a_n)}{|\log \lambda_n|} + m_0. \tag{25}$$

Since the above arguments hold for any k and N such that over $[k, k+N]$, a total of up to m_0 signals are dropped and $n \ge n_0$ signals are not dropped, system (20) is stable. □

EXAMPLE 1. *To examine the validity of the Theorem 3, we consider the following discrete time plant*

$$A = \begin{bmatrix} 1 & 0.4 \\ 3 & 0.3 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}.$$

A control input as in (14) *was used, with* $K = \begin{bmatrix} -0.7195 & -0.2157 \end{bmatrix}$. *With this K, it was found that*

$$A_n = \begin{bmatrix} 0.7842 & 0.3353 & -0.5037 & -0.1510 \\ 2.7841 & 0.2353 & -0.5037 & -0.1510 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

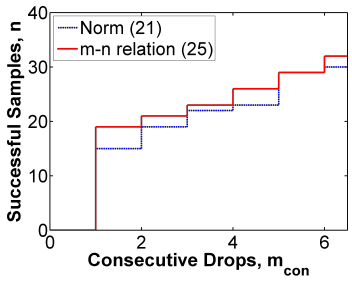


Figure 2: Comparing the methods of Example 1.

Condition (24) *for stability resulted in the dependence of n on m illustrated in Fig. 2. The red line is found by using* (24) *to find the required n after m consecutive dropped signals. The blue dashed line is derived by computing norms of* (21) *iteratively until the product never becomes larger than* 1. *The results in Fig. 2 are with assumption that the packet dropouts in in any interval* $m+n+1$ *consist of only up to m messages that occur consecutively. It can be seen that* (24) *provides a lower bound for n without the need for iterative computation of the norms.*

5 Stability Conditions with m Drops Using a Multiple Lyapunov Function Approach

Although Theorem 3 provides an analytical guarantee for stability of system (20), it is rather restrictive as it uses Lemma 2. Therefore, in this section, we examine an alternate approach for the stability of (20) using Multiple Lyapunov Functions. We define $\Gamma_{mn}^{(ij)}$ as

$$\Gamma_{mn}^{(ij)} \stackrel{\text{def}}{=} A_n^{i_p} A_m^{(j_p)} \cdots A_n^{i_2} A_m^{(j_2)} A_n^{i_1} A_m^{(j_1)} \tag{26}$$

so that the overall system with m drops and n nominal signals with $m \le m_0$ and $n \ge n_0$ can be written as

$$x[k+1] = \Gamma_{mn}^{(ij)} x[k]. \tag{27}$$

Note that $\Gamma_{mn}^{(ij)}$ is not constant and varies with the actual sequence of $j_1, i_1, ..., j_p, i_p$, and p.

ASSUMPTION 4. *System* (27) *with* $\Gamma_{mn}^{(ij)}$ *as defined in* (26) *is stable for any combination of* i_ℓ *and* j_ℓ, $\ell = 1, \cdots, p$, *any p, and* m, n *given by* (19).

We note that the above assumption is not restrictive since according to Theorem 3, for any m, there exists a large enough n that makes $\Gamma_{mn}^{(j)}$ stable.

We start with $k = k_i$, and assume without loss of generality that j_1 drops occur starting at $k_i + 1$ (see Fig. 3). System (27) consists of two modes, the dropped mode, and the stable mode, with the dropped mode occurs from k_i to $k_i + j_1 + 1$ and the stable mode from $k_i + j_1 + 1$ to $k_i + i_1 + j_1 + 1$. Defining

$$k_{2i} = k_{2i-1} + j_1 + 1 \quad \text{and} \quad k_{2i+1} = k_{2i} + N, \quad i = 1, 2, \cdots \tag{28}$$

it follows that the system dynamics is equivalent to the switched system with two modes

$$\begin{cases} z[k_{2i}] = A_m^{(j_1)} z[k_{2i-1}] & \text{(Dropped Mode)} \quad (29a) \\ z[k_{2i+1}] = A_n^{i_p} A_m^{(j_p)} \cdots A_n^{i_2} A_m^{(j_2)} A_n^{i_1} z[k_{2i}] & \text{(Stable Mode)} \quad (29b) \end{cases}$$

We now state and prove the stability of the switched system in (29) in Theorem (5).

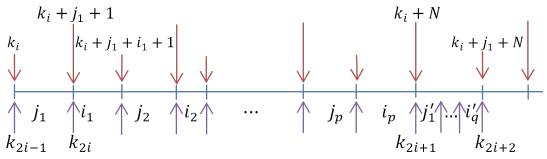


Figure 3: Relation between sequence of (28) and k_i.

THEOREM 5. *The system in* (27) *is stable if there exist symmetric matrix* $P_m \succ 0$ *and scalar* γ *such that the following are satisfied:*

(1)

$$\begin{bmatrix} -\gamma P_m & * \\ P_m A_m^{(j)} & -P_m \end{bmatrix} \prec 0, \qquad j = 0, \ldots j^*, j^* \in [0, m_0] \tag{30}$$

(2) $\Gamma_{mn}^{(ij)T} P_m \Gamma_{mn}^{(ij)} - P_m \prec 0$, *where* $\Gamma_{mn}^{(ij)}$ *is defined in* (26).

PROOF. Theorem 5 is proved by showing that there is an LLF for both modes (29a) and (29b) satisfying conditions (i) and (ii) in Theorem 1 (see for example Fig. 4).

Step 1: Dropped mode (29a): We focus on the interval $[k_{2i-1}, k_{2i-1}+N]$. Defining $V_m[k] = z[k]^T P_m z[k]$, where $P_m \succ 0$ satisfies (30), we obtain that

$$(A_m^{(j_1)T} P_m A_m^{(j_1)}) - \gamma P_m \prec 0 \qquad \text{for any} \quad j \in [0, m_0]. \tag{31}$$

Therefore for $h(V_m) = \gamma V_m$, we obtain that

$$V_m[k_{2i+1}] \le h(V_m[k_{2i-1}]) \qquad i = 1, 2, \cdots \tag{32}$$

Since k_{2i-1} and k_{2i+1} are two consecutive instants that the dropped mode is active, it follows that conditions (i) and (ii) of Theorem 1 are satisfied for the dropped mode.

Step 2: Stable mode (29b): Here, we focus on the interval $[k_{2i}, k_{2i}+N]$. It can be seen that over this interval, there are i_1 nominal signals, followed by j_2 drops, then i_2 nominal signals, and so on until i_p nominal signals ending at k_{2i+1}, followed by the sequence $\{j_1', i_1', \cdots, j_q', i_q'\}$ with the property

$$\sum_{\ell=2}^{p} j_\ell + \sum_{\ell=1}^{q} j_\ell' \le m_0. \tag{33}$$

The underlying dynamics is then given by

$$\begin{cases} z[k_{2i}+1] = A_m^{(0)} z[k_{2i}] & (34a) \\ z[k_{2i+2}] = A_n^{i_q'} A_m^{(j_q')} \cdots A_m^{(j_1')} A_n^{i_p} \cdots A_n^{i_2} A_m^{(j_2)} A_n^{i_1} z[k_{2i}^1] & (34b) \end{cases}$$

where $A_m^{(0)} = A_n$. It therefore follows from (34a) that

$$V_m[k_{2i}+1] \le h(V_m[k_{2i}]) \tag{35}$$

for $h(V_m) \stackrel{\text{def}}{=} V_m$. We note that since there are at most m_0 drops from k_{2i-1} to k_{2i+1} as well as from k_{2i} to k_{2i+2}, and since (33) holds (34b) can be written as

$$z[k_{2i+2}] = \Gamma_{mn} z[k_{2i}], \tag{36}$$

where Γ_{mn} satisfies Assumption 4. Therefore, we obtain from condition (2) of Theorem 5 that

$$V_m[k_{2i+2}] < V_m[k_{2i}] \tag{37}$$

We note that conditions (i) and (ii) of Theorem 1 are satisfied for the stable mode. This proves Theorem 5. □

Theorem 5 suggests the following procedure for determining a stable switching system (27):

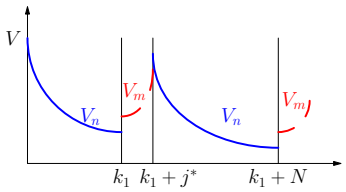


Figure 4: Multiple Lyapunov Functions: Dashed red lines show the *Lyapunov-Like Function* V_m, when the system is in unstable mode, and solid blue lines show LLF of the stable mode, V_n.

1. Find symmetric positive definite matrix P_m and constant γ such that the following LMI are satisfied:

$$\begin{bmatrix} -\gamma P_m & * \\ P_m A_m^{(j)} & -P_m \end{bmatrix} \prec 0 \qquad \text{for} \quad j = 0, \cdots, m \tag{38}$$

2. Evaluate $Q_{mn}^{(ij)} = {\Gamma_{mn}^{(ij)}}^T P_m \Gamma_{mn}^{(ij)} - P_m$ for m drops. If $Q_{mn}^{(ij)}$ is not negative definite, increase n.

REMARK 3. *A less restrictive but computationally more expensive approach than Theorem 5 is to find a common quadratic Lyapunov function (CQLF) for all possible* $\Gamma_{mn}^{(ij)}$. *We note that existence of such CQLF is guaranteed following Theorem 3.*

EXAMPLE 2. *We consider the discrete time system of example 1. Figure 5 shows the maximum number of drops in the window of size N using the results of Theorem 5. Figure 6 compares the results of Theorem 5 with those of Theorem 3, illustrating that the former are much less restrictive compared to the matrix inequalities in Lemma 2. We note that the difference between the results of Theorem 3 presented in Fig. 6 and those presented in Fig. 2 come from the possibility that m drops can occur at any time over an interval N.*

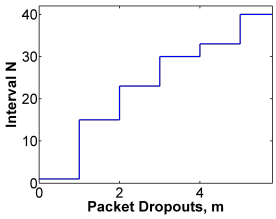


Figure 5: Number of allowed dropped samples in interval N.

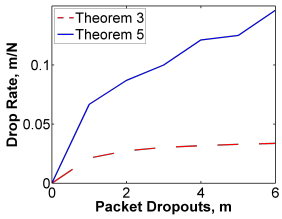


Figure 6: Comparison of Theorem 5 versus Theorem 3.

6 Lane Keeping System

The control task is to keep a vehicle in its lane with radius R (Fig. 7). A one track model of a Ford Taurus was used for this purpose [25]. Dynamics of the vehicle can be described by [24].

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = A_c \begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} + B_c \delta + G\dot{\psi}_{des} \tag{39}$$

where, e_1 is the position error, e_2 is the yaw angle error, and the control input δ is the steering angle at the wheels. We assume the vehicle is traveling on a straight road for which the desired yaw

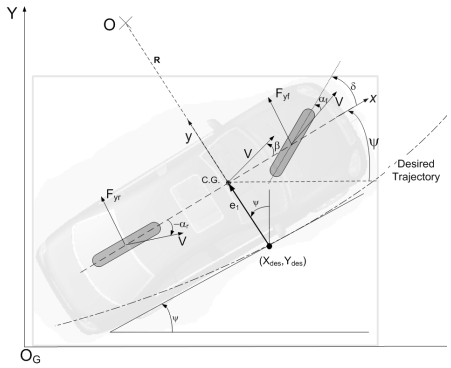


Figure 7: Vehicle model.

rate $\dot{\psi}_{des} = V/R$ is zero. This continuous time model was discretized with a sampling period $T = 50ms$ and delay threshold of $\tau_{th} = 35ms$, resulting in

$$A = \begin{bmatrix} 1.0000 & 0.0447 & 0.1319 & 0.0027 \\ 0 & 0.7969 & 5.0769 & 0.1425 \\ 0 & 0.0003 & 0.9932 & 0.0450 \\ 0 & 0.0102 & -0.2538 & 0.8063 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0.0403 \\ 2.2593 \\ 0.0203 \\ 1.1344 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0.0407 \\ 0.9065 \\ 0.0204 \\ 0.4415 \end{bmatrix}.$$

where A, B_1, and B_2 are the discrete time delayed matrices as defined in (9). The switching control strategy described in §3 was implemented, with $K = -\begin{bmatrix} 1.6258 & 0.2695 & 4.0015 & 0.0454 \end{bmatrix}$. Values m and the corresponding n were computed so that Γ_{mn} satisfied conditions (1) and (2) of Theorem 5. The variations in m with n are plotted in Fig. 8. These results show that as n and therefore N increases, m changes. This information directly provides guidance to the platform designer as it indicates the allowable number of drops over a given time interval.

Figure 9 shows resulting closed-loop system performance of e_1 and e_2 for the case of $N = 6$ and $m = 1$, which illustrates a satisfactory quality of control performance. We note that although MLF resulted in 17 nominal samples for 1 drop, increasing the window size revealed that in the window size of 40 samples, the system can tolerate even the worst case combination of up to 7 drops.

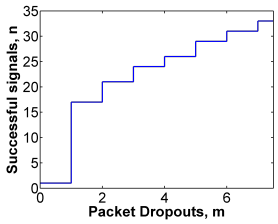


Figure 8: n versus m in the lane keeping system.

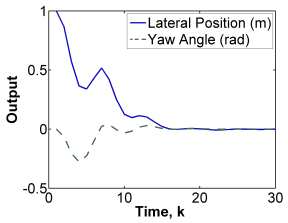


Figure 9: Response of the vehicle with $m = 1$ and $N = 6$.

7 Optimizing Resource via Platform Control

In this section, we describe our approach to optimize platform resource while ensuring the stability of the control system designed in §5. In particular, we will design the platform so that we can guarantee an upper bound m_0 for the number of drops in any interval of N samples for which stability is guaranteed as in Theorem 5. We

assume that the system has been partitioned into a set of tasks that are mapped onto different processing elements (PEs) of a fixed platform architecture (c.f. Fig. 10), which is given a priori. Our goal is to minimize the amount of resource (expressed in terms of processor frequency or communication bandwidth) that the PEs must provide to ensure the control stability.

As was discussed earlier, samples that arrive at the plant after a large delay are not useful: if its end-to-end delay is more than a threshold, τ_{th}, it will be dropped by the control algorithm. We say that a sample is *stale* if its current delay is more than τ_{th}. In the traditional resource dimensioning, the PEs must provide enough resource to ensure there are no stale data. However, such guarantee is not needed for our design, because our control algorithm allows up to (any) m_0 dropped samples over a sliding window of N samples, without loosing stability (c.f. Theorem 5).

Our Approach. To optimize the platform resource, we employ a *buffer control mechanism* that drops samples as soon as they become stale. The resource dimensioning can then be done by selecting the smallest processor frequencies (network bandwidths) for the PEs, such that the maximum number of samples that are dropped by the platform over any N input samples under the above mechanism satisfies the maximum bound m_0 given by the control design. This can be done in an iterative manner: we start with a chosen minimum frequency (bandwidth) for each PE and increase it until the computed drop bound is no more than the bound permitted by the control system.

Before discussing the buffer control mechanism and the drop bound analysis in detail, we first describe the platform architecture.

7.1 System architecture

Fig. 10 shows a typical platform architecture for the control system. It consists of multiple PEs connected via FIFO buffers, where each PE represents a processor (e.g., ECU) or a network (e.g., CAN bus).

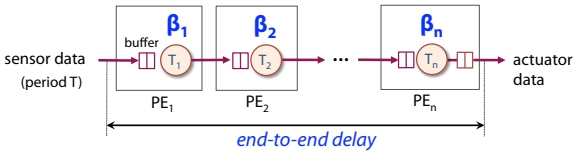


Figure 10: A platform architecture for a control system.

The sensor input data samples, upon arriving at the system, will be processed by the sequence of control tasks running on the PEs. The delay from the instant a sample enters the platform until it is fully processed is called the *end-to-end delay*. The sensor data are sampled at a sampling period of T, and all tasks are data-driven.

We assume that the input buffer of each PE is sufficiently large to avoid buffer overflows.[1] The resource available to the control task at each PE_i is modeled by a pair of *service functions*, $\beta_i = (\beta_i^u, \beta_i^l)$, where $\beta_i^u(\Delta)$ and $\beta_i^l(\Delta)$ denote the maximum and minimum number of samples that can be processed by the PE over *any* time interval of length Δ time units, respectively, for all $\Delta \geq 0$. These functions can be computed based on the PE's operating frequency (network bandwidth) and the worst-case and best-case execution demands of each sample [8].

Note that the architecture shown in Fig. 10 shows only the execution flow of the control system. In a complete setting (e.g., Fig. 14), the system may share the platform resource with other applications; hence, we first compute the service functions, β_i, of the resource available to the control task at each PE (e.g., using the Real-Time Calculus (RTC) technique [8]) and then apply them to the architecture shown above.

[1]The maximum buffer size can be computed using the method in [8].

7.2 Buffer control mechanism

Observe that partially processed samples at intermediate PEs may already become stale, i.e., their delays exceed the threshold τ_{th}. Therefore, it is safe to drop these samples at the intermediate PEs instead of continuing processing them until they are fully processed.

The buffer control mechanism works during run-time at each buffer in the system (see Fig. 11) as follows. When sensor samples enter the first PE, we record their arrival times and use these timestamps to determine their current delays. When a sample arrives at or while waiting in a buffer, if its current delay is more than or equal to the threshold τ_{th}, it will be dropped from the buffer.

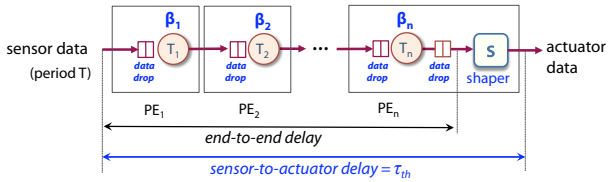


Figure 11: Buffer control mechanism for the platform.

Recall that our control algorithm assumes a delay of τ_{th} for non-drop samples; to enforce this, we implement a *shaper* at the last PE to shape the output samples (see Fig. 11). This shaper reads the fully processed samples, and it holds every sample for exactly $\tau_{th} - d$ time units before sending to the actuator, where d is the end-to-end delay of the sample.[2] Thus, the sensor-to-actuator delay of each sample is always τ_{th}.

The use of buffer mechanism helps improve the resource use efficiency in two ways. First, since some dropped stale samples are permissible, the amount of computation (bandwidth) resource required by the control system is less than that is required to guarantee that the absence of stale data. Second, the amount of resource needed to further process these stale samples can now be saved or used to process other applications.

We note that our choice of the buffer control mechanism is based on its run-time efficiency. However, our analysis can be extended to consider more complex mechanisms, such as one that drops a sample if the sample is predicted to be stale. We assume that the resource overhead required for the buffer control mechanism is negligible (or has already been accounted for when computing the service functions β_i).

7.3 Drop bound analysis under buffer control

We now present a method for computing the maximum number of samples that are dropped under the proposed buffer mechanism (see Fig. 11). Specifically, for any given $N \geq 1$, we will compute m, the maximum number of samples that are dropped over a sliding window of N consecutive input samples.

We first recall the minimum convolution operator and an existing result from [5]. Let $f, g \in \mathbb{R} \to \mathbb{R}$. The minimum convolution of f and g, denoted by $f \otimes g$, is given by

$$(f \otimes g)(\Delta) = \inf_{0 \leq \delta \leq \Delta} \{f(\delta) + g(\Delta - \delta)\}, \forall \Delta \geq 0.$$

THEOREM 6. *Consider a data stream being processed by a platform that consists of a sequence of n PEs, where each PE_i offers a pair of service functions, $\beta_i = (\beta_i^u, \beta_i^l)$, to the stream. Then, the overall resource given by the platform to the stream is given by a pair of service functions, $\beta = (\beta^u, \beta^l)$, where $\beta^u = \beta_1^u \otimes \beta_2^u \otimes \cdots \otimes \beta_n^u$ and $\beta^l = \beta_1^l \otimes \beta_2^l \otimes \cdots \otimes \beta_n^l$.*

[2]Since the buffers implement data drop mechanism, $d \leq \tau_{th}$.

Based on the above result, we will transform the original platform (see Fig. 11) into an equivalent new platform consisting of a single PE, which offers a pair of service functions equal to β (c.f. Theorem 6). As is shown in Fig. 12, this platform employs the proposed buffer control mechanism, where the buffers of the PE implement the data drop mechanism and the output data are passed through a shaper before being sent to the actuator (c.f. §7.2). Since the transformed platform offers the same amount of resource to the sensor data stream and it uses the same data drop mechanism as the original platform does, the maximum number of samples that are dropped and the maximum end-to-end delay of samples that are not dropped in both systems are the same.

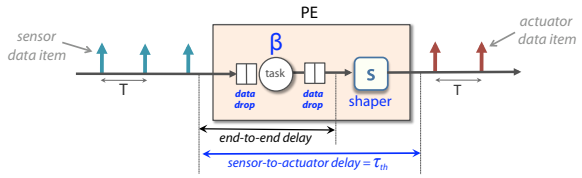


Figure 12: The transformed system that is used for the analysis.

From the above observation, we can compute the drop bound by analyzing the transformed system shown in Fig. 12. Since the shaper does not drop any samples, we only need to consider the parts of the system before the shaper. Without loss of generality, we assume β^u is sub-additive and β^l is super-additive.[3] We first verify a special case in which no samples are dropped, i.e., $m = 0$.

LEMMA 7. *No samples are dropped by the system iff*

$$\forall \Delta \geq 0,\ \alpha^u(\Delta) \leq \beta^l(\Delta + \tau_{th}), \tag{40}$$

where $\alpha^u(\Delta) = \lceil \Delta / T \rceil$ *for all* $\Delta \geq 0$.

PROOF. Since input samples arrive at the system every T time units, the maximum number of samples that arrive over any interval of length Δ is $\alpha^u(\Delta) = \lceil \Delta/T \rceil$, for all $\Delta \geq 0$. In addition, the minimum number of samples that can be processed by the PE over any interval of length Δ is $\beta^l(\Delta)$. From [8], the maximum delay experienced by the input samples is given by

$$d = \sup\{\ \inf\{\delta \mid \delta \geq 0 \wedge \alpha^u(\Delta) \leq \beta^l(\Delta+\delta)\} \mid \Delta \geq 0\}.$$

Hence, $d \leq \tau_{th}$ iff (40) holds. In other words, no samples are dropped iff (40) holds. This proves the lemma. □

We next outline the analysis for the general case, where $m > 0$.

We denote by e_i the ith input sensor sample for all $i \geq 1$, and by e_0 a dummy initial data sample that has a worst-case execution demand of 0 (hence it will not be dropped). Further, $\mathsf{drop}(S)$ denotes the number of samples in S that are dropped, where S is any set of consecutive input samples. Lemma 8 identifies a characteristic of the N samples with the highest number of samples being dropped.

LEMMA 8. *Suppose S is a set with the highest number of dropped samples among all sets of N consecutive input samples. Then, there exists a set S^* of N consecutive samples such that* $\mathsf{drop}(S^*) = \mathsf{drop}(S)$, *where (i) the first sample of S^* is dropped, and (ii) the sample immediately before the first sample of S^* (if any) is not dropped.*

PROOF. Suppose $S = [e_k\, e_{k+1} \cdots e_{k+N-1}]$, where $k \geq 1$. If S satisfies the conditions (i) and (ii), then the lemma trivially holds. Otherwise, there are two cases:

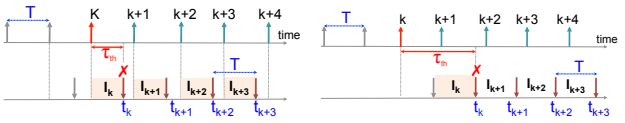
[3] A function f is sub-additive iff $f(\Delta) \leq f(\delta) + f(\Delta - \delta)$ for all $0 \leq \delta \leq \Delta$. By contrast, f is super-additive iff $f(\Delta) \geq f(\delta) + f(\Delta - \delta)$ for all $0 \leq \delta \leq \Delta$.

Case 1: e_k **is not dropped.** Then, there exists a subsequent sample in S that is dropped (since $\mathsf{drop}(S) > 0$). Let e_{k+j} be the first sample in S that is dropped, where $1 \leq j \leq N-1$. Then, all the samples from e_k to e_{k+j-1} are not dropped. This implies $\mathsf{drop}(S_1) = 0$, where $S_1 = [e_k \cdots e_{k+j-1}]$. Let $S^* = [e_{k+j} \cdots e_{k+N-1+j}]$; $S_2 = [e_{k+j} \cdots e_{k+N-1}]$; and $S_3 = [e_{k+N} \cdots e_{k+N-1+j}]$. Then, $S = S_1 \cup S_2$ and $S^* = S_2 \cup S_3$. Thus, $\mathsf{drop}(S^*) = \mathsf{drop}(S_2) + \mathsf{drop}(S_3) \geq \mathsf{drop}(S_2) = \mathsf{drop}(S_2) + \mathsf{drop}(S_1) = \mathsf{drop}(S)$. By the definition of S, we imply $\mathsf{drop}(S^*) = \mathsf{drop}(S)$. Further, by construction, S^* satisfies both conditions (i) and (ii).

Case 2: Both e_k **and** e_{k-1} **are dropped, and** $k > 1$**.** Let e_{k-j-1} be the last sample before e_k that is not dropped, where $1 \leq j \leq k-2$, or $j = k-1$ if no such sample exists. Define $S^* = [e_{k-j} \cdots e_{k-j+N-1}]$. Based on a similar argument as above, we can derive $\mathsf{drop}(S^*) \geq \mathsf{drop}(S)$, and thus $\mathsf{drop}(S^*) \geq \mathsf{drop}(S)$.

Since S^* exists in both cases, the lemma holds. □

Based on Lemma 8, we only need to consider sets of N consecutive samples that begin with a sample that will be dropped and that immediately follow a sample that will not be dropped. Let $S = [e_k\, e_{k+1} \cdots e_{k+N-1}]$, where $k \geq 1$, be any such set. Fig. 13 illustrates the arrival and drop patterns of the samples in S for the two cases, $\tau_{th} \leq T$ and $\tau_{th} > T$.



(a) Critical intervals when $\tau_{th} \leq T$. (b) Critical intervals when $\tau_{th} > T$.

Figure 13: The arrival and drop patterns of the samples in S.

In the figure, each up-arrow labeled by $k+j$ represents the arrival of the input sample e_{k+j}, for all $j \geq 0$. Each down-arrow at time t_{k+j} represents the latest instant by which e_{k+j} must be fully processed, which is also the instant when e_{k+j} is dropped if it has not yet been fully processed. Observe also that e_k is dropped at t_k, i.e., exactly τ_{th} time units after its arrival. Each following sample, e_{k+j}, will be dropped at time t_{k+j}, if it has not been fully processed by the time instant t_{k+j}, where $t_{k+j} = t_{k+j-1} + T$ for all $j \geq 1$.

Definitions. For each sample e_i, the time interval over which either e_i must be fully processed or it will be dropped at the end of the interval is called the *critical interval* of e_i, denoted by I_i. As is shown in Fig. 13, $I_i = (t_i - \tau_{th}, t_i]$, if $\tau_{th} \leq T$, and $I_i = (t_i, t_{i+1}]$, otherwise.

A *service pattern* of the PE is captured by an accumulative function $C(t)$, where $C(t)$ gives the number of samples that can be processed by the PE over the time interval $(0,t]$. By definition, we imply that $C(t)$ is a valid service pattern of the PE iff $\forall t \geq 0, \forall \Delta \geq 0,\ \beta^l(\Delta) \leq C(t+\Delta) - C(t) \leq \beta^u(\Delta)$.

Under a valid service pattern $C(t)$ of the PE, the total number of samples that can be processed over the critical interval I_i is then $C_i = C(t_i) - C(t_i - \tau_{th})$. We call I_i a *zero-service interval* under this service pattern iff $C_i = 0$.

LEMMA 9. *Given any service pattern $C(t)$ of the PE. Let s be the number of zero-service intervals in N critical intervals from I_k to I_{k+N-1} (see Fig. 13) under the service pattern $C(t)$. Then,* $\mathsf{drop}(S) = s$, *if* $\tau_{th} \leq T$, *and* $\mathsf{drop}(S) \leq s$, *otherwise.*

PROOF. **Case** $\tau_{th} \leq T$**:** As is shown in Fig. 13(a), each sample e_i $(i \geq k)$ arrives at the beginning of I_i and must be processed by the end of I_k so as not to be dropped. Further, the last sample before e_i has either been finished or dropped at time t_{i-1}, i.e., before I_i begins. Thus, the resource given to e_i is exactly the resource

available over the interval I_i. Hence, e_i is dropped iff the number of samples that can be processed over I_i is zero, i.e., I_i is a zero-service interval under the service pattern $C(t)$. Hence, the number of dropped samples in S is the number of zero-service intervals.

Case $\tau_{th} > T$: First, since e_k is dropped at the end of I_k, the number of samples that can be processed over the interval I_k is zero. Thus, I_k is a zero-service interval under $C(t)$.

Next, observe that if a critical interval is not a zero-service interval, then no samples are dropped over the interval. This is because at most one sample is required to be fully processed over each l I_i (since only e_i is required to finish by the end of I_i but this sample may have been finished before I_i begins).

Further, any sample e_{i-1}, with $i > k$, is either finished or dropped before I_i begins. Hence, all the resource available over I_i will be given to e_i first and only the remaining resource (if any) is given to the subsequent samples. As a result, e_i is dropped only if the number of samples that can be processed by the PE over I_i is zero, or in other words, I_{k+j} is a zero-service interval under $C(t)$.

From the above, the number of samples in $S\backslash\{e_k\}$ that are dropped is no more than the number of zero-service intervals from I_{k+1} to I_{k+N-1} under the service pattern $C(t)$. Since e_k is dropped and I_k is a zero-service interval, the number of samples in S that are dropped is no more than the number of zero-service intervals from I_k to I_{k+N-1} under the given service pattern. This proves the lemma. □

Based on the above lemma, we can compute an upper bound on the number of dropped samples in S by first constructing a worst-case service pattern, $\widehat{C}(t)$, that results in the maximum number of zero-service intervals; the maximum bound m is the number of zero service intervals in N consecutive intervals of $\widehat{C}(t)$. The idea is to have the PE provide resource over as minimal critical intervals as possible without violating its lower service function β^l, and within each such interval provide as much resource as possible without violating the upper service function β^u. This service pattern confirms to β while resulting in minimum zero-service intervals.

The next lemma gives a formula for a worst-case service pattern $\widehat{C}(t)$ of the PE for the case $\tau_{th} \leq T$. A worst-case service pattern for the case $\tau_{th} > T$ can be constructed in a similar manner; we omit this case due to space constraints.

LEMMA 10. *Suppose $\tau_{th} \leq T$. Let $\widehat{C}(t)$ be a service pattern such that $\widehat{C}(0) = 0$ and for all $i \in \mathbb{N}$:*

- *For all $0 < \delta \leq \tau_{th}$,*

$$\widehat{C}(iT+\delta) = \begin{cases} \widehat{C}(iT), & \text{if} \quad \widehat{C}(iT) \geq \beta^l(iT+\tau_{th}); \\ \min\{\beta^u(iT+\delta), \widehat{C}(iT)+\beta^u(\delta)\}, & \text{otherwise.} \end{cases}$$

- *For all $\tau_{th} < \delta \leq T$,*

$$\widehat{C}(iT+\delta) = \min\{\beta^u(iT+\delta), \widehat{C}(iT+\tau_{th})+\beta^u(\delta-\tau_{th})\}.$$

Then, $\widehat{C}$ is a worst-case service pattern of the PE, i.e., $\widehat{C}$ results in the maximum number of dropped samples over any N consecutive samples, for any $N \geq 1$. Further, the number of zero-service intervals of $\widehat{C}(t)$ for any given N is $\sum_{i=1}^{N}\{1 \mid \widehat{C}(iT) \geq \beta^l(iT+\tau_{th})\}$.

PROOF SKETCH. Consider the zero-offset arrival pattern of the input data stream, i.e., each sample e_i arrives at the time instant $(i-1)T$, for all $i \geq 1$. Then, $I_i = (iT, iT+\tau_{th}]$ is the critical interval of e_i, during which e_i must be fully processed so as not to be dropped. Further, $I'_i = (iT+\tau_{th}, (i+1)T]$ is an interval during which the buffer is always empty, because at the time instant $iT+\tau_{th}$, either e_i has already been fully processed or e_i is dropped. Hence, the amount of resource that is available for the critical intervals is minimum if the amount of resource allocated to all the I'_i is maximum.

Further, recall that the upper service function β^u is sub-additive and the lower service function β^l is super-additive. Hence, we imply from the construction of $\widehat{C}(t)$ that for each critical interval I_i, either (1) I_i is a zero-service interval, if providing no resource during this interval does not violate the lower service function β^l, or (2) the maximum amount of resource permissible by the upper service function β^u is allocated to I_i, otherwise. In addition, the amount of resource provided to all the intervals I'_i according to $\widehat{C}(t)$ is the maximum amount that is permissible by β^u during these intervals, and this amount of resource is wasted (since the buffer is always empty during these intervals).

Based on the above, we can derive that $\widehat{C}(t)$ results in the maximum number of zero-service intervals for the zero-offset arrival pattern. In addition, it can be shown that the zero-offset arrival pattern is a pattern that incurs the highest number of samples being dropped among all the arrival patterns of the input data stream, for all $N \geq 1$. Also, the formula for computing the number of zero-service intervals of $\widehat{C}(t)$ can be derived directly from the construction of $\widehat{C}(t)$. This proves the lemma. Due to space constraints, we omit the details. □

The next corollary follows directly from the above lemma.

COROLLARY 11. *Suppose $\tau_{th} \leq T$. The maximum number of samples that are dropped over any N consecutive samples, for any $N \geq 1$, is the number of zero-service intervals of $\widehat{C}(t)$ for the given N (defined in Lemma 10).*

Finally, as a result of Lemma 9, the drop bound given by Corollary 11 is a tight bound.

8 Case Study

This section presents a case study of the lane keeping system in §6 to demonstrate the utility of our co-design method and its benefits iagainst a baseline method (which is described in §8.1).

8.1 Experimental setup

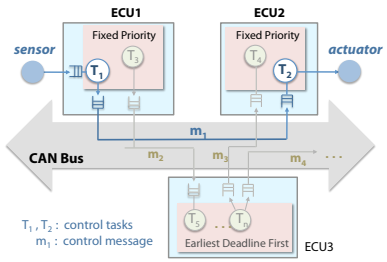
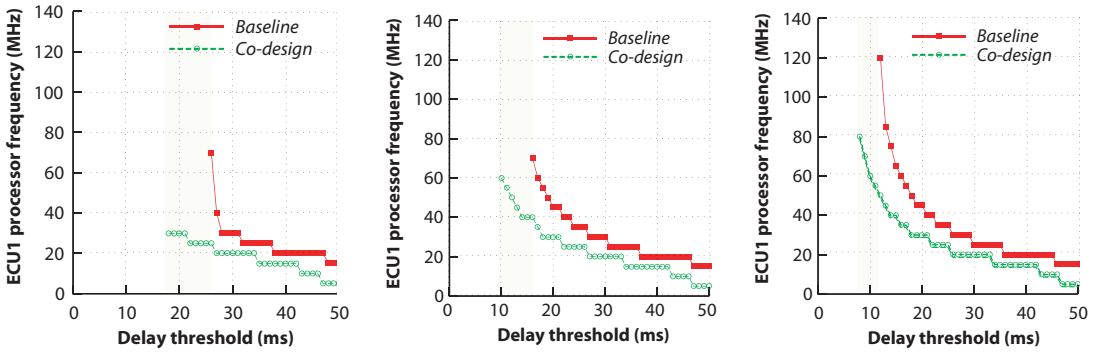


Figure 14: System architecture for the lane keeping system.

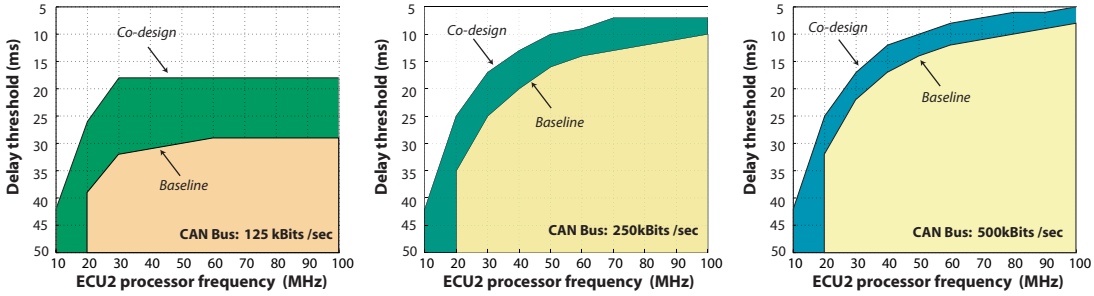
Fig. 14 shows a lane keeping system that is mapped onto a CAN architecture. Each sensor value that arrives from the sensor cluster is first processed by the control task T_1 on ECU1. The processed slip value is then sent to ECU2 via the message m_1. Upon arriving at ECU2, the slip value is used by T_2 to compute the steering angle, which is required by the wheel brake actuator for the wheel steering thus keeping the vehicle in lane. In addition, the platform also executes other applications (tasks T_3 to T_n, and messages m_i, $i \geq 2$).

In our evaluation, the sampling period of the input sensor data is $T = 50$ms. ECU1 and ECU2 employ the preemptive fixed-priority



(a) ECU2 frequency = 30 MHz (b) ECU2 frequency = 60 MHz (c) ECU2 frequency = 90 MHz

Figure 15: ECU1 processor frequency for different delay thresholds.



(a) Low speed bus (125 kBits/s) (b) Medium speed bus (250 kBits/s) (c) High speed bus (500 kBits/s)

Figure 16: ECU1 design feasibility regions with respect to different bus speed values.

(FP) scheduling policy, where T_3 (resp. T_4) has a higher-priority than that of T_1 (resp. T_2). The message m_1 shares the bus with other higher-priority messages under the non-preemptive FP scheduling. We assumed a fixed frame length for every CAN frame in the system and a fixed frequency for ECU3.

Given the above setting, we applied our control design algorithm to compute the drop bound that the lane keeping system can accommodate. For the platform analysis, we used the RTC analysis methods [8,10] to compute the service functions, β_{T_1}, β_{T_2}, β_{m_1}, that capture the resource available to T_1, T_2, and m_1, respectively. These values were used as inputs to our method and the baseline method.

Our evaluation focuses on the minimum frequency that ECU1 must operate at to guarantee the control quality of the lane keeping system. For this, we considered different frequencies of ECU2 and different bus speed values. For each frequency of ECU2, we varied the delay threshold τ_{th} within the feasible control design range ($\tau_{th} \leq T = 50ms$) and determined the minimum frequency of ECU1 to ensure the control quality under the considered threshold delay.

Our method selects the frequency using the approach in §7. The baseline method does not employ buffer control, and it selects the smallest processor frequency such that the maximum end-to-end delay of the sensor data, computed using the conventional RTC analysis [8], is no more than the delay threshold τ_{th}.

8.2 Evaluation results

Resource savings. Fig. 15 shows the minimum frequency of ECU1 computed using the two methods for three different frequencies of ECU2 (30MHz, 60MHz, and 90MHz) and a medium-speed CAN bus (250 kBits/s). We observe that the co-design method consistently outperforms the baseline method. For example, compared to the baseline method, our method reduces the frequency by 50% when $\tau_{th} = 35$ms, and it improves over 4 times when $\tau_{th} = 50$ms. Further, when the delay threshold falls within the shaded area, no solutions exist under the baseline method, whereas it is possible to design ECU1 using our co-design. For instance, when $\tau_{th} = 20$ms and ECU2 operates at 30MHz, a frequency of 30MHz for ECU1 is sufficient to guarantee the control quality using the co-design method. On the contrary, the baseline method requires an infinite-valued frequency (hence, not shown in the figures), which is infeasible.

Impact of delay threshold. It can be observed from Fig. 15 that smaller threshold values require higher processor frequencies. This is expected because the resource demands of the control tasks increase as the threshold decreases. Since smaller delay threshold typically results in better control quality, the obtained results can be used to find a threshold value that balances the tradeoff between control quality and platform resource.

Design space exploration. The obtained results also help guide the platform design exploration. For instance, for a 15ms delay threshold, ECU1 requires an unbounded frequency under both methods if ECU2 operates at 30MHz, based on the results shown in Fig. 15(a). Then, a feasible solution only exists if we increase the frequency of ECU2 or the bus bandwidth. This is validated by the results in Fig. 15(b): when the frequency of ECU2 is increased to 60 MHz, a minimum of 40MHz for ECU1 becomes sufficient to meet the control objective under the co-design method. Note, however, that under the baseline method, we need to increase the frequency of ECU2 further, since the computed frequency for ECU1 is still unbounded.

Fig. 16 illustrates the feasibility design regions for ECU1 under three different bus speeds: low (125 kBits/s), medium (250 kBits/s) and high (500 kBits/s). The areas under the baseline and co-design curves in each figure correspond to the regions for which a feasi-

ble frequency exists for ECU1 under our co-design method and the baseline method, respectively. We observe that as the frequency of ECU2 increases, the feasible range of the delay threshold is also widen, enabling smaller delay thresholds and thus better control quality. Similarly, as the bus data rate increases, the feasible region is also enlarged for both methods. This is expected, since the bus bandwidth is the constraint factor of these feasible regions. These feasible regions can be used to optimize the platform resource under a given resource constraint.

We also observe from Fig. 16 that the feasible region of the baseline method consistently falls strictly inside that of the co-design method. Further, when ECU2 operates at 10 MHz, no solution exists for the baseline method, regardless of the threshold delay and bus speed values. Thus, our co-design method not only enables resource savings but also provides more flexibility for the platform design.

9 Conclusion

We have presented a control and platform co-design method for cyber-physical systems, which allows dropped samples to optimize resource while guaranteeing the control quality. We have developed a dynamic model including delay and analyzed its stability switching theory criteria. First using matrix inequalities, an upper bound for the maximum number of packet dropouts in any interval was derived to guarantee stability. Then a more practical approach using a multiple Lyapunov functions was developed and proved. The latter allowed more freedom in the platform design. A buffer control mechanism was introduced that utilizes the control design capability in accommodating dropped samples to reduce the resource requirements of the system. We have also presented a technique for computing the drop bounds under the proposed mechanism, and demonstrated how they can be used for dimensioning the platform resource. Our evaluation results of a lane keeping control system case study shows that our co-design method not only helps improves the resource use efficiency by an order of magnitude but also enables design solutions that are infeasible under the conventional baseline design approach.

10 References

[1] A. Annaswamy, S. Chakraborty, D. Soudbakhsh, D. Goswami, and H. Voit. The arbitrated networked control systems approach to designing cyber-physical systems. In *NecSys*, 2012.

[2] A. Annaswamy, D. Soudbakhsh, R. Schneider, D. Goswami, and S. Chakraborty. Arbitrated network control systems: A co-design of control and platform for cyber-physical systems. In *Workshop on Control of Cyber Physical Systems*, 2013 (submitted).

[3] K. J. Aström and B. Wittenmark. *Computer-controlled systems (3rd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[4] G. Berry and G. Gonthier. The esterel synchronous programming language: design, semantics, implementation. *Science of Computer Programming*, 19(2):87 – 152, 1992.

[5] J.-Y. L. Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.

[6] M. S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475 – 482, 1998.

[7] A. Cervin and J. Eker. The control server: a computational model for real-time control tasks. In *ECRTS*, 2003.

[8] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *DATE*, 2003.

[9] R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, 2000.

[10] W. Haid and L. Thiele. Complex task activation schemes in system level performance analysis. In *CODES+ISSS*, 2007.

[11] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *Computers, IEEE Transactions on*, 44(12):1443–1451, 1995.

[12] T. Henzinger, B. Horowitz, and C. Kirsch. Giotto: a time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84 – 99, jan 2003.

[13] J. Hespanha and A. Morse. Stability of switched systems with average dwell-time. In *CDC*, 1999.

[14] J. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

[15] P. Kumar, D. Goswami, S. Chakraborty, A. Annaswamy, K. Lampka, and L. Thiele. A hybrid approach to cyber-physical systems verification. In *DAC*, 2012.

[16] M. Lemmon and X. S. Hu. Almost sure stability of networked control systems under exponentially bounded bursts of dropouts. In *HSCC*, 2011.

[17] Q. Ling and M. Lemmon. Robust performance of soft real-time networked control systems with data dropouts. In *CDC*, 2002.

[18] O. Mason and R. Shorten. On common quadratic Lyapunov functions for stable discrete-time LTI systems. *IMA Journal of Applied Mathematics*, 69(3):271–283, 2004.

[19] P. Naghshtabrizi and J. Hespanha. Analysis of distributed control systems with shared communication and computation resources. In *ACC*, 2009.

[20] E. Poggi, Y. Song, A. Koubaa, Z. Wang, et al. Matrix-dbp for (m, k)-firm real-time guarantee. *RTSS*, 2003.

[21] G. Quan and X. Hu. Enhanced fixed-priority scheduling with (m, k)-firm guarantee. In *RTSS*, 2000.

[22] P. Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):549–559, 1999.

[23] S. Samii, A. Cervin, P. Eles, and Z. Peng. Integrated scheduling and synthesis of control applications on distributed embedded systems. In *DATE*, 2009.

[24] D. Soudbakhsh and A. Eskandarian. Vehicle lateral and steering control. In A. Eskandarian, editor, *Handbook of Intelligent Vehicles*, pages 209–232. Springer London, 2012.

[25] D. Soudbakhsh, A. Eskandarian, and D. Chichka. Vehicle collision avoidance maneuvers with limited lateral acceleration using optimal trajectory control. *ASME Journal of Dynamic Systems, Measurement, and Control*, In Press, 2013.

[26] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680 –1685, sept. 2007.

[27] H. Voit, R. Schneider, D. Goswami, A. Annaswamy, and S. Chakraborty. Optimizing hierarchical schedules for improved control performance. In *SIES*, 2010.

[28] W. Wolf. Cyber-physical systems. *Computer*, 42(3):88 –89, march 2009.

[29] F. Xia and Y. Sun. Control-scheduling codesign: A perspective on integrating control and computing. *Dynamics of Continuous, Discrete and Impulsive Systems - Series B*, 13(S1):1352–1358, 2006.

[30] M. Yu, L. Wang, T. Chu, and G. Xie. Stabilization of networked control systems with data packet dropout and network delays via switching system approach. In *CDC*, 2004.

[31] W. Zhang, M. Branicky, and S. Phillips. Stability of networked control systems. *IEEE Control Systems*, 21(1):84–99, 2001.

[32] J. Zou, S. Matic, E. A. Lee, T. H. Feng, and P. Derler. Execution strategies for ptides, a programming model for distributed embedded systems. *RTAS*, 2009.