2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)

Synthesizing Performance-aware $\langle m, k \rangle$-firm Control Execution Patterns under Dropped Samples

Sumana Ghosh, Soumyajit Dey, and Pallab Dasgupta
Department of Computer Science & Engineering, Indian Institute of Technology, Kharagpur, India
E-mail: {sumanaghosh, soumya, pallab}@cse.iitkgp.ernet.in

***Abstract*—Performance of control loops often degrade due to various possible environmental disturbances in the control platform, like late arrival of sensor data, or corrupted readings due to transient noise. Such failures usually manifest as *drops* in control loop execution leading to unavailability of fresh control signals. In the existing literature, there is an absence of analytical methods which can compute the required patterns of control execution such that the performance of associated control loops remain satisfactory in the presence of such platform level timing uncertainties. We consider such platform level uncertainties as a collection of window based $\langle m, k \rangle$-firm specifications and synthesize an $\langle m, k \rangle$-firm based input specification for the execution patterns of the loops, so that performance can be provably ensured. Our methodology leverages Büchi automata for modeling platform uncertainties as $\langle m, k \rangle$-firm constraints and bounded model checking approach for synthesizing $\langle m, k \rangle$-firm based input specifications for the control loops.**

***Index Terms*—Embedded Systems, Performance, Execution**

I. Introduction

The control quality of a dynamical system depends on reliable periodic execution of the control law in the given communication and computational architecture. Figure 1 shows the block diagram of a typical networked control system (NCS), where n plants are closed with n controllers over a shared communication network and n controllers share a common electronic control unit (ECU). Sharing of a network or an ECU

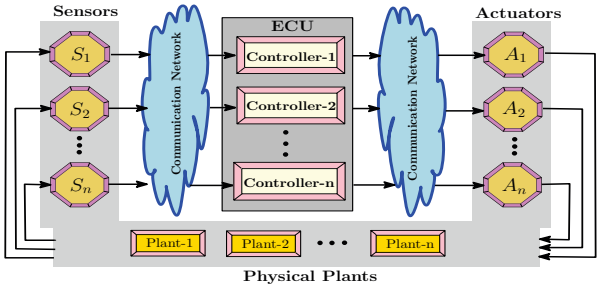

Figure 1: A Typical Networked Control System

among multiple control loops often leads to non-deterministic behavior, such as late arrival of sensor data or packet loss in a sampling interval. It is also possible that sensor readings get corrupted due to transient faults and are rejected as valid sensor data by the system itself. These may result in skipping the execution of control law in that interval. The sampling intervals with such dropped execution are referred to as *dropped samples*. A robust scheduling strategy must therefore consider such dropped samples, and ensure the desired control performance in spite of such failure scenarios.

Authors acknowledge the partial support of the imprint project-FMSAFE for this work.

In recent times, window based specification for modeling the failure process has attracted a lot of attention as a promising alternative to the traditional probabilistic definition of it; see [9], [2] and the references therein. A weakly hard real time constraint is considered in [2], [9] as the window based specification with the form $\mathcal{D} = (d, \Delta)$, where the number of failures is bounded by *maximum* of d in every Δ consecutive sampling windows. As pointed out in [2], such a specification is much more precise when compared with the rate based or probabilistic specification since the later category only represents average information over a long time period and thus allows unbounded consecutive failures which is not suitable in a real time setting. This motivates us to consider the window based specification, $\mathcal{D} = (d, \Delta)$, for modeling the distribution of dropped samples in control loops. In this regard, the works in [2], [9] synthesize stability condition and stabilizing controllers for NCS under such window based failure descriptions. However, such works do not explore the scheduling aspect of controller synthesis, i.e., how to ensure multiple control loops executing on a shared resource are schedulable while respecting timing guarantees for individual loops. Also, these works do not relate the synthesis results with any control performance criterion. Further, they do not consider a combination of such failure specifications which can capture the failure patterns at a higher granularity.

Though the traditional approaches for software based control use a well defined periodic execution of the control loops, existing research [10], [13], [4], [12], [6], [7] has demonstrated that the time between successive execution of the control law need not be regular. This essentially means that the control law computation/update is not performed in certain control loop iterations and the control signal of the previous iterations is simply carried over. The primary motivation of these works to manage the overload of the shared ECU/network by incorporating non-regularity in the execution of control loops. Works in this area provide mathematical guarantees of the desired control performance as long as the execution of the control loops follow certain specifications. Works presented in [10], [4], [12], [6] report design of stable (or robust) controllers given the plant models, the sampling period and $\langle m, k \rangle$-firm constraints that specify the execution patterns for the control loops. The execution of a control loop according to a $\langle m, k \rangle$-firm constraint means that at least m loop executions (i.e. control law computation/update) are successful in any k consecutive sampling intervals.

From our discussion it becomes evident that the robustness of periodic control loops under packet drops specified using window based specifications have been studied [9], [2]. Similarly, synthesis of controllers given their execution patterns using window based specifications have also been studied [10], [4], [12], [6], [7]. However, in a real time setting, it is also

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00019

important to study the execution of control loops following window based specifications in presence of drops induced by external uncertainties which are also specified using window based specifications. As a motivating example let us consider the harmonic oscillator system given in [4]. The control loop is stable for any possible execution pattern which satisfies a window based specification $\langle m = 1, k = 6\rangle$ [4]. Any violation of this window based requirement, for example a sequence of 7 consecutive sampling periods with no update in control law will render the system unstable. In this case, we call $\langle m = 1, k = 6\rangle$ as the *target specification* which needs to be always satisfied, even under external uncertainties. Now, let us consider that such uncertainties lead to control message drops following the window based specification $\mathcal{D} \equiv (d_1 = 1, \Delta_1 = 3) \wedge (d_2 = 2, \Delta_2 = 5)$. That means, maximum 1 drop in every 3 and maximum 2 drops in every 5 consecutive sampling windows. Note that, under such external uncertainties we shall require the control loop to execute *more often* with respect to $\langle m = 1, k = 6\rangle$ so that even under such external uncertainties, the *target specification* $\langle m = 1, k = 6\rangle$ remains satisfied. We refer such robust specification as the *required input specification*.

In light of the existing works on window based specifications for modeling control loop execution and dropped samples, our goal is to synthesize the *required input specifications*, for n control loops, that are robust enough to satisfy the respective *target specifications* under a given distribution, $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_n, \Delta_N)\}$, of dropped samples. We summarize work flow as follows.

• We consider as input a set of control loops with sampling periods, plant dynamics, control matrices and performance requirements in terms of settling time. From this, we compute $\langle m, k\rangle$-firm requirements for individual control loops i.e. the *target specification*.

• For each loop, given a distribution, $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_n, \Delta_N)\}$, of dropped samples, we compute the *required input specifications* for all the loops using a Satisfiability Modulo Theory (SMT)-based solution approach. For this we model window based specifications using Büchi automata following [1], [5] and leverage the theory of conformance checking for the synthesis process. Our framework also accounts for schedulability of the loops.

The paper is organized as follows. Section II describes the preliminaries in control system modeling. Section III presents the mathematical formalism of the target specification, whereas the synthesis framework is described in Section IV. An illustration of the key concepts with suitable case studies and experimental results are reported in Section V. Finally, Section VI concludes the paper.

II. Control System Modeling

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

$$x_p[k+1] = A_p x_p[k] + B_p u[k], \quad y[k] = C_p x_p[k]$$

The plant state at the k-th sampling interval is given by the vector $x_p[k]$, whereas $y[k]$ and $u[k]$ define the output and the control input respectively at that interval. A stabilizing controller, $\Gamma = (A_c, B_c, C_c)$, for P senses the plant output y and decides the control action by adjusting the control variables in u. The feedback control law is represented as a LTI system of the following form:

$$x_c[k+1] = A_c x_c[k] + B_c y[k], \quad u[k] = C_c x_c[k]$$

Here, $x_c[k]$ represents the state of the controller. The matrices (A_p, B_p, C_p) and (A_c, B_c, C_c) are the state transition matrix, the input map, and the output map for plant P, and the controller Γ respectively. With $x = [x_p^\mathsf{T}, x_c^\mathsf{T}]^\mathsf{T}$, the dynamics of the resulting closed loop, $\langle P, \Gamma\rangle$, is defined as follows:

$$x[k+1] = \begin{bmatrix} A_p & B_p C_c \\ B_c C_p & A_c \end{bmatrix} x[k] \;=\; A_1\, x[k] \qquad (1)$$

where A_1 represents the closed loop dynamic matrix for the periodic loop execution.

If a loop execution is dropped in a sampling interval $[k, k+1]$, then the control variables will not change and we get $x_c[k+1] = x_c[k]$. In this interval the closed loop dynamics is therefore: $x[k+1] = A_0\, x[k]$ where the dynamic matrix A_0 is same as A_1 with A_c and B_c being replaced by the identity matrix I, and null matrix, 0, respectively. Hence, the dynamic matrix for the dropped sample is: $A_0 = \begin{bmatrix} A_p & B_p C_c \\ O & I_c \end{bmatrix}$.

For a given control loop $\langle P, \Gamma\rangle$ together with its dynamic matrices $\{A_0, A_1\}$, a *loop execution pattern* is an infinite computation schedule of that control loop, generated by an infinitely repeating finite length string $\rho \in \{A_0, A_1\}^*$. Therefore $\forall k \in \mathbb{N}$, the dynamics of the closed loop system, $\langle P, \Gamma\rangle$, is defined as: $x[k+1] = \rho[k\%l]x[k]$, where l is the length of ρ. For example, according to the loop execution pattern $\rho = A_1A_1A_0A_0A_1A_0$, we have:

$$x[6] = A_0x[5] = A_0A_1x[4] = \ldots = A_0A_1A_0A_0A_1A_1x[0]$$

For notational convenience, we express loop execution pattern as binary string. For e.g., according to $A_1A_1A_0A_0A_1A_0$ we get the loop execution pattern as $\rho = 110010$.

III. Deriving Target Specification from Stability and Performance Requirement

The stability of a control system in the presence of dropped samples has been studied in [14], which provides a sufficient condition on the *rate of dropped samples* guaranteeing the closed loop stability as outlined below.

Theorem 1. [14] *Given a control loop $\langle P, \Gamma\rangle$, with associated Schur stable closed loop dynamics A_1 and unstable open-loop dynamics A_0, the rate, r, of successful execution of the loop over an infinite horizon is bounded by $r_{min} < r \leq 1$ where:*

$$r_{min} = \frac{\log_e(\gamma_0)}{\log_e(\gamma_0) - \log_e(\gamma_1)}, \quad \gamma_0 > \gamma_1, \quad \gamma_1 < 1 \qquad (2)$$

with $\gamma_j = \lambda^2_{max}(A_j)$, $\lambda_{max}(A_j)$ is the maximum eigenvalue of A_j, $j = 0, 1$. □

This means, for a window of l consecutive sampling intervals, total $(l - \lceil l \times r_{min}\rceil)$ dropped samples can exist without compromising the closed loop stability. Therefore, the window based target specification, $\mathcal{T}$, is a $\langle m, l\rangle$-firm constraint with $m = \lceil l \times r_{min}\rceil$.

The target specification can be refined further by modifying the value of r_{min} according to the performance requirement which is captured by the (l, ϵ)-exponential stability criterion [13] of the system.

Definition 1 ((l,ϵ)-Exponential Stability Criterion)**.** *Given a dynamical system of Eq. 1, this criterion requires the system to reduce any error by at least a factor of ϵ in every l sampling intervals, i.e., to meet $\frac{||x[k+l]||}{||x[k]||} < \epsilon$ for every $k \in \mathbb{N}$ and $x[k] \in \mathbb{R}^n$, with $l \in \mathbb{N}$ and $\epsilon \in (0,1]$.* □

Such (l,ϵ)-exponential stability criterion can be easily gleaned from the standard performance index, such as, *settling time* and the desired system norm as follows.

The settling time, τ_s, for a control system specifies that given a reference value χ, for any perturbation σ, the controller is required to move the system from the perturbed state $\chi+\sigma$ back to the reference value χ (with an error margin 2-5 %) within τ_s amount of time. To meet the settling time criterion the number of sampling intervals needed is $l = \lceil \frac{\tau_s}{h} \rceil$, where h is the sampling period. The respective damping factor becomes $\epsilon = \frac{\chi}{\chi+\sigma}$. Thus with the given settling time requirement, the equivalent stability requirement is $(l = \lceil \frac{\tau_s}{h} \rceil, \epsilon = \frac{\chi}{\chi+\sigma})$.

Given such a (l,ϵ)-performance criterion of the system, we can find the corresponding minimum decay rate (α) to be satisfied. Following the standard exponential stability definition [8], we set the decay rate $\alpha = \frac{\log 1/\epsilon}{l}$ and Theorem 1 can be redefined based on this minimum decay rate, α, as follows:

Theorem 2. *For a control loop with the associated closed loop matrix A_1 being Schur stable and r being the rate of successful execution of the loop over an infinite horizon, if there exists a Lyapunov function $V(x(t)) = x'(t)Px(t)$ and scalars α_0, α_1 such that*

$$\alpha_1^r \alpha_0^{1-r} > \alpha, \;\; A_1^T P A_1 \leq \alpha_1^{-2} P, \;\; A_0^T P A_0 \leq \alpha_0^{-2} P \qquad (3)$$

then the system remains exponentially stable with a decay rate greater than α and the bound on the execution rate r becomes, $\frac{2\log_e(\alpha)+\log_e(\gamma_0)}{\log_e(\gamma_0)-\log_e(\gamma_1)} < r \leq 1$, where $\gamma_1 = \alpha_1^{-2} = \lambda^2_{max}(A_1)$, $\gamma_0 = \alpha_0^{-2} = \lambda^2_{max}(A_0)$, $\gamma_1 < 1$, $\gamma_0 > \gamma_1$ and $\lambda_{max}(A_j)$ is the maximum eigenvalue of A_j, $j = 0, 1$.

The $\langle m,l\rangle$-firm constraint for the target specification, $\mathcal{T}$, is then modified accordingly with this new value of $r_{min} = \frac{2\log_e(\alpha)+\log_e(\gamma_0)}{\log_e(\gamma_0)-\log_e(\gamma_1)}$, which resembles the minimum loop execution requirement as per the given (l,ϵ)-performance criterion.

We consider a set of n LTI plants, $\{P_1, \cdots, P_n\}$, and n controllers, $\{\Gamma_1, \cdots, \Gamma_n\}$. Given these plant-controller models, we follow the methodology as outlined above and create the target specifications,$\{\mathcal{T}_1, \cdots, \mathcal{T}_n\}$. With these target specifications, our goal is to synthesize the required input specifications, $\{\mathcal{I}_1, \cdots, \mathcal{I}_n\}$, for n control loops in terms of $\langle m,l\rangle$-firm constraint, that are robust enough to meet $\{\mathcal{T}_1, \cdots, \mathcal{T}_n\}$ respectively, under a given distribution, $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_n, \Delta_N)\}$, of dropped samples.

IV. Synthesis Methodology

Given $\langle m,l\rangle$-firm constraints, $\mathcal{I}$ and $\mathcal{T}$ for the required input and target specifications of a control loop together with the distribution of dropped samples $\mathcal{D}$, we can determine the robustness of $\mathcal{I}$ against $\mathcal{D}$, by checking whether the composite distribution, $\mathcal{D} \otimes \mathcal{I}$, is contained in $\mathcal{T}$. That means, we check whether $f \otimes \rho \in \mathcal{T}$ for any drop pattern $f \in \mathcal{D}$ and loop execution pattern $\rho \in \mathcal{I}$, where $f \otimes \rho$ denotes the composition of f and ρ, i.e., the loop execution pattern resulting from injection of the drops in f over ρ. This can be achieved by searching for a suitable value assignment to $\mathcal{I}$ iteratively and employing SMT-based Bounded Model Checking (BMC) approach for verifying whether $\mathcal{D} \otimes \mathcal{I} \models \mathcal{T}$. This section elucidates the underlying language theoretic basis of this BMC method followed by the equivalent SMT formalism for the verification. In our generalized solution, this method is employed to synthesize the required input specifications of the n control loops.

A. Automata Based Model of $\langle m,l\rangle$-firm Constraints

Using the language theoretic equivalence relation between the window based specification and the Büchi automata as provided in [1], [5], we construct an equivalent Büchi automaton corresponding to a given $\langle m,l\rangle$-firm constraint. Each state q in this automaton represents a l-length sequence $q[1], ..., q[l]$, where $q[i] \in \{0,1\}, 1 \leq i \leq l$ indicates whether the loop is executed or not in the i^{th} sampling interval. Therefore, a state represents the loop execution information in the previous l sampling intervals. In the first few states of the automaton when the l-length history is not available, we use $q[i] = *$ for indicating the lack of execution information. Any loop execution sequence refuting the $\langle m,l\rangle$-firm constraint must have a l-length window where that constraint is violated. This defines the notion of *dead states* as stated below:

Definition 2 (Dead State)**.** *A state $q[1, ..., l]$ is a dead state if corresponding loop execution pattern $q[1], ..., q[l]$ at that state violates the given $\langle m,l\rangle$-firm constraint. A state all of whose successors are dead states is also a dead state.* □

Formally, given a $\langle m,l\rangle$-firm constraint, we define the equivalent Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ as follows:

- $\Sigma = \{0,1\}$ is the input alphabet set, where 1 denotes the loop execution and 0 for loop skip.
- Q is the set of states.
- q_0 is the start state. Note that, $q_0[i] = *, \; \forall i = 1, ..., l$, since initially we have no exact execution information.
- Let, Q_{dead} be the set of dead states. $\delta : Q \times \Sigma \to Q$ is the transition function defined as follows:
  1. For any $q \in Q - Q_{dead}$ and for any $w \in \Sigma$, $\delta(q,w) = q'$ iff $q'[1, ..., l-1] = q[2, ..., l]$ and $q'[l] = w$.
  2. For any $q \in Q_{dead}$, for all $w \in \Sigma$, $\delta(q,w) = q$.
- F is set of final states. All non-dead states are final states.

The Büchi acceptance criterion requires an accepting run to visit some final states infinitely often. Every path which enters into a dead state gets trapped in that dead state forever. Fig. 2 shows the Büchi automaton corresponding to the $\langle m,l\rangle$-firm constraint: $\mathcal{I} = \langle 2,3\rangle$. Each state in the last level represents a loop execution pattern of length 3. The word, $011(011)^\omega$, is one accepted string of this automaton since it never reaches a dead state, and represents the loop execution pattern 011. State labeled as 001, is a dead state since the corresponding loop execution pattern, 001, enables one loop execution in 3 consecutive sampling intervals, violating the $\langle 2,3\rangle$ constraint. For similar reason we get the other dead states as highlighted by gray color. State labeled as $*00$ is a dead state since both of its successor are dead states.

This method of automata construction can also take as input a set of such $\langle m,l\rangle$-firm constraints and construct a Büchi automaton which is equivalent to the conjunction of the constraints [1]. Hence, the same method can be used

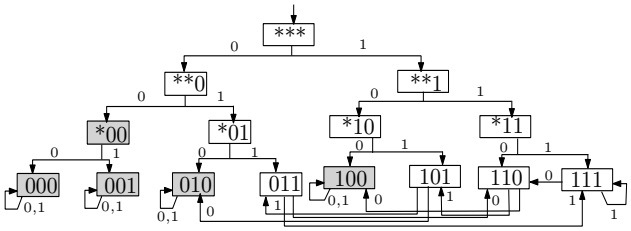

Figure 2: Automaton for $\langle m, l\rangle$-firm constraint: $\mathcal{I} = \langle 2, 3\rangle$

to construct a Büchi automaton corresponding to the $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_n, \Delta_N)\}$ specification of dropped samples. While considering dropped samples as the specification, the input alphabet, $\Sigma = \{0, 1\}$ of the automaton represents the non-occurrence/occurrence of a dropped sample. Each state in that automaton represents a Δ_{max}-length sequence, $q[1], \ldots, q[\Delta_{max}]$, of failure information, where $\Delta_{max} = max\{\Delta_1, \ldots, \Delta_N\}$. In that way, a complex failure specification can be used in our methodology which is not considered in previous works addressing the question of robustness against dropped samples in control loops. The composition of the two automata derived from input specification and drop distribution is defined as follows.

Let, the automata for $\mathcal{I} = \langle m, l\rangle$ and $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_n, \Delta_N)\}$ are $\mathcal{A}_I = (Q_I, \Sigma_I, q_{I_0}, \delta_I, F_I)$ and $\mathcal{A}_D = (Q_D, \Sigma_D, q_{D_0}, \delta_D, F_D)$ respectively. We denote l_{max} as $max(l, \Delta_1, \ldots, \Delta_N)$. The composite automaton is defined as $\mathcal{A}_c = \mathcal{A}_D \otimes \mathcal{A}_I = (Q_c, \Sigma_c, q_{c_0}, \delta_c, F_c)$:

- $\Sigma_c = \{0, 1\}$ is the input alphabet set. The successful and unsuccessful loop execution in each sampling interval is represented by 1 and 0 respectively.
- $Q_c = Q_I \times Q_D$ is the set of states. Each state $q_c \in Q_c$ is a l_{max}-length sequence, $q[1], ..., q[l_{max}]$. For any q_c either $q_c[i] \in \{0, 1\}$ representing the resultant execution outcome of the control loop at i^{th} interval after being mutated by dropped sample at that interval or $q_c[i] = *,\ i \in \{1, \ldots, l_{max}\}$.
- $q_{c_0} = q_{I_0} \times q_{D_0}$ is the start state.
- $F_c = F_I \times F_D$ is the set of final states.
- δ_c is the transition function defined by the following rules:

1. $\dfrac{q_I \xrightarrow{1} q'_I \wedge q_D \xrightarrow{0} q'_D}{(q_I, q_D) \xrightarrow{1} (q'_I, q'_D)}$ [successful execution due to no drop]
2. $\dfrac{q_I \xrightarrow{1} q'_I \wedge q_D \xrightarrow{1} q'_D}{(q_I, q_D) \xrightarrow{0} (q'_I, q'_D)}$ [unsuccessful execution due to drop]
3. $\dfrac{q_I \xrightarrow{0} q'_I \wedge q_{D_1} \xrightarrow{X} q'_I}{(q_I, q_D) \xrightarrow{0} (q'_I, q'_D)}$ [No execution at all - don't care drop]

For any $q_{c_p}, q_{c_q} \in Q_c$ and for any $w \in \Sigma$, $\delta(q_{c_p}, w) = q_{c_q}$ iff $q_{c_q}[1, ..., l_{max} - 1] = q_{c_p}[2, ..., l]$ and $q_{c_q}[l_{max}] = w$. Note that every run of $\mathcal{D} \otimes \mathcal{I}$ is accepted by $\mathcal{A}_c$. To see that no run outside of $\mathcal{D} \otimes \mathcal{I}$ is accepted by $\mathcal{A}_c$, we examine the definition of final state of $\mathcal{A}_c$. Since $F_c = F_I \times F_D$, an accepting run of $\mathcal{A}_c$ is guaranteed to never reach a dead state of either $\mathcal{A}_I$ or $\mathcal{A}_D$. Because, if either of them reach a dead state, that automaton remains trapped in the dead states and thereafter $\mathcal{A}_c$ never returns to a final state.

In our SMT-based BMC approach for verifying the containment of $\mathcal{D} \otimes \mathcal{I}$ in $\mathcal{T}$, we unfold this composite automaton $\mathcal{A}_c$ up to the sequential depth of $l_{max} + 1$ (completeness threshold of BMC). Next section describes the SMT formalism in details.

B. The SMT Based Formulation

Clauses are generated by unfolding the $\langle m, l\rangle$-firm constraints for the input specification, $\mathcal{I}$, target specification, $\mathcal{T}$, and $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_n, \Delta_N)\}$, the specification for dropped samples. For $\mathcal{I}$, we use the indicator variable s_i to denote a loop execution at the i-th sampling interval (that is, s_i is 1 if there is a loop execution, or 0 otherwise). Analogously for $\mathcal{D}$ we use the variables $f_1, f_2, \ldots$. Here, $f_i = 1$ indicates a dropped sample at i-th interval and 0 otherwise. Clauses generated from $\mathcal{I} = \langle m, l\rangle$ are of the following form (for $i = l, l + 1, \ldots$),

$$C^i_{1,I} : m \leq s_{i-(l-1)} + ... + s_{i-1} + s_i \leq l$$

where, $(s_{i-(l-1)} + ... + s_{i-1} + s_i)$ denotes the total number of loop executions in a l-length sampling window up to i-th time step. For each i, we constrain the domain of s_i, by the clause:

$$C^i_{2,I} : (0 \leq s_i \leq 1)$$

In contrast, for $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_N, \Delta_N)\}$, the clauses $C^i_{1,D}$ of following form are constructed $\forall j = 1, 2, \ldots, N$ and $\forall i = \Delta_j, \Delta_j + 1, \ldots$,

$$C^i_{1,D} : \bigwedge_{j=1}^{N} 0 \leq f_{i-(\Delta_j - 1)} + ... + f_{i-1} + f_i \leq d_j$$

whereas $C^i_{2,D}$ is of the form: $(0 \leq f_i \leq 1), \forall i = 1, 2, \ldots$.

To formally verify whether $\mathcal{D} \otimes \mathcal{I} \models \mathcal{T}$, we use another set of indicator variables $p_1, p_2, \ldots$. For capturing the effect of a dropped sample in i-th time step we add clauses of the form:

$$C^i_3 : \quad p_i = s_i \times (1 - f_i)$$

Let the target specification be $\mathcal{T} = \langle \bar{m}, l\rangle$. Since our intent is to search for an assignment to the variables $p_1, p_2, \ldots$ that refutes $\mathcal{T}$, we generate clauses for $i = l, l + 1, \ldots$ as:

$$C^i_4 : \ (\bar{m} > p_{i-(l-1)} + ... + p_{i-1} + p_i) \vee (l < p_{i-(l-1)} + ... + p_{i-1} + p_i)$$

Therefore, the overall clauses progressively generated up to the sequential depth bound of $\eta = l_{max} + 1$ are:

$$\left(\bigwedge_{i=l}^{\eta} C^i_{1,I}\right) \wedge \left(\bigwedge_{i=1}^{\eta} C^i_{2,I}\right) \wedge \left(\bigwedge_{j=1}^{N} \bigwedge_{i=\Delta_j}^{\eta} C^i_{1,D}\right) \wedge \left(\bigwedge_{i=1}^{\eta} C^i_{2,D}\right) \wedge \left(\bigwedge_{i=1}^{\eta} C^i_3\right) \wedge \left(\bigvee_{i=l}^{\eta} C^i_4\right)$$

The progressive generation of the clauses in this manner actually conforms the progressive construction of the composite automaton $\mathcal{A}_c$, together with the verification of $\mathcal{D} \otimes \mathcal{I} \models \mathcal{T}$. We continue designating this verification using a procedure, `Verify_Containment`$(\mathcal{I}, \mathcal{D}, \mathcal{T})$, which returns $True$ on unsatisfiability of $\neg\mathcal{T}$ over $\mathcal{D} \otimes \mathcal{I}$, that means, $\mathcal{D} \otimes \mathcal{I} \models \mathcal{T}$. In next section, we use this procedure to find the required input specifications for n loops.

C. Overall Solution Methodology

The methodology for obtaining the robust input specifications, $\{\mathcal{I}_1, \cdots \mathcal{I}_n\}$, for n control loops is presented in Algorithm 1. It takes the target distributions, $\{\mathcal{T}_1, \cdots, \mathcal{T}_n\}$ as input together with the distribution, $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_N, \Delta_N)\}$, of dropped samples, where $\mathcal{T}_i = \langle \bar{m}_i, l_i\rangle, \forall i = 1, \ldots, n$.

The algorithm employ binary search mechanism to find $\mathcal{I}_i = \langle m_i, l_i\rangle$ for i-th control loop, where $\bar{m}_i \leq m_i \leq l_i$. For each control loop, it runs in a while loop (line 3) by

Algorithm 1: `Find_Input_Specification()`

Input: Target Spec.: $\{\mathcal{T}_1 = \langle \bar{m}_1, l_1\rangle, \cdots, \mathcal{T}_n = \langle \bar{m}_n, l_n\rangle\}$, Dropped Samples Spec.: $\mathcal{D} = \{(d_1, \Delta_1), (d_2, \Delta_2), \ldots, (d_N, \Delta_N)\}$
Output: Input Distributions: $\{\mathcal{I}_1, \cdots, \mathcal{I}_n\}$

1 **for** *each* $i = 1, \cdots, n$ **do**
2 $lb = \bar{m}_i,\ ub = l_i,\ m_i = 0;$
3 **while** $lb \leq ub$ **do**
4 $mid = (ub + lb)/2;$
5 $\mathcal{I}_i = \langle mid, l_i\rangle;$
  `// Check for Robustness of` $\mathcal{I}_i$ `under` $\mathcal{D}$
6 **if** `Verify_Containment`$(\mathcal{I}_i, \mathcal{D}, \mathcal{T}_i) == False$ **then**
7 $lb = mid + 1$; `// Increase the value further`
8 **else**
  `// One solution is found. Search for best.`
9 $m_i = mid$; `// Record current best solution`
10 $ub = mid - 1$;
11 **if** $m_i \neq 0$ && $lb > ub$ **then**
12 $\mathcal{I}_i = \langle m_i, l_i\rangle$; Report_output($\mathcal{I}_i$);
13 Report "No Required Input Specification is found for this loop";

iteratively selecting the value of m_i from the range $[\bar{m}_i, l_i]$ as done in binary search method (lines 4-5). If $\langle m_i, l_i\rangle$ is found as robust, then $\langle q_i, l_i\rangle$ is robust too, $\forall q_i, m_i \leq q_i \leq l_i$ by natural reason. Thus, in lines 8-10 current range is updated for searching the minimum value of $m_i \in [\bar{m}_i, l_i]$ for which the function `Verify_Containment`$(\mathcal{I}_i, \mathcal{D}, \mathcal{T}_i)$ returns $True$. Specifically, in line 9, it stores the current solution to m_i, and in line 10, it updates the upper bound by decreasing it so that searching can be done in lower range. The while loop in line 3 terminates either by picking up the minimum value of m_i (lines 11-12) or by facing a failure situation when no robust solution exists (line-13) for that control loop.

As it is evident from Algorithm 1 that, robustness of the required input specification is achieved by increasing the number of loop executions more than its lower limit as defined in the respective target specification. This over-execution of the control loops may effect their schedulability in the given architectural platform. Hence, we need to check the schedulability of the synthesized input specification before actually reporting it. This is done as follows.

We consider Earliest Deadline First (EDF) algorithm as the underlying scheduling policy. The necessary and sufficient condition for scheduling n control loops executed according to the loop execution patterns following certain $\langle m, l\rangle$-firm constraints to be feasible on a shared preemptive ECU is introduced in [6]. Let $BR(\rho, [t_1, t_2))$ be the cumulative execution requirement of the control tasks initiated according to the loop execution pattern ρ, within the finite time interval $[t_1, t_2)$. Therefore, the total bandwidth requirement of all the control tasks initiated according to $\{\rho_1, \rho_2 \ldots, \rho_n\}$ is,

$$BR(\rho_1, \ldots, \rho_n, [t_1, t_2)) = \sum_{i=1}^{n} BR(\rho_i, [t_1, t_2))$$

The necessary and sufficient condition for n control loops executed according to the loop execution patterns, $\{\rho_1, \ldots, \rho_n\}$, to be feasible on a shared preemptive ECU is [6]:

$$BR(\rho_1, \ldots, \rho_n, [t_1, t_2)) \leq (t_2 - t_1), \quad \forall t_1, t_2,\ t_1 < t_2 \leq t_B$$

where $t_B = lcm(|\rho_1| \times h_1, |\rho_2| \times h_2, \ldots, |\rho_n| \times h_n)$ and h_i be the period of i-th loop. We model this condition using suitable SMT clauses and check EDF-schedulability of the control loops which are executed following the newly generated required input specifications. Since the detailed formalism is out of the scope of this paper, we are skipping it.

V. Experimental Result

We illustrate our results on a set of control loops taken from the automotive domain. The setup comprises a DC motor speed (MS), electromagnetic brake (EMB), electronic wedge brake (EWB), cruise control (CC), and suspension control (SC) system. All the plant models are adopted from [11]. The second order MS system controls the rotational speed of the motor by adjusting the motor terminal voltage. EMB is fifth order system with five state variables as motor current, motor angular velocity, motor angular position, caliper velocity, and caliper position. The control input is the applied voltage on the motor used to control the braking caliper. EWB is a second order system with state variables as position and the velocity of the braking wedge. The position of braking wedge is controlled by applying the force provided by the motor. The linearized third order CC system regulates the vehicle speed at a reference level by adjusting the engine throttle angle. The fourth order SC system has four state variables representing the position, velocity of the car, and position, velocity of the suspension mass. Force is applied to the body by the suspension system to control the car position.

Table I: System Parameters

ID	h_i(ms)	e_i(ms)	τ_{si}(s)	l_i	$r_{min,i}$	$\mathcal{T}_i = \langle \bar{m}_i, l_i\rangle$
MS	40	20	3	15	0.6	$\langle 10, 15\rangle$
EMB	20	7.5	0.5	18	0.58	$\langle 11, 18\rangle$
EWB	10	2.5	0.4	20	0.56	$\langle 12, 20\rangle$
CC	40	17.5	2.3	15	0.61	$\langle 10, 15\rangle$
SC	20	10	1.2	20	0.58	$\langle 12, 20\rangle$

Our choice of the various design parameters, namely, sampling periods (h_i), worst case execution times (WCET) (e_i) and settling time (τ_{si}) are provided in col.2, col.3 and col.4 of Table I respectively. The pattern length, $l_i, \forall i = 1, .., 5$ are computed from τ_{si}, and h_i. From the reference value χ and the maximum perturbation σ, first we compute ϵ_i and then the (l_i, ϵ_i)-exponential stability criteria are used to compute $r_{min,i}$ for $i = 1, ..., 5$, by applying Theorem 2. Note that, the running time and memory requirement of the containment checking phase (Algorithm 1, line 6) depends on the size of the automata constructed which again is exponential in l_i-s. For any control loop, the stability criterion (l, ϵ) can always be replaced by a stricter criterion $(\frac{l}{q}, \epsilon^{\frac{1}{q}})$, for some $q > 1$, with a lesser number of sampling intervals, such that satisfaction of $(\frac{l}{q}, \epsilon^{\frac{1}{q}})$ always implies satisfaction of (l, ϵ). In case, the algorithm reports memory or time overflow, it is restarted with a scaled criterion $(\frac{l}{q}, \epsilon^{\frac{1}{q}})$ with suitable choice of q. Col.5 of Table I reports l_i values actually used after such scaling. The reference value, χ, for these five plant models are considered as, $0.5rad/s, 2mm, 2.5mm, 50km/hr$, and $0.02m$ respectively. The chosen values for maximum perturbation, σ, for them are $4.5rad/s, 8mm, 10mm, 160km/hr$, and $4.98m$ respectively. The values of $r_{min,i}$ are calculated using these values as reported in col.6 of Table I. Finally, col.7 shows the target specification, $\mathcal{T}_i$ obtained from l_i and $r_{min,i}$ as discussed in Theorem 2.

We run Algorithm 1 for each of these five loops under three different distributions of dropped samples identified as three

Table II: Results on varying failure distribution $\mathcal{D}$

Scenario 1: $\mathcal{D} \equiv (2,10) \wedge (4,18) \wedge (6,25)$			Scenario 2: $\mathcal{D} \equiv (3,20)$			Scenario 3: $\mathcal{D} \equiv (1,15) \wedge (2,28)$		
ID	$\mathcal{I}_i = \langle m_i, l_i \rangle$	Time (s)	ID	$\mathcal{I}_i = \langle m_i, l_i \rangle$	Time (s)	ID	$\mathcal{I}_i = \langle m_i, l_i \rangle$	Time (s)
MS	$\langle 14, 15 \rangle$	112.75	MS	$\langle 13, 15 \rangle$	435.89	MS	$\langle 11, 15 \rangle$	325
EMB	$\langle 14, 18 \rangle$	320.15	EMB	$\langle 13, 18 \rangle$	452.13	EMB	$\langle 12, 18 \rangle$	332.12
EWB	$\langle 16, 20 \rangle$	225.37	EWB	$\langle 15, 20 \rangle$	504.55	EWB	$\langle 14, 20 \rangle$	441.15
CC	$\langle 14, 15 \rangle$	118.02	CC	$\langle 13, 15 \rangle$	443.34	CC	$\langle 11, 15 \rangle$	349.9
SC	$\langle 16, 20 \rangle$	304.05	SC	$\langle 15, 20 \rangle$	512.53	SC	$\langle 14, 20 \rangle$	423.05

Table III: Outcome of Schedulability Test

Scenario 1: $\mathcal{D} \equiv (2,10) \wedge (4,18) \wedge (6,25)$			Scenario 2: $\mathcal{D} \equiv (3,20)$			Scenario 3: $\mathcal{D} \equiv (1,15) \wedge (2,28)$		
ID	$\mathcal{I}_i = \langle m_i, l_i \rangle$	Schedulable ?	ID	$\mathcal{I}_i = \langle m_i, l_i \rangle$	Schedulable ?	ID	$\mathcal{I}_i = \langle m_i, l_i \rangle$	Schedulable ?
MS	$\langle 14, 15 \rangle$	Not Sch.	MS	$\langle 13, 15 \rangle$	Sch.	MS	$\langle 11, 15 \rangle$	Sch.
EMB	$\langle 14, 18 \rangle$		EMB	$\langle 13, 18 \rangle$		EMB	$\langle 12, 18 \rangle$	
EWB	$\langle 16, 20 \rangle$		EWB	$\langle 15, 20 \rangle$		EWB	$\langle 14, 20 \rangle$	
EMB	$\langle 14, 18 \rangle$	Sch.	EMB	$\langle 13, 18 \rangle$	Sch.	EMB	$\langle 12, 18 \rangle$	Sch.
EWB	$\langle 16, 20 \rangle$		EWB	$\langle 15, 20 \rangle$		EWB	$\langle 14, 20 \rangle$	
CC	$\langle 14, 15 \rangle$		CC	$\langle 13, 15 \rangle$		CC	$\langle 11, 15 \rangle$	
EWB	$\langle 16, 20 \rangle$	Not Sch.	EWB	$\langle 15, 20 \rangle$	Not Sch.	EWB	$\langle 14, 20 \rangle$	Sch.
CC	$\langle 14, 15 \rangle$		CC	$\langle 13, 15 \rangle$		CC	$\langle 11, 15 \rangle$	
SC	$\langle 16, 20 \rangle$		SC	$\langle 15, 20 \rangle$		SC	$\langle 14, 20 \rangle$	

different scenarios in Table II. In the table, col.2, col.5, and col.8 show the required input specifications ($\mathcal{I}_i$) obtained by running Algorithm 1 for the five loops under these three failure scenarios, whereas col.3, col.6, and col.9 report the time taken by Algorithm 1 to generate those input specifications. Note that, the distribution of dropped samples become sparse as we move from Scenario 1 to Scenario 3. Consequently, the $\langle m, l \rangle$-firm constraints synthesized for the loops have less number of loop executions (i.e. m) for the same control loop as we move from Scenario 1 to Scenario 3. For e.g., in case of MS, the value of m goes down from 14 to 11.

As discussed earlier, our framework can also check the schedulability of a set of control loops given their WCETs and other parameters. The EDF-schedulability is tested over three separate sets of control loops taking three arbitrary plants at a time from the given set of five models, under those above mentioned three failure scenarios. We consider the set of control loops, namely, *Set-1:* {*MS,EMB,EWB*}, *Set-2:* {*EMB,EWB,CC*}, and *Set-3:* {*EWB,CC,SC*}. The outcome of the schedulability test on each of those nine different cases are reported in Table III. For any case, if no EDF-schedulable combination is found then it is reported as `Not Sch.`.

Under Scenario 1, the control loops in *Set-1* becomes unschedulable as reported in Table III. For other two failure scenarios, this set is schedulable due to less number of loop executions inside each window length. On the other hand, *Set-2* is schedulable for all the failure scenarios. Note that, *Set-1* and *Set-2* differ by only one plant which is MS in former one and CC in later one, but the schedulability answer is different mainly because of the different bandwidth requirement of MS and CC (see Table I). As reported in Table III, *Set-3* becomes unschedulable under first two failure scenarios due to more loop executions inside each window length as specified by the respective required input specifications and the WCET requirement of the control loops.

All experiments have been performed on a 3.10 GHz Intel Core-i5 processor with 4 GB of memory, running 64-bit Ubuntu 14.04. We use Z3 [3] as the back-end SMT solver and Z3py, a Z3 API in Python as a front-end modeling language.

VI. Conclusion

In this work we present a framework for modeling and synthesizing the robust input specification for the control loops in terms of $\langle m, k \rangle$ -firm constraints. Our framework is useful for formally correlating control performance with control architecture platform mapping in resource constrained environments where environmental uncertainties and their effects in the form of dropped samples are formally quantifiable. Such analysis is also useful for estimation of redundancy requirements very early in the design flow of complex embedded control systems.

References

[1] K. Banerjee and P. Dasgupta. Acceptance and random generation of event sequences under real time calculus constraints. In *Proc. DATE*, pages 1–6, 2014.

[2] R. Blind and F. Allgwer. Towards networked control systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process. In *Proc. CDC*, pages 7510–7515, Dec 2015.

[3] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS*, pages 337–340. Springer, 2008.

[4] F. Flavia et al. Optimal on-line (m, k)-firm constraint assignment for real-time control tasks based on plant state information. In *Proc. ETFA*, pages 908–915, 2008.

[5] S. Ghosh and P. Dasgupta. Formal methods for pattern based reliability analysis in embedded systems. In *Proc. VLSID*, pages 192–197, 2015.

[6] S. Ghosh, S. Dutta, S. Dey, and P. Dasgupta. A structured methodology for pattern based adaptive scheduling in embedded control. *ACM Trans. Embed. Comput. Syst.*, 16(5s):189:1–189:22, Sept. 2017.

[7] A. Gujarati, M. Nasri, and B. B. Brandenburg. Quantifying the Resiliency of Fail-Operational Real-Time Networked Control Systems. In *ECRTS 2018*, volume 106, pages 16:1–16:24.

[8] A. Hassibi et al. Control of asynchronous dynamical systems with rate constraints on events. In *Proc. CDC*, volume 2, pages 1345–1351, 1999.

[9] S. Linsenmayer and F. Allgower. Stabilization of networked control systems with weakly hard real-time dropout description. In *Proc. CDC*, pages 4765–4770, Dec 2017.

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

[11] D. Roy et al. Multi-objective co-optimization of flexray-based distributed control systems. In *Proc. RTAS*, pages 1–12, 2016.

[12] D. Soudbakhsh, L. T. X. Phan, O. Sokolsky, I. Lee, and A. Annaswamy. Co-design of control and platform with dropped signals. In *Proc. ICCPS*, pages 129–140, April 2013.

[13] G. Weiss and R. Alur. Automata based interfaces for control and scheduling. In *Proc. HSCC*, pages 601–613, 2007.

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