Quantifying notions of extensibility in FlexRay schedule synthesis
Schneider, R.; Goswami, D.; Chakraborty, S.; Bordoloi, U.; Eles, P.; Peng, Z.

Published in:
ACM Transactions on Design Automation of Electronic Systems

DOI:
10.1145/2647954

Published: 01/01/2014

Document Version
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. Dec. 2018
1 Introduction

The design of distributed control systems – where control tasks are implemented on different ECUs communicating via a shared bus – typically require end-to-end timing guarantees from the embedded implementation platform. For example, in Fig. 1 the controller may be designed with the assumption of a specified maximum sensor-to-actuator delay, which includes the computation times of the software tasks and the signal transmission delays on the communication bus. Such delay assumptions are guaranteed by using appropriate scheduling policies along with necessary timing or schedulability analysis. For many real-life communication protocols like CAN or FlexRay, providing tight analytical timing bounds is often difficult and leads to either pessimistic results or resource overprovisioning (e.g., see [2]). In many cases, this problem is circumvented by using resource reservation techniques like time-triggered protocols that provide better timing guarantees and are easier to analyze. But, they are also usually more conservative and lead to poor resource utilization.

In contrast to this constraint on all control signals having to meet their deadlines, in this paper we propose controller design techniques that allow control signals to occasionally miss their deadlines (or be dropped). Our main technical contribution is to provide an analytical bound on such deadline misses such that stability and control performance are nevertheless guaranteed. We believe that such relaxation may be exploited in the implementation platform design and analysis stages. For example, instead of using time-triggered communication that ensures all control signals are delivered in time, priority-based communication protocols may be used with certain signals missing their deadlines. Further, bounds on deadline misses from our controller design stage may be used as an additional “safety margin”, thereby allowing less precise and hence less pessimistic – e.g., simulation-based – timing analysis techniques for the implementation platform. It is worth mentioning that recently there has also been work on verification of communication schedules, which ensure that bounds on allowable control signal deadline misses are satisfied by the implementation platform [1], [8], [9], [16]. These results may be coupled with our proposed design to provide certifiable implementations where needed.

The issue of incorporating feedback signal delay into controller design has been widely studied by the Networked Control Systems (NCS) community (e.g., see [10], [11], [12]). One of the seminal results on bounding signal deadline misses in a digital implementation platform of such feedback loops, while guaranteeing stability, along the lines of our work, may be found in [17]. Since then a number of other papers have reported various special cases of this result (e.g., see [7], [13], [15]). These results mostly provide bounds over an infinite horizon of samples (i.e., address asymptotic behavior), which are usually difficult to check or formally verify for an implementation platform. Moreover, all such results are concerned with stability and not performance, which is important in real-life settings. We attempt to overcome these shortcomings; our proposed design method has better applicability, which we illustrate using a FlexRay-based [5] distributed controller design example from the automotive domain.

2 Problem Formulation

We study the design of feedback controllers for linear time-invariant (LTI) dynamical systems (or plants),
\[
    \begin{align*}
    \dot{x}(t) &= A_t x(t) + B_t u(t) \\
    y(t) &= C_t x(t)
    \end{align*}
\]
where \(x(t)\) is the \(n\times1\) vector of state variables and \(u(t)\) is the control input to the system. \(A_t\) is a \(n\times n\) system matrix and \(B_t\) is a \(n\times1\) vector. A typical feedback control loop performs the following three sequential operations:

- measure the states \(x(t)\) (measure),
- compute input signal \(u(t)\) (compute) and,
- apply the computed \(u(t)\) to the plant (1) (actuate).

In a digital implementation platform of such feedback loops, these operations are performed only at discrete-time intervals (sampling instants). When the time interval between two
consecutive sampling instants is constant, the continuous-time system (1) can be transformed into the discrete-time system,
\[
\begin{align*}
x[k+1] &= Ax[k] + Bu[k] \\
y[k] &= Cx[k]
\end{align*}
\]
where the sampling instants are \( t = h_k \) (\( k = \{1, 2, 3, \ldots\} \)) and \( h_{k+1} - h_k = h \) (sampling period). \( x[k] \) and \( u[k] \) are the values of \( x(t) \) and \( u(t) \) at \( t = h_k \) and
\[
A = e^{Ah}, B = \int_0^h (e^{At}dt) \cdot B, C = C_t.
\]
In the rest of this work, we consider the discrete-time state-space model shown in (2) as our model of dynamical systems.

Control objectives: We consider the set-point tracking problem where the control objectives are the following:
1) Achieve exponential stability, i.e.,
\[
\|x[k]\| \leq c\lambda^k \|x[0]\|,
\]
where \( 0 < \lambda < 1 \) and \( c \) is a constant.
2) Achieve \( y[k] \to r \) as \( k \to \infty \), \( r \) is the reference. The input \( u[k] \) is designed to meet these objectives.

2.1 Distributed Implementation
The distributed implementation platforms we consider are of the form shown in Fig. 1. In this setup, there are multiple Electronic Control Units (ECUs); multiple tasks are mapped onto each ECU and are executed according to the scheduling policy implemented on the ECU. Here, a control application is partitioned in two tasks – a plant task \( T_p \) and a controller task \( T_c \). These tasks run on two different ECUs that communicate over a shared bus. A bus protocol allows messages (generated by tasks on the ECUs) to be scheduled on the bus. Each such message is associated with an input and an output buffer. When a message is generated, it is placed on the output buffer of the ECU, it waits for bus access and gets transmitted according to the bus schedule. Similarly, when a message arrives at the receiving ECU, it is placed on its input buffer and the corresponding task reads the message when necessary. \( T_p \) runs periodically with period \( h \) and performs two operations (i) reading the feedback signal \( x[k] \) from sensors (i.e., measure) and placing it as message \( m_x \) in the output buffer (ii) reading message \( m_c \) (i.e., the feedback component of \( u[k] \)) from the input buffer, adding feedforward part to it, and applying (i.e., actuate) it to the physical system using an actuator. \( T_c \) runs periodically with period \( h \) and receives \( m_x \) (i.e., \( x[k] \)) from the input buffer, computes the feedback component of \( u[k] \) (i.e., compute) and places it in the output buffer as message \( m_c \). Fig. 2 shows the timing diagram of the control application.

The time interval between the measurement of \( x[k] \) and the corresponding actuation signal \( f(x[k]) \) (see Fig. 3) is the sensor-to-actuator delay \( \tau \). As shown in Fig. 2, the time duration between reading the sensor data and receiving the next control input from the input buffer is the delay \( \tau \). It should be noted that \( T_p \) sends \( m_x \), which is the current state \( x[k] \). At the same time, \( T_p \) reads \( m_c \) which is the feedback component of the control input computed using the older state \( x[k − \lceil \frac{\tau}{h} \rceil] \) and applies it to the physical system. Clearly, \( \tau > 0 \) and \( \lceil \frac{\tau}{h} \rceil \geq 1 \). Such a setting is common in real-life applications where the computational processes and control algorithms run at distributed locations and signals are exchanged over the network or a communication bus [6], [14].

2.2 Communication Bus
We will illustrate our proposed design approach using the priority-based (or event-triggered) dynamic segment of FlexRay as the communication bus. FlexRay [5] is commonly used in the automotive domain and supports both time-triggered and event-triggered communication. The time-triggered communication guarantees that all control signals are delivered in time, but the time-triggered (or static) segment is considered as a “premium” resource. The dynamic segment on the other hand might cause high variations in feedback delays and occasional deadline misses (depending on the other messages being scheduled). Our proposed controller design opens up the possibility of using the less expensive dynamic segment of FlexRay, while ensuring that both stability and performance constraints are nevertheless satisfied. Fig. 4 shows a typical delay variation experienced by control messages in the dynamic segment. We briefly describe the working of the FlexRay protocol in Appendix A5 (see [5] for more details).

2.3 Ideal and Non-ideal Samples
As discussed above, the control signal \( u[k] \) waits as message \( m_c \) in the input buffer. If \( \lceil \frac{\tau}{h} \rceil = 1 \), \( m_c \) gets updated by \( u[k−1] \) and we henceforth refer to such samples as ideal samples. Otherwise, if \( \lceil \frac{\tau}{h} \rceil > 1 \) and the control input gets further delayed, resulting in \( u[k−\lceil \frac{\tau}{h} \rceil] \) and the corresponding control signals are referred to as non-ideal samples.
Fig. 5. Sequence of ideal and non-ideal samples.

2.4 Control Scheme

Based on the above classification of samples, we apply two different control algorithms for ideal and non-ideal sampling. The overall control scheme is the following:

\[ u[k] = \begin{cases} Kx[k-1] + F_1r, & \text{ideal samples} \\ F_2r, & \text{non-ideal samples} \end{cases} \tag{5} \]

where \( K \) is the feedback gain and \( F_1 \) and \( F_2 \) are the feedforward gains. Clearly, we apply feedback control only for the ideal samples and apply feedforward control (which does not need any communication over the bus) in the case of non-ideal samples. The closed-loop system follows (6) for the ideal samples and (7) for the non-ideal samples.

\[ x[k+1] = Ax[k] + BKx[k-1] + BFr \tag{6} \]
\[ x[k+1] = Ax[k] + BFr \tag{7} \]

With the ideal samples, the system is \textit{stabilizable} if there exist a feedback gain \( K \) such that the closed-loop system (6) is stable. The stabilizability condition for the above system is derived in Appendix A1.

2.5 Closed-loop System

Depending on the occurrence of ideal and non-ideal samples (see Fig. 5), the closed-loop dynamics of the system switches between (42) and (45). The occurrence of ideal and non-ideal samples causes \textit{switching} between the system matrices \( A_{cl} \) and \( A_o \) (see Appendix A3 and A4). An arbitrary ordering of ideal and non-ideal samples can be modeled as follows:

\[ x[k] = A_{cl}^{n_1} A_o^{n_2} A_{cl}^{n_3} A_o^{n_4} A_{cl}^{n_5} A_o^{n_6} \ldots x[0] + w[k]B_{cl}r \tag{8} \]

where \( n_i \) are integers such that \( \sum_i n_i = k \) (integer) and \( w[k] \) is given by

\[ w[k+1] = A_{cl}w[k] + F_1 \tag{9} \]

for the ideal samples and

\[ w[k+1] = A_ow[k] + F_2 \tag{10} \]

for the non-ideal samples. Since the number of non-ideal samples is (typically) significantly lower than the number of ideal samples (for a sufficiently large value of \( k \)), the steady state value of \( w[k] \) is mainly dominated by (9). Thus, by putting \( w[k+1] = w[k] \), we obtain from (9),

\[ w_{ss} = (I - A_{cl})^{-1}F_1. \]

Let us define the steady state values of the states as,

\[ x_{ss} = w_{ss}B_{cl}r = (I - A_{cl})^{-1}B_{cl}F_1r. \tag{11} \]

When \( k \) is sufficiently large (i.e., \( k \to \infty \)),

\[ x[k] = A_{cl}^{n_1} A_o^{n_2} A_{cl}^{n_3} A_o^{n_4} A_{cl}^{n_5} A_o^{n_6} \ldots x[0] + x_{ss}, \] \[ \Rightarrow x[k] - x_{ss} = A_{cl}^{n_1} A_o^{n_2} A_{cl}^{n_3} A_o^{n_4} A_{cl}^{n_5} A_o^{n_6} \ldots x[0]. \tag{12} \]

Clearly, with \( k \to \infty \) and stable (12), \( x[k] \to x_{ss} \) which implies \( y[k] \to r \). In the following, we analyze the stability and the performance of the system given by (12).

3 Stability and Performance Analysis

For any matrix \( A_i \), there exist constant scalars \( \eta_i \) and \( \lambda_i \) such that the following inequality holds:

\[ \| A_i^k \| \leq \eta_i \lambda_i^k, \tag{13} \]

where \( k \geq 1 \). Therefore, we have

\[ \| A_o^k \| \leq \eta_1 \lambda_1^k, \| A_{cl}^k \| \leq \eta_2 \lambda_2^k, \tag{14} \]

Here, we assume that the system (6) is stabilizable satisfying the condition (34) and thus,

\[ \frac{\alpha_n}{n+1} \leq \lambda_1 < 1. \]

\( \lambda_2 \) depends on the open-loop system \( A_o, \lambda_2 \geq 1 \) and \( \lambda_2 < 1 \) for open-loop unstable and stable system respectively. Let us define the following quantities:

\[ \mu(k) = \frac{n_1 + n_3 + n_5 \ldots}{n_2 + n_4 + n_6 \ldots} \] \[ k = (\mu(k) + 1) \times \kappa. \tag{15} \]

The value of \( \mu(k) \) denotes the ratio between number of ideal and non-ideal samples (see Fig. 5) and \( \kappa \) is the number of non-ideal samples among \( k \) consecutive samples.

3.1 Stability Analysis

Here, we study exponential stability as defined in (4) of the system (12).

\textbf{Theorem 1: (Exponential stability)} Consider the switched system (12) where \( A_{cl} \) and \( A_o \) satisfy (14). Let \( \mu(k) \) and \( \kappa \) be defined as per equation (15). The switched system (12) will be exponentially stable if the following conditions are satisfied.

\textbf{C1:} \( \mu(k) > \mu^* > 1 \) for \( k > \left( \mu^* - 1 \right) \) where

\[ \mu^* = \frac{\ln \frac{1}{\lambda_2 \eta_1 \eta_2 \lambda_2}}{\ln \lambda_1}. \tag{16} \]

\textbf{C2:} \( \eta_1 \eta_2 \lambda_1^{\mu(k)} \lambda_2 < 1 \)

\textbf{Proof:} We assume that the number of ideal samples is greater than the number of non-ideal samples over \( k \) initial sampling intervals, i.e., \( \mu(k) > 1 \).

Let us start with the case where \( \kappa = 1 \), i.e., there is only one non-ideal sample among the \( \mu(k) + 1 \) samples. In this case a maximum of two switchings are possible, i.e., the non-ideal sample occurs somewhere in between \( \mu(k) \) ideal samples. Without loss of generality, we assume \( r = 0 \). Hence,

\[ x[k] = A_{cl}^{n_1} A_o A_{cl}^{n_2} x[0], \]

and by utilizing the properties given by (14), we get,

\[ \| x[k] \| \leq \eta_1 \eta_2 \lambda_1^{\mu(k)} \lambda_2 \| x[0] \|. \tag{17} \]

Similarly, for \( \kappa = 2 \), a maximum of four switchings are possible, e.g.,

\[ x[k] = A_{cl}^{n_1} A_o A_{cl}^{n_2} A_o A_{cl}^{n_3} A_o A_{cl}^{n_4} \ldots x[0] \]

and

\[ \| x[k] \| \leq \eta_1 \eta_2 \lambda_1^{2 \mu(k)} \lambda_2^2 \| x[0] \|. \tag{18} \]
Hence, for any $\kappa$,  
\[ \|x[k]\| \leq \eta_1(\eta_1 \eta_2 \lambda_1^k \lambda_2) \kappa \|x[0]\|. \tag{19} \]

**Case I with $\kappa = 1$:** In equation (17), if $\eta_1^2 \eta_2 \lambda_1^k \lambda_2 = 1$, then,  
\[ \mu(k) = \mu^* = \frac{1}{\eta_1 \eta_2} \ln \frac{1}{\lambda_1}. \]
As $\lambda_1 < 1$, if condition C1 is satisfied, i.e., $\mu(k) > \mu^*$, then $\eta_1^2 \eta_2 \lambda_1^k \lambda_2 < 1$.

**Case II with $\kappa > 1$:** If $\kappa > 1$ and condition C2 is satisfied,  
\[ (\eta_1 \eta_2 \lambda_1^k \lambda_2)^{\kappa} < \eta_1^2 \eta_2 \lambda_2 \]
Hence,  
\[ (\eta_1 \eta_2 \lambda_1^k \lambda_2)^{\kappa} < 1. \]

From Cases I and II, it may be noticed that if the conditions C1 and C2 are satisfied, we get $\eta_1(\eta_1 \eta_2 \lambda_1^k \lambda_2)^{\kappa} < 1$, which results in $\|x[k]\| < \|x[0]\|$. This shows that $\|x[k]\|$ decreases with $k$ when conditions C1 and C2 are satisfied.

Furthermore,  
\[ (\lambda_1^k \lambda_2)^{\kappa} = (\lambda^*)^k, \]
where  
\[ \ln(\lambda^*) = \frac{\mu(k) \ln \lambda_1 + \ln \lambda_2}{\mu(k) + 1}. \tag{20} \]

Clearly, for a given combination of $\lambda_1$, $\lambda_2$, and $\mu(k)$ satisfying the conditions C1 and C2 stated in Theorem 2, $\lambda^* < 1$. Therefore, the system (12) is globally exponentially stable with stability degree $\lambda^*$ if $C1$ and $C2$ are satisfied. The stability degree is higher with higher $\mu(k)$ and lower $\lambda^*$. \hfill \square

### 3.2 Performance Analysis

In this work, we consider the ability to respond to an external disturbance as a measure of performance. This is defined by the tuple $\{S, \chi\}$ whose components are related as:

\[ S \geq \frac{\|x[k + \chi] - x_{ss}\|}{\|x[k]\|}, \tag{21} \]

where $0 \leq S \leq 1$, $\chi$ is a positive integer and $x_{ss}$ is as per (11). We assume that $\|x[k]\| = (x_{ss} + d)$ due to certain external disturbance at the $k^{th}$ sampling instant. $S$ indicates how much disturbance is rejected over any $\chi$ consecutive sampling intervals. For example, $\{S, \chi\} = \{0.02, 100\}$ indicates that 98% of the disturbance is rejected within 100 consecutive sampling intervals. It should be noted that the performance metric $\{S, \chi\}$ requires the minimum inter-arrival time between two consecutive disturbance arrivals to be $\chi$ samples. In this work, we consider a performance requirement of $\{S, \chi\}$ and a sporadic disturbance arrival model with the minimum inter-arrival time between disturbances to $\chi$ samples. A change in reference $r$ implies a change in $x_{ss}$, requiring a different performance as per (21). Therefore, our analysis works for both regulation and servo problems.

**Theorem 2:** (*Performance guarantee*) Consider the switched system (12) with a performance requirement $\{S, \chi\}$ and let $A_{cl}$, $A_s$ satisfy (14). The switched system is guaranteed to meet the performance requirement if no more than $\kappa_s$ non-ideal samples occur within any interval of $\chi$ consecutive samples, where $\kappa_s$ is given by

\[ \kappa_s = \frac{\ln S - \ln \eta_1 - \chi \ln \lambda_1}{\ln \eta_1^2 \eta_2 \lambda_2 \lambda_1}. \tag{22} \]

**Proof:** We assume that the minimum ratio between the number of ideal and non-ideal samples in any $\chi$ consecutive samples is $\mu_s$, $\mu_s > 1$ and,  
\[ \chi = (\mu_s + 1) \kappa_s, \tag{23} \]
where $\kappa_s$ is the maximum number of non-ideal samples that can occur in any $\chi$ consecutive samples. From (19), we get

\[ \|x[k + \chi] - x_{ss}\| \leq \eta_1(\eta_1 \eta_2 \lambda_1^\kappa \lambda_2) \kappa^s \|x[k]\|. \tag{24} \]

Next, from (21), we have  
\[ S = \eta_1(\eta_1 \eta_2 \lambda_1^\kappa \lambda_2) \kappa^s. \tag{25} \]

Combining (23) and (25), we obtain

\[ \ln S = \ln \eta_1 + \kappa_s \ln \eta_1 \eta_2 + \kappa_s \mu_s \ln \lambda_1 + \kappa_s \ln \lambda_2, \]
\[ \Rightarrow \ln S = \ln \eta_1 + \chi \ln \lambda_1 + \kappa_s \ln \eta_1 \eta_2 \lambda_2 \lambda_1, \]
\[ \Rightarrow \kappa_s = \frac{\ln S - \ln \eta_1 - \chi \ln \lambda_1}{\ln \eta_1 \eta_2 \lambda_2 \lambda_1}. \tag{26} \]

### 3.3 Illustrative Example

We now illustrate the above results using an example. Let us assume that $\mu_s = 1.9$ for a given system and consider a sequence of ideal and non-ideal samples as shown in Fig. 5. Clearly, $\mu(k) > \mu^*$ for all $k > 3$ and the system under consideration is exponentially stable as per *Theorem 1*. Further, we assume that a given performance requirement $\{S, \chi\} = \{*, 5\}$ results in $\kappa_s = 2$. The sequence shown in Fig. 5 does not guarantee this performance requirement.

### 4 Experimental Validation

To illustrate the applicability of our proposed design method we used an automotive cruise control system as an example. It receives the reference or the commanded vehicle’s speed from the driver and regulates the speed following the driver’s command. Based on the reference speed and the feedback signals, the cruise control system regulates the vehicle’s speed by adjusting the engine throttle angle to increase or decrease the engine drive force. In this case study, we have used a model of a cruise controller that was developed in consultation with a major German automotive company. The linearized continuous-time model of this cruise control system is shown in (27). The state $v_1(t)$ captures the speed of the vehicle and $u(t)$ is the engine throttle angle. The objective is to choose $u(t)$ such that $v_1(t) = r$, i.e., a constant desired speed. We have chosen $r = 100$. Moreover, we need to satisfy design requirements, such as the settling time of the velocity $v_1(t)$ should be less than 5 sec.

\[
\begin{align*}
\frac{dv_1(t)}{dt} &= A_r v(t) + B_r u(t), \\
v(t) &= \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix}, \\
A_r &= \begin{bmatrix} 0 & 1.0 & 0 \\ 0 & -6.05 & -5.29 \\ 0 & -2.48 & -0.24 \end{bmatrix}, \\
B_r &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \\
C_r &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.
\end{align*}
\tag{27}
\]

### 4.1 Implementation Details

This cruise control application was implemented in a distributed fashion, as shown in Fig. 1. Following the notation introduced earlier, the states $v_1(t)$ are measured by the task $T_p$ at the sampling times $t = h_k$ with $k = \{0, 1, 2, \ldots\}$, and $h_{k+1} - h_k = h$. $v_1(t)$ at $t = h_k$ is denoted by $v[k]$. The control input $u(t)$ is computed in $T_c$, and $u(t)$ (which utilizes $v[k]$) is denoted by $u[k]$. The output $u[k]$ is sent to the task $T_p$ via the dynamic segment of the FlexRay communication bus. $T_p$ applies the control input to the engine throttle angle. The resulting discrete-time system is modeled as (2).
TABLE 1

Existing FlexRay schedules.

<table>
<thead>
<tr>
<th>$m_i$</th>
<th>$(S_i, B_i, R_i)$</th>
<th>$o_i$(ms)</th>
<th>$p_i$(ms)</th>
<th>$[r_{min,i}, r_{max,i}]$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(12, 0.2)</td>
<td>9</td>
<td>20</td>
<td>[0.7, 2.7]</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>(14, 0.2)</td>
<td>2</td>
<td>20</td>
<td>[0.7, 2.7]</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>(14, 2.4)</td>
<td>2</td>
<td>40</td>
<td>[1.7, 6.7]</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>(16, 0.4)</td>
<td>3</td>
<td>50</td>
<td>[1.7, 6.7]</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>(17, 8)</td>
<td>12</td>
<td>100</td>
<td>[1.7, 6.7]</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>(17, 4.8)</td>
<td>14</td>
<td>100</td>
<td>[1.7, 6.7]</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>(20, 2.4)</td>
<td>18</td>
<td>80</td>
<td>[2.7, 10.7]</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>(25, 0.2)</td>
<td>0</td>
<td>20</td>
<td>[0.7, 2.7]</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>(25, 1.4)</td>
<td>17</td>
<td>100</td>
<td>[6.1, 26.1]</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>(28, 0.4)</td>
<td>21</td>
<td>60</td>
<td>[2.8]</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>(30, 1.2)</td>
<td>11</td>
<td>50</td>
<td>[1.7, 6.7]</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>(33, 0.2)</td>
<td>25</td>
<td>20</td>
<td>[1.4]</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>(34, 0.1)</td>
<td>2</td>
<td>20</td>
<td>[0.7, 2.7]</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>(38, 0.2)</td>
<td>30</td>
<td>40</td>
<td>[1.3, 5.3]</td>
<td>5</td>
</tr>
</tbody>
</table>

FlexRay parameters: Configuring the FlexRay dynamic segment involves assigning values to a number of parameters, which in turn determines the priorities of messages and hence the delays experienced by them. These configuration parameters were specified using the EB Designer Pro tool (from Elektrobit). The cycle length was set to $gdCycle = 5ms$ with the static segment of length $2ms$ and 10 static slots. The rest of the cycle was distributed to the dynamic segment and the NIT (see Appendix A5). Further, the dynamic segment consisted of 60 minislots where the duration of one minislot was $0.05ms$. The value $pLatestTx$ was set to 50 for all messages in the network (the last minislot where a message transmission may begin is when the minislot counter is equal to $pLatestTx$). In order to simulate higher priority network traffic coming from the rest of the network (i.e., from other ECUs), we considered a pre-existing FlexRay network with several messages already mapped on to the DYN segment. The configuration parameters for these message are listed in Table 1; they include the schedule parameters $S_i, B_i, R_i$, task offsets $o_i$, periods $p_i$, uniformly distributed response times between $[r_{min,i}, r_{max,i}]$, and message sizes $c_i$ in minislots.

4.2 Design and Analysis

We chose a sampling period of 40ms and discretized the continuous system in (27) with $h = 40ms$ according to equation (3). The resulting discrete-time system is given by (28). The absolute values of the eigenvalues of $A$ in (28) are given by $\{0.6703, 1.1646, 1.1646\}$. Therefore, the resulting discrete-time open-loop system is unstable.

$$v[k+1] = Av[k] + Bu[k], ||y[k] = Cv[k]$$

$$A = \begin{bmatrix} 0.9398 & 0.3412 & 0.0718 \\ -0.4340 & 0.5605 & 0.3241 \\ -1.9603 & -2.1473 & 0.4833 \end{bmatrix},$$

$$B = \begin{bmatrix} 0.0247 \\ 0.1777 \\ 0.8028 \end{bmatrix}, C = [1, 0, 0].$$

(28)

Stabilizability: For the system in (28), $a_n = 1.9836$. Since $n = 3$ and $|a_n| < (n+1) = 4$, the system is stabilizable with control input $u[k] = Kx[k-1]$.

Controller synthesis: The feedback gain was designed (see appendix for details) with $p_i = 0.4959$ ($i = 1, 2, 3, 4$) and $K = [1.9833 \ 2.3580 \ 0.3652]$. The corresponding $A_{cd}$ and $A_o$ are given by the following

$$A_{cd} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.0605 & 0.4878 & -1.4755 & 1.9836 \end{bmatrix},$$

$$A_o = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0.9092 & -2.2367 & 1.9836 \end{bmatrix}.$$
control system of the form (2). As described earlier, the control application was partitioned into two tasks: \( T_p \) and \( T_c \). The plant task \( T_p \) was triggered with an offset \( o_p \), the controller task \( T_c \) was run on a different ECU with an offset \( o_c > o_p + r_{\text{max,} p} \) where \( r_{\text{max,} p} \) is the worst-case response time of the task \( T_p \) on its ECU. Both \( T_p \) and \( T_c \) were triggered periodically with period \( h \) (which is the sampling period of the control application).

\( T_p \) packetizes the sensor signals \( x[k] \) in the message \( m_x \) that is transmitted via the dynamic segment of FlexRay and receives \( m_c \) that is coming from \( T_c \). \( T_c \) receives \( m_x \) sent by \( T_p \), computes \( u[k] \), sends \( m_c \) via the dynamic segment of the bus to \( T_p \). While being transmitted via the FlexRay bus, \( m_x \) gets delayed by \( d_x \) and \( m_c \) gets delayed by \( d_c \). The resulting delay (in terms of samples) is given by

\[
\left\lceil \frac{\tau}{h} \right\rceil = \max\{0, \left\lceil \frac{d_x - o_c + o_p}{h} \right\rceil + \left\lceil \frac{d_c + o_c - o_p}{h} \right\rceil \}. \tag{30}
\]

The sensor-to-actuator delay has two components (see Fig. 7): (i) resulting from the offset between \( o_p \) and \( o_c \), and the delay \( d_x \) experienced by \( m_x \), (ii) resulting from the delay \( d_c \) experienced by \( m_c \). If \( d_c > (o_c - o_p) \), the corresponding sensor signal \( m_c \) can only be used in the next triggering instant of \( T_c \), resulting in a delay of \( \left\lceil \frac{d_x - o_c + o_p}{h} \right\rceil \) samples in the feedback loop. The input signal \( m_c \) experiences a delay of \( \left\lceil \frac{d_x - o_c + o_p}{h} \right\rceil \) samples. If the resulting delay \( \left\lceil \frac{\tau}{h} \right\rceil > 1 \), the corresponding sample is considered to be a non-ideal sample. The stability analysis and controller synthesis was done based on the occurrence of such non-ideal samples. Further, we assumed the response times \( r_i \) and \( r_i^* \) of the tasks \( T_p \) and \( T_c \) to be uniformly distributed between \([0, 3\text{ms}, 1.2\text{ms}]\), \( o_p = 0.7\text{ms}, o_c = 2.3\text{ms} > o_p + r_{\text{max,} p} \) and the periods of the tasks are 40ms. The FlexRay parameters for \( m_x \) were \((S_x, B_x, R_x) = (11, 0, 2)\) (i.e., it is the highest priority message). As \( m_x \) is the highest priority message, \( d_x < (o_c - o_p) \).

Therefore, \( \max\{\left\lceil \frac{d_x - o_c + o_p}{h} \right\rceil \} = 0 \). The delay variation in the feedback loop stems from the transmission delay of \( m_c \), i.e., \( d_c \). The sizes of \( m_x \) and \( m_c \) were set to 4 minislots. All of these parameters are typical of the automotive case study under consideration.

### 4.4 Results

We carried out 120 simulations with different schedules (i.e., values of \( (S_x, B_x, R_x) \) for \( m_c \)) at each simulation run. The simulation time was set to \( 100\text{sec} \) which corresponds to 2500 generated samples with a sampling period \( h = 40\text{ms} \). During each simulation we plotted the distribution of sensor-to-actuator delay \( \tau \) and analyzed the stability and performance of the discrete-time system. We will now discuss our observations for three example schedules that were synthesized for \( m_c \).

**Example Schedule 1:** The message \( m_c \) was assigned the schedule \((45, 0, 4)\). Fig. 8 (a) shows the delay distribution obtained from the co-simulation framework. \( \tau \) varies between \( \tau_{\text{min}} = 3.95\text{ms} \) and \( \tau_{\text{max}} = 44.5\text{ms} \). The number of non-ideal samples is 36 (i.e., those for which \( \left\lceil \frac{\tau}{h} \right\rceil > 1 \)). Fig. 8 (b) shows how \( \mu(k) \) varies with the sample number \( k \). Here, \( \mu^* = 27.9895 \), and \( \mu(k) > \mu^* \) for all \( k \geq 29 \) (since \( \mu^* + 1 = 29 \)). Hence, we conclude that condition C1 for stability is satisfied and clearly, condition C2 is also satisfied. The resulting system is stable. The sequence of the ideal and the non-ideal samples meets the criterion (26) which is reflected in the resulting output plot Fig. 8 (c) (i.e., settling time is 5.8sec or \( \{5, \chi\} = \{0.05, 125\} \)).

In the simulation, \( r = 100 \) and the initial speed \( v_1[0] = 0 \). \( v_1[k] \) reaches \( r = 100 \) in 1.76sec (i.e., the settling time = 1.76sec). We simulated external disturbances by periodically making \( v_1[k] = 80 \) (from \( v_1[k] = 100 \)). We could see that the maximum time taken to reject a disturbance is less than 2sec. Therefore, both stability and performance requirements are met for this choice of the schedule.

**Example Schedule 2:** In this example, we assigned the schedule \((47, 0, 4) \) to \( m_c \). Note that the slot number is 47 (instead of 45 as in the previous example) indicating a lower priority of the message \( m_c \). Naturally, transmission with a lower priority results in a higher probability of getting delayed, which is reflected in the following results. The delay distribution for \( m_c \) is shown in Fig. 8 (d). There are 241 non-ideal samples, i.e., where \( \left\lceil \frac{\tau}{h} \right\rceil > 1 \). The \( \mu(k) \approx 14 \) violates C1. The conditions C2 and (26) are satisfied. We simulated the discrete-time model with same disturbance that was applied to Example 1 and Fig. 8 shows the plot of \( v_1[k] \) for our new schedule. The overall system is stable even though C1 is violated. However, the system response deteriorated (which can be seen from the overshoot in \( v_1[k] \) in Fig. 8 (e)) due to higher number of non-ideal samples.

**Example Schedule 3:** We now show that the system can become unstable when the conditions C1, C2 and (26) are not satisfied. We assigned the schedule \((48, 0, 4) \) to \( m_c \); \( m_c \) now has an even lower priority than in the previous examples. Here, there are 316 non-ideal samples and \( \mu(k) \approx 9 \). Clearly, the condition C1 is violated. The plot of \( v_1[k] \) (see Fig. 8 (f)) shows the resulting instability in the output.

### 5 Concluding Remarks

The relaxed requirement on feedback signals to meet deadlines offers an additional safety margin while designing the implementation platform. In particular, we showed that the analytical bounds on allowed deadline misses from the controller design phase, may be checked against deadline misses suffered by messages in an implementation platform. The latter were obtained via simulation in this paper (as illustrated in the examples in Section 4.4). By having the latter bounds to be smaller than the former bounds, a safety margin may be ensured (as in Example Schedule 1). We believe that this would be the more pragmatic use of our results in real-life scenarios, where simulation-based timing analysis is prevalent. However, checks that “guarantee” that deadline miss bounds from the platform are smaller than what can be tolerated by the controller, may be designed using formal verification techniques; some results in this direction have already been reported recently (see [1], [8], [9], [16]). As a part of future work, they could be further refined to better match our proposed controller design.
REFERENCES


APPENDIX

(A1) Stabilizability with ideal sampling: Without loss of generality, we assume \( r = 0 \) in the closed-loop system (6). The resulting system becomes,

\[
x[k+1] = Ax[k] + BKx[k-1].
\]

The stabilizability analysis essentially reduces to finding the conditions for which there exists a feedback gain \( \hat{K} \) that can make the system (31) stable. First, we transform the state-space model into a controllable canonical form by a transformation \([3],[3]\).

\[
z[k] = Tx[k] \Rightarrow x[k] = T^{-1}z[k],
\]

where \( z[k] \) are the new states and \( T \) is the non-singular transformation matrix. We obtain the controllable canonical form as shown in (33).

\[
z[k+1] = A_c z[k] + B_c u[k],
\]

\[
A_c = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & \cdots & a_{n-1} & a_n \end{bmatrix},
\]

\[
B_c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.
\]

where \( a_i \) is obtained by the transformation and \( n \) is the dimension of system (2).

Theorem 3: (Stabilizability Condition) There exists a feedback gain \( K \) that places all the poles of the system (6) within the unit circle iff the following condition holds:

\[
|a_n| < (n+1).
\]

Proof: The control input for the ideal samples is shown in (35) with \( F_1 = 0 \).

\[
u[k] = K x[k-1] = \hat{K} z[k-1] = \hat{K}_1 z_1[k-1] + \hat{K}_2 z_2[k-1] + \cdots \hat{K}_n z_n[k-1] = \tilde{K}_1 z_1[k-1] + \tilde{K}_2 z_2[k-1] + \cdots \tilde{K}_n z_n[k-1].
\]

where \( K = \hat{K} T \). Let us introduce an additional state \( z_0[k] = z_1[k-1] \). Based on this additional state, we introduce new...
system states $Z[k] = \begin{bmatrix} z_0[k] \\ z[k] \end{bmatrix}$. Therefore, the closed-loop system leads to (36) which is of dimension $(n+1)$.

\[ z_0[k+1] = z_1[k], z_1[k+1] = z_2[k], z_2[k+1] = z_3[k] \ldots \]
\[ z_n[k+1] = \hat{K}_1 z_0[k] + (a_1 + \hat{K}_2) z_1[k] \ldots + (a_{n-1} + \hat{K}_n) z_{n-1}[k] + a_n z_n[k]. \]  

(36)
The above closed-loop system can be rewritten as follows:

\[ Z[k+1] = A_{cl} Z[k] \]

\[ A_{cl} = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 1 & 0 \\ \hat{K}_1 & \cdots & (a_{n-1} + \hat{K}_n) & a_n \end{bmatrix} \]  

(37)
The resulting characteristic equation for the above closed-loop system is obtained by making $|\lambda I - A_{cl}| = 0$ and is given by,

\[ \lambda^{n+1} - a_n \lambda^n - \cdots - (a_1 + \hat{K}_2) \lambda - \hat{K}_1 = 0. \]  

(38)
The system (31) is stabilizable if it is possible to place $(n+1)$ stable poles of the closed-loop system. It may be noted that all the coefficients of the above polynomial can be designed except the coefficient of $\lambda^n$. Suppose \( p = [-p_1 \ p_2 \cdots p_{n+1}] \) is the vector consisting poles of the closed-loop system. Then, the characteristic equation of the closed-loop system becomes,

\[ (\lambda + p_1)(\lambda + p_2)(\lambda + p_3) \cdots (\lambda + p_{n+1}) = 0 \]
\[ \Rightarrow \lambda^{n+1} + f_1(p) \lambda^n + f_2(p) \lambda^{n-1} + \cdots + f_n(p) = 0. \]  

(39)
For stabilizability, \( |p_i| < 1 \) and \( \hat{K} \) should be such that equations (38) and (39) are identical, i.e.,

\[ f_1(p) = -a_n, f_2(p) = -(a_{n-1} + \hat{K}_n), \cdots \]
\[ f_{n-1}(p) = -(a_1 + \hat{K}_2), f_n(p) = -\hat{K}_1. \]  

(40)
We can see from equation (40) that it is possible to design functions \( f_2(p) \cdots f_n(p) \) with appropriate choice of \( K \) only if \( f_1(p) = -a_n \). It is easy to see that \( f_1(p) = \sum_{i=1}^{n+1} |p_i| \).

Hence, the stabilizability condition is,

\[ \sum_{i=1}^{n+1} |p_i| = -a_n. \]  

(41)
For stability of the system (31), \( |p_i| < 1 \), \( \sum_{i=1}^{n+1} |p_i| < (n + 1) \) and \( \sum_{i=1}^{n+1} |p_i| \geq |a_n| \) using (41). Therefore, the condition (34) is the necessary condition for stabilizability.

\( \square \)

(A2) Design of feedback gain \( K \): For a given open-loop system matrices \((A, B)\) of \( n \)-order system:

1) Compute \( a_n \) and if \( |a_n| < (n + 1) \), the system is stabilizable and controller synthesis is possible.
2) Choose \( p = [-p_1 \ p_2 \cdots p_{n+1}] \) such that each element lies within the unit circle and \( \sum_{i=1}^{n+1} |p_i| = -a_n \).
3) Utilizing the closed-loop poles \( p \), compute \( \hat{K} \) from the equations (40).
4) Compute \( K \) by \( K = \hat{K} T \).

(A3) Design of feedforward gain \( F_1 \) for ideal samples: In the cases of ideal samples, the closed-loop dynamics is as follows,

\[ Z[k+1] = A_{cl} Z[k] + B_{cl} F_1 r \]
\[ y[k] = C_{cl} Z[k] \]  

(42) \[ B_{cl} = [0 \ 0 \ \cdots \ 1 \ 0 \ \cdots \ 0] \] and \( C_{cl} = [1 \ 0 \ \cdots \ 0 \ \cdots \ 0] \) are of dimensions $(n+1) \times 1$ and $1 \times (n+1)$ respectively.

The feedforward gain \( F_1 \) has to be chosen such that \( y[k] \to r \) as \( k \to \infty \), i.e., the steady state error is zero. In the steady state of the closed-loop system (42), \( Z[k+1] = Z[k] \). Hence,

\[ Z[k] = A_{cl} Z[k] + B_{cl} F_1 r \Rightarrow Z[k] = (I - A_{cl})^{-1} B_{cl} F_1 r \]
\[ y[k] = C_{cl} Z[k] = C_{cl}(I - A_{cl})^{-1} B_{cl} F_1 r. \]  

(43)
For \( y[k] = r \) in steady state, we obtain,

\[ F_1 = \frac{1}{C_{cl}(I - A_{cl})^{-1} B_{cl}}. \]  

(44)

(A4) Design of \( F_2 \) for non-ideal samples: In the case of non-ideal samples, the closed-loop dynamics is as follows,

\[ Z[k+1] = A_0 Z[k] + B_{cl} F_2 r \]
\[ y[k] = C_{cl} x[k], \]  

(45)
where \( A_0 = \{ A_{cl,} K = 0 \} \). For \( y[k] \to r \) as \( k \to \infty \) (similar to the computation of \( F_1 \)),

\[ F_2 = \frac{1}{C_{cl}(I - A_{cl})^{-1} B_{cl}}. \]  

(46)

(A5) FlexRay Protocol: The FlexRay communication protocol [5] is organized as a periodic sequence of communication cycles. Each cycle is of fixed length \( gdCycle \) and is indexed by a cycle counter that is incremented from 0 to 63 after which the counter is reset to 0. This communication pattern that is repeated periodically is known as the 64-cycle matrix. Further, every cycle consist of (i) a mandatory static segment (ST), (ii) an optional dynamic segment (DYN), and (iii) a segment for clock synchronization which is referred to as Network Idle Time (NIT). In the following we will discuss the communication specification of the DYN segment of FlexRay.

FlexRay dynamic segment: The DYN segment is partitioned into equal-length minislots that are indexed by a minislot counter which starts counting from 1 up to \( N \) minislots in every cycle. Additionally, a slot counter counts the communication slots that indicate time windows for admissible message transmissions. Each FlexRay message \( m_i \) is assigned a static schedule \( (S_i, B_i, R_i) \) for uniquely specified transmission points. A message \( m_i \) can successfully be transmitted via the DYN segment if the following requirements are satisfied:

- the assigned slot number \( S_i \in S_{DYN} \) is equal to the current slot counter value, where \( S_{DYN} \) is the set of available slot numbers in the DYN segment,
- the actual communication cycle is element of the set of feasible cycles \( \gamma_n \in \Gamma_n \) where \( \gamma_n = (B_i + N \times R_i) \ mod \ 64 \) with \( N \in \{0, 1, 2, \ldots \} \), \( R_i = 2^{r} \) for \( r \in \{0, 6 \} \) and \( B_i < R_i \),
- the minislot counter must not exceed the specified value of \( pLatestTx \) of the corresponding ECU.

A more detailed description of FlexRay protocol can be found in [5]. In a FlexRay schedule \( (S_i, B_i, R_i) \), the slot number \( S_i \) denotes the priority of the message and a higher \( S_i \) indicates a lower message priority. Therefore, a higher \( S_i \) essentially indicates a lower quality (and less expensive) communication.