Quantifying notions of extensibility in FlexRay schedule synthesis

Relaxing Signal Delay Constraints in Distributed Embedded Controllers

Dip Goswami, TU Eindhoven, Netherlands
Reinhard Schneider and Samarjit Chakraborty, TU Munich, Germany
d.goswami@tue.nl, reinhard.schneider@rcs.ei.tum.de, samarjit@tum.de

Abstract—Embedded systems often involve transmitting feedback signals between multiple control tasks that are implemented on different ECUs communicating via a shared bus. For ensuring stability and control performance, such designs require *all* control signals to be delivered within a specified deadline, which is ensured through appropriate timing or schedulability analysis. In this paper we study controller design that allows control feedback signals to occasionally miss their deadlines. In particular, we provide analytical bounds on deadline misses such that the control loop retains its stability and meets its control performance requirements. We argue that such relaxation allows us to (i) use *lower quality* communication resources (e.g., event-triggered instead of time-triggered communication) and (ii) provide more flexibility – e.g., use simulation – in communication timing analysis since analytical worst-case delay bounds for real-life communication protocols are often pessimistic. We illustrate this approach using the FlexRay communication protocol for distributed automotive control systems.

Index Terms—Distributed embedded controllers, flexible delay constraints, FlexRay, timing analysis

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 occassionally *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 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.

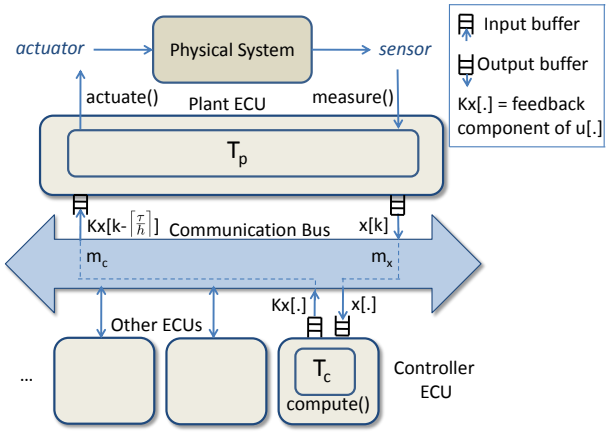

Fig. 1. A distributed embedded controller.

2 Problem Formulation

We study the design of feedback controllers for linear time-invariant (LTI) dynamical systems (or plants),

$$\begin{aligned} \dot{x}(t) &= A_t x(t) + B_t u(t) \\ y(t) &= C_t x(t) \end{aligned} \quad (1)$$

where $x(t)$ is the $n \times 1$ 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 $n \times 1$ 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,

$$\begin{aligned} x[k+1] &= Ax[k] + Bu[k] \\ y[k] &= Cx[k] \end{aligned} \tag{2}$$

where the sampling instants are $t = h_k$ ($k = \{1, 2, 3 \cdots\}$) 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

$$A = e^{A_t h}, B = \int_0^h (e^{A_t t} dt) \cdot B_t, C = C_t. \tag{3}$$

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.,

$$\|x[k]\| \leq c\lambda^k \|x[0]\|, \tag{4}$$

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 application of the corresponding actuation signal $f(x[k])$ (see Fig. 3) is the sensor-to-actuator delay τ. 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 τ. 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].

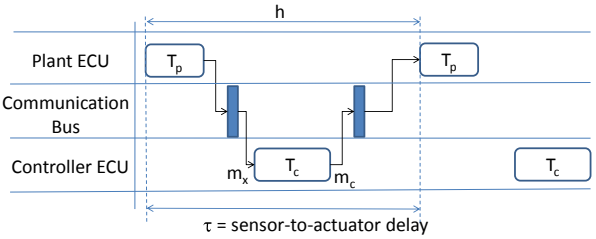

Fig. 2. Timing diagram.

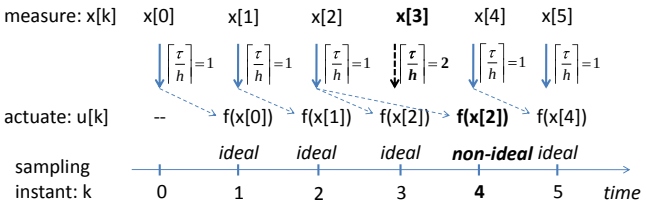

Fig. 3. *Ideal* and *non-ideal* samples – f(x[k]) denotes that the actuation signal uses the *measurement* from the k^{th} sample.

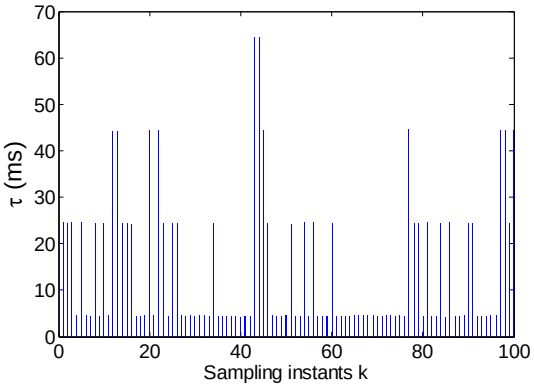

Fig. 4. Delay variation over the FlexRay dynamic segment.

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 $u[.]$ 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.

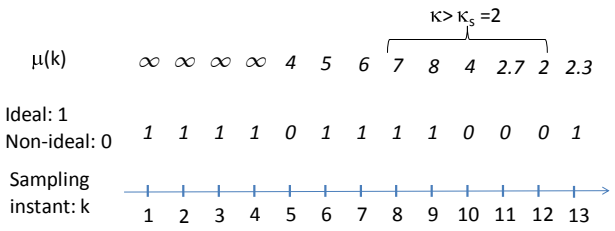

Fig. 5. Sequence of ideal and non-ideal samples.

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:

$$\begin{aligned} u[k] &= Kx[k-1] + F_1 r, \forall \textit{ ideal samples} \\ &= F_2 r, \forall \textit{non-ideal samples} \end{aligned} \quad (5)$$

where K is the feedback gain and F_1 and F_2 are the feedforward gains[1]. 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.

$$x[k+1] = Ax[k] + BKx[k-1] + BF_1 r \quad (6)$$

$$x[k+1] = Ax[k] + BF_2 r \quad (7)$$

With the ideal samples, the system is *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. We present a design methodology for computing the feedback and the feedforward gains K, F_1 and F_2 in Appendix A2, A3 and A4.

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 *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:

$$x[k] = A_{cl}^{n_1} A_o^{n_2} A_{cl}^{n_3} A_o^{n_4} A_{cl}^{n_5} A_o^{n_6} ... x[0] + w[k] B_{cl} r \quad (8)$$

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

$$w[k+1] = A_{cl} w[k] + F_1 \quad (9)$$

for the ideal samples and

$$w[k+1] = A_o w[k] + F_2 \quad (10)$$

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,

$$x_{ss} = w_{ss} B_{cl} r = (I - A_{cl})^{-1} B_{cl} F_1 r. \quad (11)$$

1. In the architecture shown in Fig. 1, the message m_c is packed with the feedback component of the control input ($Kx[k-1]$). Depending on the delay experienced by m_c, the plant task T_p applies one of the two inputs in (5).

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

$$\begin{aligned} x[k] &= A_{cl}^{n_1} A_o^{n_2} A_{cl}^{n_3} A_o^{n_4} A_{cl}^{n_5} A_o^{n_6} ... x[0] + x_{ss}, \\ \Rightarrow x[k] - x_{ss} &= A_{cl}^{n_1} A_o^{n_2} A_{cl}^{n_3} A_o^{n_4} A_{cl}^{n_5} A_o^{n_6} ... x[0]. \end{aligned} \quad (12)$$

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 η_i and λ_i such that the following inequality holds:

$$\left\| A_i^k \right\| \leq \eta_i \lambda_i^k, \quad (13)$$

where $k \geq 1$. Therefore, we have

$$\left\| A_{cl}^k \right\| \leq \eta_1 \lambda_1^k, \left\| A_o^k \right\| \leq \eta_2 \lambda_2^k, \quad (14)$$

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

$$\left| \frac{a_n}{n+1} \right| \leq \lambda_1 < 1.$$

λ_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:

$$\begin{aligned} \mu(k) &= \frac{n_1 + n_3 + n_5 ...}{n_2 + n_4 + n_6 ...} \\ k &= (\mu(k) + 1) \times \kappa. \end{aligned} \quad (15)$$

The value of $\mu(k)$ denotes the ratio between number of ideal and non-ideal samples (see Fig. 5) and κ 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).

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

C1: $\mu(k) > \mu^* > 1$ for $k > (\mu^* + 1)$ where

$$\mu^* = \frac{\ln \frac{1}{\lambda_2 \eta_1^2 \eta_2}}{\ln \lambda_1} \quad (16)$$

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

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 initial $\mu(k) + 1$ 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, $x[k] = A_{cl}^{n_1} A_o A_{cl}^{\mu(k) - n_1} x[0]$, and by utilizing the properties given by (14), we get,

$$\|x[k]\| \leq \eta_1^2 \eta_2 \lambda_1^{\mu(k)} \lambda_2 \|x[0]\|. \quad (17)$$

Similarly, for $\kappa = 2$, a maximum of four switchings are possible, e.g., $x[k] = A_{cl}^{n_1} A_o A_{cl}^{n_2} A_o A_{cl}^{\mu(k) - n_1 - n_2} x[0]$ and

$$\|x[k]\| \leq \eta_1^3 \eta_2^2 \lambda_1^{2\mu(k)} \lambda_2^2 \|x[0]\|. \quad (18)$$

Hence, for any κ,

$$\|x[k]\| \leq \eta_1(\eta_1\eta_2\lambda_1^{\mu(k)}\lambda_2)^{\kappa}\|x[0]\|. \quad (19)$$

Case I with $\kappa = 1$: In equation (17), if $\eta_1^2\eta_2\lambda_1^{\mu(k)}\lambda_2 = 1$, then,

$$\mu(k) = \mu^* = \frac{\ln \frac{1}{\lambda_2\eta_1^2\eta_2}}{\ln \lambda_1}.$$

As $\lambda_1 < 1$, if condition C1 is satisfied, i.e., $\mu(k) > \mu^*$, then $\eta_1^2\eta_2\lambda_1^{\mu(k)}\lambda_2 < 1$.

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

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

Furthermore, $(\lambda_1^{\mu(k)}\lambda_2)^{\kappa} = (\lambda^*)^k$, i.e., $\lambda_1^{\mu(k)}\lambda_2 = (\lambda^*)^{\mu(k)+1}$ where

$$\ln(\lambda^*) = \frac{\mu(k)\ln\lambda_1 + \ln\lambda_2}{\mu(k)+1}. \quad (20)$$

Clearly, for a given combination of λ_1, λ_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 λ^* if C1 and C2 are satisfied. The stability degree is higher with higher $\mu(k)$ and lower λ^*. □

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:

$$S \geq \frac{\|x[k+\chi] - x_{ss}\|}{\|x[k]\|}, \quad (21)$$

where $0 \leq S \leq 1$, χ 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 χ 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 χ 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 χ 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_o satisfy (14). The switched system is guaranteed to meet the performance requirement if no more than κ_s non-ideal samples occur within any interval of χ consecutive samples, where κ_s is given by

$$\kappa_s = \frac{\ln S - \ln\eta_1 - \chi\ln\lambda_1}{\ln\frac{\eta_1\eta_2\lambda_2}{\lambda_1}} \quad (22)$$

Proof: We assume that the minimum ratio between the number of ideal and non-ideal samples in any χ consecutive samples is μ_s, $\mu_s > 1$ and,

$$\chi = (\mu_s + 1)\kappa_s, \quad (23)$$

where κ_s is the maximum number of non-ideal samples that can occur in any χ consecutive samples. From (19), we get

$$\|x[k+\chi] - x_{ss}\| \leq \eta_1(\eta_1\eta_2\lambda_1^{\mu_s}\lambda_2)^{\kappa_s}\|x[k]\| \quad (24)$$

Next, from (21), we have

$$S = \eta_1(\eta_1\eta_2\lambda_1^{\mu_s}\lambda_2)^{\kappa_s} \quad (25)$$

Combining (23) and (25), we obtain

$$\begin{aligned} \ln S &= \ln\eta_1 + \kappa_s\ln\eta_1\eta_2 + \kappa_s\mu_s\ln\lambda_1 + \kappa_s\ln\lambda_2, \\ \Rightarrow \ln S &= \ln\eta_1 + \chi\ln\lambda_1 + \kappa_s\ln\frac{\eta_1\eta_2\lambda_2}{\lambda_1}, \\ \Rightarrow \kappa_s &= \frac{\ln S - \ln\eta_1 - \chi\ln\lambda_1}{\ln\frac{\eta_1\eta_2\lambda_2}{\lambda_1}}. \end{aligned} \quad (26)$$

3.3 Illustrative Example □

We now illustrate the above results using an example. Let us assume that $\mu^* = 1.9$ 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 $5sec$.

$$\begin{aligned} \dot{v}(t) = A_t v(t) + B_t u(t), y(t) = C_t v(t), \\ v(t) = \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix}, A_t = \begin{bmatrix} 0 & 1.0 & 0 \\ 0 & 0 & 1.0 \\ -6.05 & -5.29 & -0.24 \end{bmatrix}, \\ B_t = \begin{bmatrix} 0 \\ 0 \\ 2.48 \end{bmatrix}, C_t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \end{aligned} \quad (27)$$

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 $v_i(t)$ are measured by the task T_p at the sampling times $t = h_k$ with $k = \{0, 1, 2...\}$, and $h_{k+1} - h_k = h$. $v_i(t)$ 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.

m_i	(S_i, B_i, R_i)	o_i(ms)	p_i(ms)	$[r_{min,i}, r_{max,i}]$	c_i
1	$(12, 0, 2)$	1	20	[0.7,2.7]	4
2	$(13, 0, 2)$	2	20	[0.7,2.7]	4
3	$(14, 2, 4)$	2	40	[1.7,6.7]	3
4	$(16, 0, 4)$	3	50	[1.7,6.7]	3
5	$(17, 7, 8)$	12	100	[1.7,6.7]	3
6	$(17, 4, 8)$	14	100	[1.7,6.66]	3
7	$(20, 2, 4)$	18	80	[2.7,10.7]	5
8	$(25, 0, 2)$	0.1	20	[0.7,2.7]	4
9	$(25, 1, 4)$	17	100	[6.7,26.7]	6
10	$(28, 0, 4)$	21	60	[2,8]	4
11	$(30, 1, 2)$	11	50	[1.7,6.7]	5
12	$(33, 0, 2)$	28	20	[1,4]	4
13	$(34, 0, 1)$	2	20	[0.7,2.7]	4
14	$(38, 0, 2)$	30	40	[1.3,5.3]	5

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 $h = 40$ms 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.

$$\begin{aligned} v[k+1] &= Av[k] + Bu[k], y[k] = Cv[k] \\ A &= \begin{bmatrix} 0.9398 & 0.3412 & 0.0718 \\ -0.4340 & 0.5605 & 0.3241 \\ -1.9603 & -2.1473 & 0.4833 \end{bmatrix}, \\ B &= \begin{bmatrix} 0.0247 \\ 0.1777 \\ 0.8028 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \end{aligned} \quad (28)$$

Stabilizability: For the system in (28), $a_n = 1.9836$. Since $n = 3$ and $|a_n| < (n+1)$, 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 $K = \begin{bmatrix} 1.9833 & 2.3580 & 0.3652 \end{bmatrix}$. The corresponding A_{cl} and A_o are given by the following

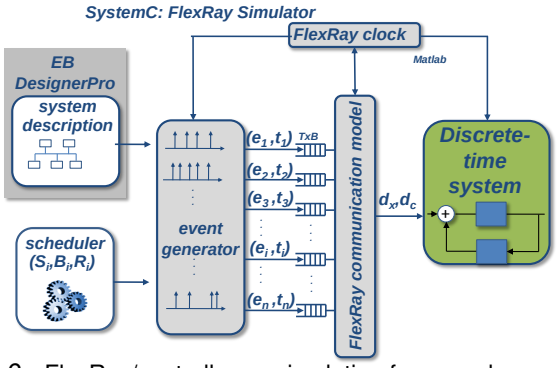

Fig. 6. FlexRay/controller co-simulation framework.

$$\begin{aligned} A_{cl} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.0605 & 0.4878 & -1.4755 & 1.9836 \end{bmatrix}, \\ A_o &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.9092 & -2.2367 & 1.9836 \end{bmatrix}. \end{aligned} \quad (29)$$

Based on the above A_{cl} and A_o, we computed $F_1 = 2.1809$ and $F_2 = 0.4095$ using (44) and (46).

Stability analysis: For the above A_{cl} and A_o, $\eta_1 = 6314.5$, $\eta_2 = 7.2002$, $\lambda_1 = 0.4959$, $\lambda_2 = 1.1646$ and $\mu^* = 27.9895$ from (16).

Performance analysis: To meet the settling time requirement of $5sec$, we chose $\chi = \frac{5}{h} = 125$ samples ($h = 40ms$) and $S = 0.05$ (i.e., 95% disturbance is to be rejected within any 125 samples). Therefore, the performance requirement becomes $\{S, \chi\} = \{0.05, 125\}$. For this performance requirement, $\mu_s = 16.8571$ and $\kappa_s = 7$, i.e., a maximum of 7 non-ideal samples may be tolerated within any 125 consecutive samples, in order to meet the settling time requirement.

4.3 Co-simulation

We developed a SystemC (www.systemc.org) based co-simulation framework to simulate the behavior of the resulting implementation. This simulation framework, as depicted in Fig. 6, is made up of two main modules: the *FlexRay event simulator* to simulate communication delays, and the *discrete-time system model* to simulate the discrete-time system under consideration. The FlexRay simulator consists of several submodules: (i) the *FlexRay clock* provides the *FlexRay communication model* with the actual *slot counter*, *minislot counter* and *cycle counter* values, (ii) an *event generator* generates input event streams based on the system description and (iii) the *FlexRay communication model* implements the FlexRay specification and computes the message delays for the transmitted event streams. Further, the message delays serve as an input to the *discrete-time control system model* in order to compute sensor-to-actuator delays and to simulate the stability and performance of the system.

We used the Elektrobit (EB) Tresos Designer Pro tool [4] to specify the FlexRay bus configuration parameters such as *gdCycle*, *pLatestTx*, the lengths of the static and dynamic segments and other protocol parameters. Additionally, message properties and schedule parameters of existing messages were imported into the simulation framework.

Discrete-time Control System Model: The *discrete-time system model* was implemented in Matlab as a discrete-time

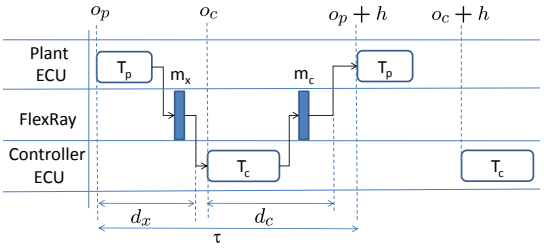

Fig. 7. Computation of sensor-to-actuator delay τ.

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 $o_c > o_p + r_{max,p}$ where $r_{max,p}$ 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

$$\left\lceil \frac{\tau}{h} \right\rceil = \max\{0, \left\lceil \frac{d_x - o_c + o_p}{h} \right\rceil\} + \left\lceil \frac{d_c + o_c - o_p}{h} \right\rceil . \quad (30)$$

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 $d_x > (o_c - o_p)$, the corresponding sensor signal m_x 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 $\left\lceil \frac{o_c - o_p + d_c}{h} \right\rceil$ 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_p^* and r_c^* of the tasks T_p and T_c to be uniformly distributed between $[0.3ms, 1.2ms]$, $o_p = 0.7ms$, $o_c = 2.3ms > o_p + r_{max,p}$ and the periods of the tasks to be 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, $\max\{0, \left\lceil \frac{d_x - o_c + o_p}{h} \right\rceil\} = 0$. The delay variation in the feedback loop stems from the transmission delay of m_c, i.e., from 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_c, B_c, R_c) for m_c) at each simulation run. The simulation time was set to $100sec$ which corresponds to 2500 generated samples with a sampling period $h = 40ms$. During each simulation we plotted the distribution of sensor-to-actuator delay τ 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. τ varies between $\tau_{min} = 3.95ms$ and $\tau_{max} = 44.5ms$. 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-deal samples meets the criterion (26) which is reflected in the resulting output plot Fig. 8 (c) (i.e., settling time is $5sec$ or $\{S, \chi\} = \{0.05, 125\}$).

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 possibility 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 prevalant. 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.

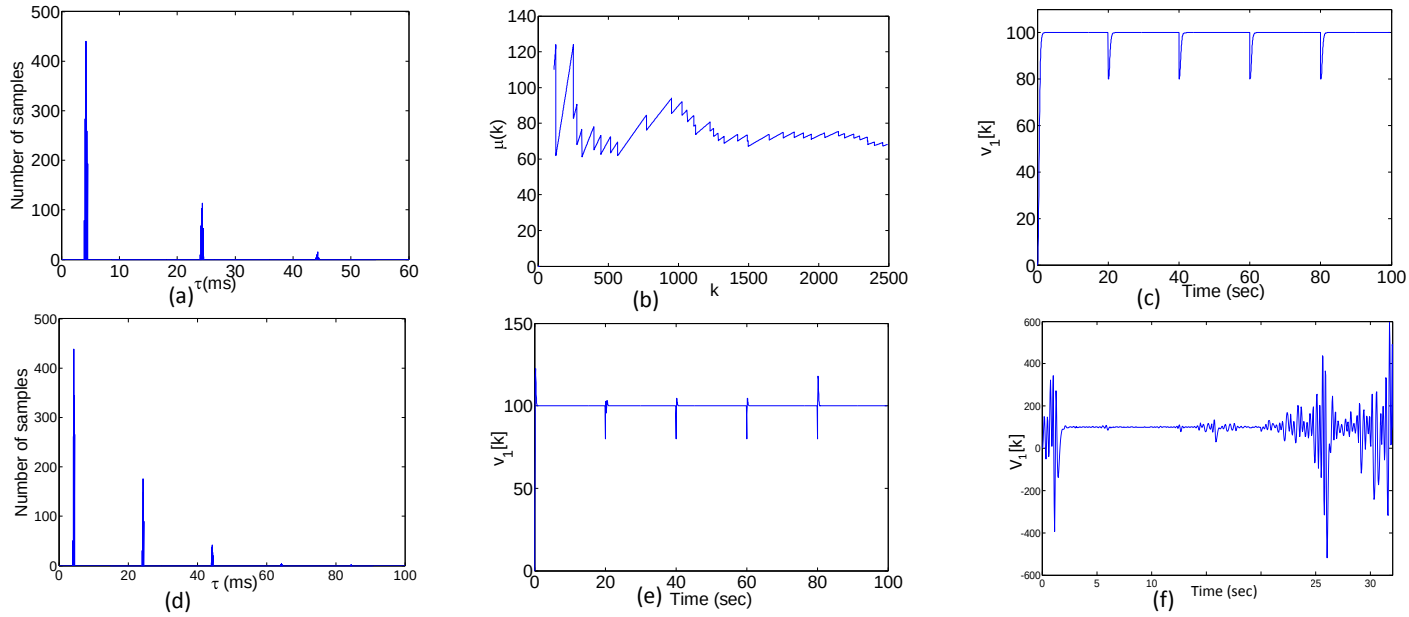

Fig. 8. Results: (a) Delay distribution for $m_c : (45, 0, 4)$ (b) $\mu(k)$ vs k for $m_c : (45, 0, 4)$ (c) $v_1[k]$ with $m_c : (45, 0, 4)$ (d) Delay distribution for $m_c : (47, 0, 4)$ (e) $v_1[k]$ with $m_c : (47, 0, 4)$ (f) $v_1[k]$ with $m_c : (48, 0, 4)$.

REFERENCES

[1] R. Alur and G. Weiss. Regular specifications of resource requirements for embedded control software. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2008.

[2] R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[3] R. C. Dorf and R. H. Bishop. *Modern Control Systems*. Addison Wesley, 1995.

[4] Elektrobit. www.elektrobit.com.

[5] The FlexRay Communications System Specifications, Ver. 2.1. www.flexray.com.

[6] M.E.M.B. Gaid, A. Cela, and Y. Hamam. Optimal integrated control and scheduling of networked control systems with communication constraints: Application to a car suspension system. *IEEE Transactions on Control Systems Technology*, 14(4):776–787, 2006.

[7] S. Hu and W.Y. Yan. Stability robustness of networked control systems with respect to packet loss. *Automatica*, 43(7):1243–1248, 2007.

[8] M. Kauer, S. Steinhorst, D. Goswami, R. Schneider, M. Lukasiewycz, and S. Chakraborty. Formal verification of distributed controllers using time-stamped Event Count Automata. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013.

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

[10] F.L. Lian, J. Moyne, and D. Tilbury. Network design consideration for distributed control systems. *IEEE Transactions on Control Systems Technology*, 10:297–307, 2002.

[11] A.A. Martinez and P. Tabuada. On the benefits of relaxing the periodicity assumption for networked control systems over CAN. In *IEEE Real-Time Systems Symposium (RTSS)*, 2009.

[12] P. Naghshtabrizi and J. P. Hespanha. Analysis of distributed control systems with shared communication and computation resource. In *American Control Conference (ACC)*, 2009.

[13] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam. The wireless control network: A new approach for control over networks. *IEEE Transactions on Automatic Control*, 56(10):2305–2318, 2011.

[14] G. C. Walsh, H. Ye, and L. G. Bushnell. Stability analysis of networked control systemss. *IEEE Transactions on Control Systems Technology*, 10(3):438–446, 2002.

[15] X. Wang and M. Lemmon. Event-triggering in distributed networked control systems. *IEEE Transactions on Automatic Control*, 56(3):586–601, 2011.

[16] G. Weiss and R. Alur. Automata based interfaces for control and scheduling. *Hybrid Systems: Computation and Control (HSCC)*, 2007.

[17] W. Zhang, M. S. Branicky, and S. M. Phillips. Stability of networked control systems. *IEEE Transactions on Control Systems Technology*, 21:8499, 2001.

APPENDIX

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

$$x[k+1] \quad = \quad Ax[k] + BKx[k-1]. \tag{31}$$

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

$$z[k] = Tx[k] \Rightarrow x[k] = T^{-1}z[k], \tag{32}$$

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).

$$\begin{aligned} z[k+1] &= A_c z[k] + B_c u[k], \\ A_c &= \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ a_1 & \cdots & a_{n-1} & a_n \end{bmatrix}, B_c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \end{aligned} \tag{33}$$

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:

$$|a_n| < (n+1). \tag{34}$$

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

$$\begin{aligned} u[k] &= Kx[k-1] = \hat{K}z[k-1] \\ &= \hat{K}_1 z_1[k-1] + \hat{K}_2 z_2[k-1] + \cdots \hat{K}_n z_n[k-1] \\ &= \hat{K}_1 z_1[k-1] + \hat{K}_2 z_1[k] + \cdots \hat{K}_n z_{n-1}[k]. \end{aligned} \tag{35}$$

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)*.

$$\begin{aligned} z_0[k+1] &= z_1[k], z_1[k+1] = z_2[k], z_2[k+1] = z_3[k] \cdots \\ z_n[k+1] &= \hat{K}_1 z_0[k] + (a_1 + \hat{K}_2) z_1[k] \cdots \\ &+ (a_{n-1} + \hat{K}_n) z_{n-1}[k] + a_n z_n[k]. \end{aligned} \quad (36)$$

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

$$\begin{aligned} Z[k+1] &= A_{cl} Z[k] \\ A_{cl} &= \overbrace{\begin{bmatrix} 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ \hat{K}_1 & \cdots & (a_{n-1} + \hat{K}_n) & a_n \end{bmatrix}}^{n+1} \end{aligned} \quad (37)$$

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

$$\lambda^{n+1} - a_n \lambda^n - \cdots - (a_1 + \hat{K}_2)\lambda - \hat{K}_1 = 0. \quad (38)$$

The system (31) is *stablizable* 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 λ^n. Suppose $p = -\begin{bmatrix} p_1 & p_2 \cdots p_{n+1} \end{bmatrix}'$ is the vector consisting poles of the closed-loop system. Then, the characteristic equation of the closed-loop system becomes,

$$\begin{aligned} &(\lambda + p_1)(\lambda + p_2)(\lambda + p_3) \cdots (\lambda + p_{n+1}) = 0 \\ &\Rightarrow \lambda^{n+1} + f_1(p)\lambda^n + f_2(p)\lambda^{n-1} \cdots + f_n(p) = 0. \end{aligned} \quad (39)$$

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

$$\begin{aligned} f_1(p) &= -a_n, f_2(p) = -(a_{n-1} + \hat{K}_n), \cdots \\ f_{n-1}(p) &= -(a_1 + \hat{K}_2), f_n(p) = -\hat{K}_1. \end{aligned} \quad (40)$$

We can see from equation (40) that it is possible to design functions $f_2(p) \cdots f_n(p)$ with appropriate choice of $\hat{K}$ 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,

$$\sum_{i=1}^{n+1} p_i = -a_n. \quad (41)$$

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. □

(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 $p = -\begin{bmatrix} p_1 & p_2 \cdots p_{n+1} \end{bmatrix}'$ 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,

$$\begin{aligned} Z[k+1] &= A_{cl} Z[k] + B_{cl} F_1 r \\ y[k] &= C_{cl} Z[k] \end{aligned} \quad (42)$$

where $B_{cl} = \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix}'$ and $C_{cl} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$ 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,

$$\begin{aligned} Z[k] &= A_{cl} Z[k] + B_{cl} F_1 r \Rightarrow Z[k] = (I - A_{cl})^{-1} B_{cl} F_1 r \\ \Rightarrow y[k] &= C_{cl} Z[k] = C_{cl}(I - A_{cl})^{-1} B_{cl} F_1 r. \end{aligned} \quad (43)$$

For $y[k] = r$ in steady state, we obtain,

$$F_1 = \frac{1}{C_{cl}(I - A_{cl})^{-1} B_{cl}}. \quad (44)$$

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

$$\begin{aligned} Z[k+1] &= A_o Z[k] + B_{cl} F_2 r, \\ y[k] &= C_{cl} x[k], \end{aligned} \quad (45)$$

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

$$F_2 = \frac{1}{C_{cl}(I - A_o)^{-1} B_{cl}}. \quad (46)$$

(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* $\gamma_n \in \Gamma_i$ where $\gamma_n = (B_i + N \times R_i)\ mod\ 64$ with $N \in [0, 1, 2, ...]$, $R_i = 2^r$ 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.