2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Self-Triggered Control for Near-Maximal Average Inter-Sample Time

Gabriel de A. Gleizer, Khushraj Madnani and Manuel Mazo Jr.

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9682986

***Abstract*— Self-triggered control (STC) is a sample-and-hold control method aimed at reducing communications in networked-control systems; however, existing STC mechanisms often maximize how late the next sample is, thus not optimizing sampling performance in the long-term. In this work, we devise a method to construct self-triggered policies that provide near-maximal average inter-sample time (AIST) while respecting given control performance constraints. To achieve this, we rely on finite-state abstractions of a reference event-triggered control, while also allowing earlier samples. These early triggers constitute controllable actions of the abstraction, for which an AIST-maximizing strategy can be obtained by solving a mean-payoff game. We provide optimality bounds, and how to further improve them through abstraction refinement techniques.**

I. INTRODUCTION

As networked control systems (NCSs) become the norm in the automation industry, significant attention has been given to sampling and scheduling mechanisms for control. In particular, self-triggered control (STC) [1] was proposed as a method to reduce required resource usage. The idea behind STC is that the controller decides the next sampling instant in an aperiodic fashion. In [2], Tabuada proposed a related method called event-triggered control (ETC), finally establishing a rigorous treatment for stabilization under such aperiodic methods. In ETC, a function between actual state measurements and the past samples is monitored on the sensor side, and data is only transmitted upon an occurrence of a significant event. After [2], much work was dedicated to the design of triggering mechanisms that reduce sampling while guaranteeing stability and performance criteria [3], [4], [5]. STC was then typically designed by estimating when such events would happen on the controller (e.g., [6]).

Among all the aforementioned works, the commonality lies on the philosophy of sampling: it must be as late as possible. Quite often, the *triggering condition* is a surrogate of an underlying Lyapunov function condition, either ensuring its decrease in continuous time [2], its boundedness with respect to a decaying function [3], [6], [7], or its decrease in average [4]. The sampling happens as soon as the condition is violated, that is, as late as possible. Posing this sampling strategy as a mechanism to maximize *average inter-sample time* (AIST) — a natural metric of sampling performance —, this corresponds to a *greedy* approach for optimization: one maximizes the short-term reward hoping that this brings long-term maximization. It is not surprising that this approach is very rarely optimal, and we have no reason to believe this would be the case with sampling. Aiming at performance improvements, Antunes et al [8] addressed the problem of co-designing controllers and sampling strategies that minimize a quadratic control performance while ensuring that the AIST is not smaller than some reference periodic controller. Before the ETC era, the problem of minimizing a weighted sum of quadratic performance and AIST for linear controllers with random disturbances was addressed in [9], using an elegant dynamic programming approach; unfortunately, this approach lacks the practicality of its STC successors, as the decision to sample or not requires solving an online recursion involving complicated operators. Therefore, we choose to focus on the more practical Lyapunov-based ETC and STC strategies, even though their simplicity comes with a cost: the emerging traffic patterns are often erratic. In fact, even computing the AIST of an ETC implementation is challenging, and only recently been made possible in [10].

In this work, we address the problem of synthesizing a self-triggered mechanism for state feedback of linear systems to maximize the AIST, while ensuring the same control performance as a reference periodic event-triggered control (PETC, [11]). In PETC, events are checked only periodically, which is a more practical version of ETC. The improvement over PETC's AIST is attained by considering predicted PETC triggering times as *deadlines*: sampling earlier always ensures equal or better control performance, and done in the right way it can provide long-term benefits in terms of AIST. To determine when these long-term rewards are attained, we build on our recent work in [10] by using finite-state abstractions [12] of the reference PETC, and appending weights on the transitions systems, obtaining *weighted automata* [13]. Finite-state abstractions can simulate all possible behaviors of the concrete systems, while permitting computations that are otherwise intractable or impossible in infinite-state systems. By augmenting the finite-state simulations with *controllable* inter-event times up to the PETC deadline, we obtain an alternating simulation relation which also retains quantitative properties. As we will see, solving a *mean-payoff game* on the abstraction gives a strategy that approximately maximizes the AIST; the value of the abstraction game is in fact a lower bound to the optimal early-triggering strategy, while computing the (cooperative) maximum AIST of the system gives an upper bound to it. Our approach gives a state-dependent sampling strategy (SDSS) that requires predicting the next l inter-sample times that the reference PETC would generate, l being a chosen parameter. As we will see in a numerical example, even $l = 1$ can provide massive improvements to the closed-loop system's AIST by simply using the strategy obtained from the mean-payoff game. The computational cost of this prediction is proportional to l,

This work is supported by the European Research Council through the SENTIENT project (ERC-2017-STG #755953).

G. de A. Gleizer, K. Madnani and M. Mazo Jr. are with the Delft Center for Systems and Control, Delft Technical University, 2628 CD Delft, The Netherlands {g.gleizer, k.n.madnani-1, m.mazo}@tudelft.nl

making our approach implementable on hardware with limited capabilities or time-critical applications. In addition, a major benefit of using abstractions is that the methodology is general in the sense that it can be used for more complex control specifications; and while we focus on the control of a single linear system, extensions to multiple controllers sharing a network [14], [15] and nonlinear systems [16] are possible using existing abstraction methods.

Notation. We denote by $\mathbb{N}_0$ the set of natural numbers including zero, $\mathbb{N} \coloneqq \mathbb{N}_0 \setminus \{0\}$, $\mathbb{N}_{\leq n} \coloneqq \{1, 2, ..., n\}$, by $\mathbb{Q}$ the set of rational numbers, and by $\mathbb{R}_+$ the set of non-negative reals. For a set $\mathcal{X} \subseteq \Omega$, we denote by $\bar{\mathcal{X}}$ its complement $\Omega \setminus \mathcal{X}$, and by $|\mathcal{X}|$ its cardinality. We often use a string notation for sequences, e.g., $\sigma = abc$ reads $\sigma(1) = a, \sigma(2) = b, \sigma(3) = c$. Powers and concatenations work as expected, e.g., $\sigma^2 = \sigma\sigma = abcabc$. We denote by $\mathcal{X}^+$ (resp. $\mathcal{X}^\omega$) the sets of finite (resp. infinite) sequences with elements on $\mathcal{X}$. The inverse of a relation $\mathcal{R} \subseteq \mathcal{X}_a \times \mathcal{X}_b$ is denoted by $\mathcal{R}^{-1} = \{(x_b, x_a) \in \mathcal{X}_b \times \mathcal{X}_a \mid (x_a, x_b) \in \mathcal{R}\}$. Finally, $\pi_{\mathcal{R}}(\mathcal{X}_a) \coloneqq \{x_b \in \mathcal{X}_b \mid (x_a, x_b) \in \mathcal{R} \text{ for some } x_a \in \mathcal{X}_a\}$ denotes the natural projection $\mathcal{X}_a$ onto $\mathcal{X}_b$. Given an autonomous system $\dot{\boldsymbol{\xi}}(t) = f(\boldsymbol{\xi}(t))$, we say that the origin is globally exponentially stable (GES) if there exist $M < \infty$ and $b > 0$ such that every solution of the system satisfies $|\boldsymbol{\xi}(t)| \leq M\mathrm{e}^{-bt}|\boldsymbol{\xi}(0)|$ for every initial state $\boldsymbol{\xi}(0)$. When the inital state is not obvious from context, we use $\boldsymbol{\xi}_{\boldsymbol{x}}(t)$ to denote a trajectory from initial state $\boldsymbol{\xi}(0) = \boldsymbol{x}$.

II. PROBLEM FORMULATION

Consider a linear plant controlled with sample-and-hold state feedback described by

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{A}\boldsymbol{\xi}(t) + \boldsymbol{B}\boldsymbol{K}\hat{\boldsymbol{\xi}}(t), \tag{1}$$

where $\boldsymbol{\xi}(t) \in \mathbb{R}^{n_\mathrm{x}}$ is the plant's state, $\hat{\boldsymbol{\xi}}(t) \in \mathbb{R}^{n_\mathrm{x}}$ is the measurement of the state available to the controller, $\boldsymbol{K}\hat{\boldsymbol{\xi}}(t) \in \mathbb{R}^{n_\mathrm{u}}$ is the control input, n_x and n_u are the state-space and input-space dimensions, respectively, and $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{K}$ are matrices of appropriate dimensions. We consider a zero-order hold mechanism: let $t_i \in \mathbb{R}_+, i \in \mathbb{N}_0$ be a sequence of sampling times, with $t_0 = 0$ and $t_{i+1} - t_i > \varepsilon$ for some $\varepsilon > 0$; then $\hat{\boldsymbol{\xi}}(t) = \boldsymbol{\xi}(t_i), \forall t \in [t_i, t_{i+1})$. In STC, the controller uses available information at time t_i to determine the next sampling time t_{i+1}. Because the system of interest is time-invariant, it suffices to determine the next inter-sample time (IST) $\tau_i \coloneqq t_{i+1} - t_i$. In general terms, when full-state information is available, STC is a controller composed with a *state-dependent sampling strategy* (SDSS) $s : (\mathbb{R}^{n_\mathrm{x}})^i \to \mathbb{R}_+$ that can consider the first i state samples to determine τ_i. As we consider mean-payoff objectives, which admit *positional* (or *static*) strategies [17], we focus on the particular case of *static* state-dependent sampling strategies, which have the form $s : R^{n_\mathrm{x}} \to \mathbb{R}_+$. The algorithm that dictates τ_i must guarantee given performance specifications while being fast enough to execute in real time, depending on the application.

Given system (1), and an initial state $\boldsymbol{x} \coloneqq \boldsymbol{\xi}(0)$, an SDSS determines a unique state trajectory $\boldsymbol{\xi}_{\boldsymbol{x}}(t)$, as well as the sequence of ISTs τ_i, since $\tau_0 = s(\boldsymbol{x})$ and the following recursion on i holds:

$$\begin{aligned} \boldsymbol{\xi}_{\boldsymbol{x}}(t) &= \boldsymbol{M}(t - t_i)\boldsymbol{\xi}_{\boldsymbol{x}}(t_i),\ \forall t \in [t_i, t_{i+1}] \\ t_{i+1} &= t_i + \tau_i, \\ \tau_i &= s(\boldsymbol{\xi}_{\boldsymbol{x}}(t_i)), \end{aligned} \tag{2}$$

where $\boldsymbol{M}(t) \coloneqq \boldsymbol{A}_\mathrm{d}(t) + \boldsymbol{B}_\mathrm{d}(t)\boldsymbol{K} \coloneqq \mathrm{e}^{\boldsymbol{A}t} + \int_0^t \mathrm{e}^{\boldsymbol{A}\tau}\mathrm{d}\tau\boldsymbol{B}\boldsymbol{K}$ is the state transition matrix under the held control input. Thus, for convenience, we denote the unique IST sequence of a given SDSS s from a given initial condition $\boldsymbol{x}$ by $\tau_i(\boldsymbol{x}; s)$. With this, we can define the average IST (AIST) from $\boldsymbol{x}$ as

$$\mathrm{AIST}(\boldsymbol{x}; s) \coloneqq \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \tau_i(\boldsymbol{x}; s),$$

and the smallest AIST (SAIST) across all initial states as

$$\mathrm{SAIST}(s) \coloneqq \inf_{\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}}} \mathrm{AIST}(\boldsymbol{x}; s). \tag{3}$$

The SAIST of a given SDSS gives its sampling performance, which we are interested in maximizing.

A standard approach to SDSS design is predicting the time at which relevant Lyapunov-based event would happen, as in [6], [18], [5]. We consider *quadratic triggering conditions*, which encompass the most common ones in the ETC and STC literature, see [11], where the idea is to design a matrix $\boldsymbol{Q} \in \mathbb{S}^{2n_\mathrm{x}}$ such that

$$\begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix}^\mathsf{T} \boldsymbol{Q} \begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix} \leq 0 \implies \dot{V}(\boldsymbol{x}) \leq -aV(\boldsymbol{x}) \tag{4}$$

for some $a > 0$, where $V : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}_+$ is a Lyapunov function. That is, while the triggering condition (LHS above) is satisfied, some Lyapunov condition is also satisfied, ensuring stability and performance specifications are met. In ETC, this condition is checked continuously and the sample occurs as soon as it is violated. In PETC, this condition is checked every h time units, and further Lyapunov analysis needs to be made to ensure specifications are met, see [11]. In STC, instead of checking events, the controller predicts them at every h units and typically chooses the last one such that the condition is still met:

$$t_{i+1} = \sup\{t \in h\mathbb{N} \mid t > t_i \text{ and } c(t - t_i, \boldsymbol{\xi}(t), \hat{\boldsymbol{\xi}}(t))\}, \tag{5}$$

$$c(\tau, \boldsymbol{x}, \hat{\boldsymbol{x}}) \coloneqq \begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix}^\mathsf{T} \boldsymbol{Q} \begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix} \leq 0 \text{ or } \ \tau \leq \bar{\tau}, \tag{6}$$

where $\bar{\tau}$ is a chosen maximum inter-event time that ensures the search space is finite.

Clearly, the ETC and STC strategies described above are greedy sampling strategies: they maximize the next IST while ensuring the triggering condition is not violated. Therefore, we can use the ETC-generated IST as a state-dependent *deadline* $d : \mathbb{R}^{n_\mathrm{x}} \to \mathcal{T}$ of an SDSS:

$$d(\boldsymbol{x}) \coloneqq \max\{\tau \in \mathcal{T} \mid c(\tau, \boldsymbol{\xi}_{\boldsymbol{x}}(\tau), \boldsymbol{x})\}, \tag{7}$$

where $\mathcal{T} = \{h, 2h, ..., \bar{\tau}\}$ is the set of possible inter-event times, and $\bar{\tau} = hK$, with $K \in \mathbb{N}$. The question that naturally arises is whether sampling earlier than this deadline can provide long-term benefits in terms of average inter-sample time. This possibility is exactly what we exploit in this work.

Objective of this paper. Hereafter, we shall refer to an SDSS $s^* : \mathbb{R}^{n_x} \to \mathcal{T}$ that respects the deadlines in Eq. (7) as an *early-triggering SDSS*; doing so provides the RHS of Eq. 4 as a performance guarantee. Let the reference PETC strategy be s; our main objective is to design an early-triggering SDSS s^* that ensures $\text{SAIST}(s^*) \geq \text{SAIST}(s) + v$, i.e, its SAIST is bigger than that of the PETC by a value $v > 0$. In addition, we want to estimate how far s^* is from the optimal strategy, i.e., find ϵ such that $\text{SAIST}(s^*) \geq \text{SAIST}(s') - \epsilon$ for all early triggering SDSSs s'.

III. Finding an SDSS through abstractions

To solve our main problem, we adapt the abstraction framework of [12] to include quantitative capabilities; i.e., we abstract our system as a *weighted automaton*.

A. Transition systems

In [12], Tabuada gives a generalized notion of transition system. Here, we include a weight function as in [13] for the quantitative aspect.

Definition 1 (Weighted transition system): A system $\mathcal{S}$ is a tuple $(\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{E}, \mathcal{Y}, H, \gamma)$ where:

- $\mathcal{X}$ is the set of states,
- $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states,
- $\mathcal{U}$ is the set of actions,
- $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{U} \times \mathcal{X}$ is the set of edges (or transitions),
- $\mathcal{Y}$ is the set of outputs, and
- $H : \mathcal{X} \to \mathcal{Y}$ is the output map.
- $\gamma : \mathcal{E} \to \mathbb{Q}$ is the weight function.

A system is said to be finite (infinite) state when the cardinality of $\mathcal{X}$ is finite (infinite), and it is said to be just finite if both $\mathcal{X}$ and $\mathcal{U}$ are finite. A transition in $\mathcal{E}$ is denoted by a triple (x, u, x'). We define $\text{Post}_{\mathcal{S}}(x) \coloneqq \{x' \mid \exists u : (x, u, x') \in \mathcal{E}\}$ as the set of states that can be reached from x in one step. System $\mathcal{S}$ is said to be *non-blocking* if $\forall x \in \mathcal{X}, \text{Post}_{\mathcal{S}}(x) \neq \emptyset$. We call $x_0 u_0 x_1 u_1 x_2 ...$ an *infinite internal behavior*, or *run* of $\mathcal{S}$ if $x_0 \in \mathcal{X}_0$ and $(x_i, u_i, x_{i+1}) \in \mathcal{E}$ for all $i \in \mathbb{N}$, $y_0 y_1 ...$ its corresponding *infinite external behavior*, or *trace*, if $H(x_i) = y_i$ for all $i \in \mathbb{N}$, and $v_0 v_1 ...$ its *value trace* if $v_i = \gamma(x_i, u_i, x_{i+1})$ for all $i \in \mathbb{N}$. We denote by $\gamma_{\mathcal{S}}(r)$ the value trace of $r = x_0 u_0 x_1 ...$ (in the case above, $\gamma_{\mathcal{S}}(r) = v_0 v_1 ...$), by $\mathcal{V}^{\omega}_x(\mathcal{S})$ the set of all possible value traces of $\mathcal{S}$ starting from state x, and by $\mathcal{V}^{\omega}(\mathcal{S}) \coloneqq \bigcup_{x \in \mathcal{X}_0} \mathcal{V}^{\omega}_x(\mathcal{S})$ the set of all possible value traces of $\mathcal{S}$. We denote by $\gamma(\mathcal{E})$ the set of all possible weight valuations of $\mathcal{S}$, which is guaranteed to be finite if $\mathcal{S}$ is finite. A state x is called *reachable* if there exists a run r of $\mathcal{S}$ containing x; for the rest of this paper, we assume every state is reachable. This does not affect generality as one can always remove unreachable states from a transition system without changing its behavior. We denote by $U(x) \coloneqq \{u \mid \exists x' \in \mathcal{X} : (x, u, x') \in \mathcal{E}\}$ the action set of $x \in \mathcal{X}$.

A system $\mathcal{S}_b$ is loosely called an *abstraction* of another system $\mathcal{S}_a$ if it retains some properties of $\mathcal{S}_a$ while being simpler. Typically, we use the term abstraction to denote a finite system $\mathcal{S}_a$ that *simulates* or is *alternatingly simulated* by a possibly infinite system $\mathcal{S}_b$ (see [12] for a thorough exposition of these concepts). When one is interested in verification, simulation is used; if one wants to use $\mathcal{S}_a$ to find a strategy for $\mathcal{S}_b$ that ensures certain properties, the relation of interest is alternating simulation. We use the following definitions, which are amenable to quantitative properties.

Definition 2 (Simulation relation): Consider two systems $\mathcal{S}_a$ and $\mathcal{S}_b$ with $\gamma_a(\mathcal{E}_a) = \gamma_b(\mathcal{E}_b)$. A relation $\mathcal{R} \subseteq \mathcal{X}_a \times \mathcal{X}_b$ is a *simulation relation* from $\mathcal{S}_a$ to $\mathcal{S}_b$ if the following conditions are satisfied:

i) for every $x_{a0} \in \mathcal{X}_{a0}$, there exists $x_{b0} \in \mathcal{X}_{b0}$ with $(x_{a0}, x_{b0}) \in \mathcal{R}$;
ii) for every $(x_a, x_b) \in \mathcal{R}$, we have that $(x_a, u_a, x'_a) \in \mathcal{E}_a$ implies the existence of $(x_b, u_b, x'_b) \in \mathcal{E}_b$ satisfying $\gamma_a(x_a, u_a, x'_a) = \gamma_b(x_b, u_b, x'_b)$ and $(x'_a, x'_b) \in \mathcal{R}$.

We say that $\mathcal{S}_b$ simulates $\mathcal{S}_a$ if there is a simulation relation from $\mathcal{S}_a$ to $\mathcal{S}_b$, denoting it by $\mathcal{S}_a \preceq \mathcal{S}_b$. It is a simple exercise to see that $\mathcal{S}_a \preceq \mathcal{S}_b$ implies $\mathcal{V}^{\omega}(\mathcal{S}_a) \subseteq \mathcal{V}^{\omega}(\mathcal{S}_b)$, but the converse is not true.

Definition 3 (Alternating simulation relation): Consider two systems $\mathcal{S}_a$ and $\mathcal{S}_b$ with $\mathcal{U}_a \subseteq \mathcal{U}_b$ and $\gamma_a(\mathcal{E}_a) = \gamma_b(\mathcal{E}_b)$. A relation $\mathcal{R} \subseteq \mathcal{X}_a \times \mathcal{X}_b$ is a *alternating simulation relation* from $\mathcal{S}_a$ to $\mathcal{S}_b$ if the following conditions are satisfied:

i) for every $x_{b0} \in \mathcal{X}_{b0}$, there exists $x_{a0} \in \mathcal{X}_{a0}$ with $(x_{a0}, x_{b0}) \in \mathcal{R}$;
ii) for every $(x_a, x_b) \in \mathcal{R}$, it holds that $U_a(x_a) \subseteq U_b(x_b)$;
iii) for every $(x_a, x_b) \in \mathcal{R}, u \in U_a(x_a), x'_b \in \text{Post}_u(x_b)$, there exists $x'_a \in \text{Post}_u(x_a)$ such that $\gamma_a(x_a, u, x'_a) = \gamma_b(x_b, u, x'_b)$ and $(x'_a, x'_b) \in R$.

We say that $\mathcal{S}_b$ is an alternating simulation of $\mathcal{S}_a$, denoting it by $\mathcal{S}_a \preceq_{\text{AS}} \mathcal{S}_b$. This definition is useful for using strategies for $\mathcal{S}_a$ on $\mathcal{S}_b$: for any initial state in $\mathcal{X}_{b0}$, we can initialize the abstraction $\mathcal{S}_a$ with a related state. Then, any action u available at x_{a0} is also available in $\mathcal{S}_b$ thanks to condition (ii). Finally, whatever transition (x_b, u, x'_b) system $\mathcal{S}_b$ takes, we can pick a transition with same weight in $\mathcal{S}_a$ that again leads to related states (x'_a, x'_b), and both systems can continue progressing. The main difference in our definitions from those in [12], apart from the weighted aspect, is that we are concerned with an "output map" on transitions rather than on states; this is more convenient for our intended application, but not fundamentally different as one could convert a system with weights on transitions to one with weights on states. The following results are useful for our application:

Proposition 1: Consider two systems $\mathcal{S}_a$ and $\mathcal{S}_b$ and an alternating simulation relation $\mathcal{R} \subseteq \mathcal{X}_a \times \mathcal{X}_b$ from $\mathcal{S}_a$ to $\mathcal{S}_b$. If $\forall (x_a, x_b) \in \mathcal{R}, U_a(x_a) = U_b(x_b)$, then $\mathcal{R}^{-1}$ is a simulation relation from $\mathcal{S}_b$ to $\mathcal{S}_a$.

Proof: Condition (i) of Def. 2 is the flipped version of Def. 3 (i). Now assume Def. 2 (ii) is false for some $(x_b, x_a) \in \mathcal{R}^{-1}$; then there is (x_b, u_b, x'_b) without a matching transition in $\mathcal{S}_a$. However, by assumption, $u_b \in U_a(x_a)$; hence, from Def. 3 (iii), every $x'_b \in \text{Post}_{u_b}(x_b)$ has a matching $x'_a \in \text{Post}_{u_b}(x_a)$ s.t. $(x'_a, x'_b) \in \mathcal{R}$, which is a contradiction. □

Now, denote by $\mathcal{S}|s$ the system $\mathcal{S}$ controlled by strategy s; the following result is very similar to other results on alternating simulations (e.g., [12]):

Proposition 2: Let $\mathcal{S}_a$ and $\mathcal{S}_b$ be two non-blocking systems such that $\mathcal{S}_a \preceq_{\text{AS}} \mathcal{S}_b$. Let $\mathcal{V}^* \subseteq \mathcal{V}^{\omega}(\mathcal{S}_b)$ be a set of value traces. If there exists a strategy $s : X_b^+ \to \mathcal{U}$ such that

$\mathcal{V}^\omega(\mathcal{S}_b|s) \subseteq \mathcal{V}^*$, then there exists a strategy $s' : X_a^+ \to \mathcal{U}$ such that $\mathcal{V}^\omega(\mathcal{S}_a|s') \subseteq \mathcal{V}^*$.

Proof: (Sketch) Consider the alternating simulation relation $\mathcal{R} \subseteq (X_a, X_b)$. Take any state $x_b \in X_{b0}$: we can pick a related x_a (Def. 3 (i)). Choose $u = s(x_a)$; then $\forall (x_a, u, x_a') \in \mathcal{E}_a$, $\gamma_a(x_a, u, x_a')$ is the first element of some sequence in $\mathcal{V}^*$. From Def. 3 (ii), we can use u from x_b. From (iii), for any $(x_b, u, x_b') \in \mathcal{E}_b$ there exists $(x_a, u, x_a') \in \mathcal{E}_a$ with $(x_a', x_b') \in \mathcal{R}$, with matching weights. Since all such (x_a, u, x_a') have valid weights, so does any (x_b, u, x_b'). Hence, from any x_b' we can pick a related x_a' and choose $u = s(x_a x_a')$ for the next iteration. Repeating the same arguments recursively concludes the proof. □

The proof of Prop. 2 gives a way to use a strategy from an abstraction on the concrete system: run the abstraction in parallel with the concrete system: from any state x_b, pick the winning action u from a related state x_a and move the abstraction forward. If the strategy s is static, running the abstraction in parallel is unnecessary: the alternating simulation relation alone suffices, as long as every concrete state has a single related abstract state. This is the case for quotient systems [12], which we employ here. Hereafter we assume this is true; then, given a static strategy $s_a : \mathcal{X}_a \to \mathcal{U}_a \subseteq \mathcal{U}_b$, we call $s_b : \mathcal{X}_b \to \mathcal{U}_b$ a *refined strategy to* $\mathcal{S}_b$, or simply a *refinement of* s_a, by setting $s_b(x_b) = s_a(x_a)$, x_a being the unique state satisfying $(x_a, x_b) \in \mathcal{R}$.

B. Abstractions for optimal average weight

A transition system according to Def. 1 can be regarded as a game, where player 0 picks the action and player 1 antagonistically picks the transition. The problem of finding a strategy for a finite-state weighted transition system in order to maximize the average weight is known as mean-payoff game, a quantitative game. Quantitative games are games on weighted transition systems where a value function $\mathrm{Val} : \mathcal{V}^\omega(\mathcal{S}) \to \mathbb{R}$ is defined on runs, and one wants to find a strategy s for player 0 that

i) ensures a minimum value v, i.e, $\mathrm{Val}(w) \geq v$ for all $w \in \mathcal{V}^\omega(\mathcal{S}|s)$, or, if possible,
ii) maximizes the value, i.e., finds $\bar{v}$ such that $\mathrm{Val}(w) \geq \bar{v}$ for all $w \in \mathcal{V}^\omega(\mathcal{S}|s)$ and that, for any $v > \bar{v}$, there is no strategy s' s.t. $\mathrm{Val}(w') \geq v$ for all $w' \in \mathcal{V}^\omega(\mathcal{S}|s')$.

In the case of mean-payoff objectives, the value function is $\mathrm{Val}(v_0 v_1 ...) = \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} v_i$. There exist positional (i.e., static) optimal strategies for mean-payoff games [17],[1] which implies that every $x \in \mathcal{X}$ admits an *optimal value* $V(x)$ that is the smallest value obtained from every run starting at x when both players 0 and 1 play optimally. By considering that we cannot control the initial state, i.e., player 1 picks it, we have that the *game value* is $V(\mathcal{S}) := \bar{v} = \inf\{V(x) \mid x \in \mathcal{X}_0\}$. Once a strategy s has been decided for a system $\mathcal{S}$, $\mathcal{S}|s$ becomes a 1-player game, containing only the (antagonistic) environment. In the case of a 1-player game, where the environment chooses initial states and transitions, the value of the game can be taken as a function of its weight behaviors, i.e., $V_{\mathrm{adv}}(\mathcal{S}|s) = \inf\{\mathrm{Val}(w) \mid w \in \mathcal{V}^\omega(\mathcal{S}|s)\}$. We also define a cooperative value from a given initial condition $V^x_{\mathrm{coop}}(\mathcal{S}) := \sup\{\mathrm{Val}(w) \mid w \in \mathcal{V}^\omega_x(\mathcal{S})\}$, where player 0 has control of transitions and actions for any given initial state x; furthermore, let $V_{\mathrm{U}}(\mathcal{S}) := \inf_{x\in\mathcal{X}_0} V^x_{\mathrm{coop}}(\mathcal{S})$: essentially, player 1 picks the initial state, but player 0 can choose the run from it. The following result, very similar to what was done in [20] for time-optimal control, gives that a strategy from an abstraction, refined to the concrete system, ensures that at least the abstraction value is attained in the concrete case, while an upper bound can be obtained from V_{U}:

Proposition 3: Let $\mathcal{S}_a \preceq_{\mathrm{AS}} \mathcal{S}_b \preceq \mathcal{S}_a$. Then, $V(\mathcal{S}_b) \leq V_{\mathrm{U}}(\mathcal{S}_a)$. Moreover, let s_a be a strategy for $\mathcal{S}_a$ such that $V_{\mathrm{adv}}(\mathcal{S}_a|s_a) \geq v$. Then, if s_b is a refinement of s_a, it holds that $V_{\mathrm{adv}}(\mathcal{S}_b|s_b) \geq v$.

Proof: (Sketch) The first inequality comes from $V(\mathcal{S}_b) \leq V_{\mathrm{U}}(\mathcal{S}_b) \leq V_{\mathrm{U}}(\mathcal{S}_a)$; the first holds by definition of V_{U}, where player 1 is more powerful than in the original game; the second inequality holds from (weight) behavioral inclusion. For the last statement, it can be seen using similar arguments as [12, Sec. 8.2] that $\mathcal{S}_b|s_b \preceq \mathcal{S}_a|s_a$; hence, $\mathcal{V}^\omega(\mathcal{S}_b|s) \subseteq \mathcal{V}^\omega(\mathcal{S}_a|s)$, which implies that $V_{\mathrm{adv}}(\mathcal{S}_a|s_a) \geq v \implies V_{\mathrm{adv}}(\mathcal{S}_b|s_b) \geq v$. □

Prop. 3 implies that one can use an abstraction $\mathcal{S}_a$ to find a near-optimal strategy for the concrete system $\mathcal{S}_b$, then estimate its optimality gap by computing $\epsilon = V_{\mathrm{U}}(\mathcal{S}_a) - V_{\mathrm{adv}}(\mathcal{S}_a|s_a)$.[2]

C. SDSS design

Now that we have the relevant notions of abstraction in place, we can proceed to apply them towards an SDSS design for System (1). The first step is to describe the system as an infinite transition system; in this case, the "control action" is the (discrete) sampling time, where all sampling times from h until the deadline $d(\boldsymbol{x})$ (Eq. (7)) are allowed. The system is $\mathcal{S} := (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{E}, \mathcal{Y}, H, \gamma)$ where

$$\begin{aligned}
&\mathcal{X} = \mathcal{X}_0 = \mathbb{R}^{n_\times};\\
&\mathcal{U} = \mathcal{Y} = \{1, 2, ..., K\};\\
&\mathcal{E} = \{(\boldsymbol{x}, u, \boldsymbol{x}') \mid hu \leq d(\boldsymbol{x}) \text{ and } \boldsymbol{x}' = \boldsymbol{\xi}_{\boldsymbol{x}}(hu)\};\\
&H(\boldsymbol{x}) = d(\boldsymbol{x})/h;\\
&\gamma(\boldsymbol{x}, u, \boldsymbol{x}') = hu.
\end{aligned} \tag{8}$$

Note the peculiarities of this transition system: (i) the input set of a given state is determined by the state's output, which is its deadline; and, (ii) the weight is solely a function of the chosen input. These characteristics simplify the job of finding an abstraction that is alternatingly weight simulated by $\mathcal{S}$. We can construct a quotient system [12] as in [15], where the motivation was to allow early triggers for scheduling of multiple control loops in a single shared network. Better than that, we can start from the l-complete models [22] from [10],

[1] Solving the optimal value and strategy of a mean-payoff game has pseudopolynomial complexity, with the best known bound of $\mathcal{O}(|\mathcal{X}|^2|\mathcal{E}|W)$ due to [19], where W is the maximum weight. Weights are assumed to be integers for complexity analysis; since we assume weights to be in $\mathbb{Q}$, this can always be done by appropriate normalization.

[2] The 1-player-game values can be obtained for finite systems using Karp's algorithm [21], whose complexity, $\mathcal{O}(|\mathcal{X}||\mathcal{E}|)$, is much smaller than that for optimal mean-payoff games; for V_{U}, a combination of Karp's algorithm and reachability on graphs can be used, retaining the same complexity.

which can predict the next l deadlines if the PETC triggering strategy is used, and further augment the abstraction with early-triggering actions. Hence, let us recover the relation in [10] that allows the construction of the quotient state-space:

Definition 4 (Deadline sequence relation [10]): Given a sequence length l, we denote by $\mathcal{R}_l \subseteq \mathcal{X} \times \mathcal{Y}^l$ the relation satisfying $(\boldsymbol{x}, k_1 k_2 ... k_l) \in \mathcal{R}_l$ if and only if

$$\boldsymbol{x} \in \mathcal{Q}_{k_1}, \tag{9a}$$
$$\boldsymbol{M}(hk_1)\boldsymbol{x} \in \mathcal{Q}_{k_2}, \tag{9b}$$
$$\boldsymbol{M}(hk_2)\boldsymbol{M}(hk_1)\boldsymbol{x} \in \mathcal{Q}_{k_3}, \tag{9c}$$
$$\vdots$$
$$\boldsymbol{M}(hk_{l-1})...\boldsymbol{M}(hk_1)\boldsymbol{x} \in \mathcal{Q}_{k_l}, \tag{9d}$$

where

$$\mathcal{Q}_k \coloneqq \mathcal{K}_k \setminus \left(\bigcap_{j=1}^{k-1} \mathcal{K}_j \right) = \mathcal{K}_k \cap \bigcap_{j=1}^{k-1} \bar{\mathcal{K}}_j,$$
$$\mathcal{K}_k \coloneqq \begin{cases} \{\boldsymbol{x} \in \mathcal{X} \mid \boldsymbol{x}^\mathsf{T} \boldsymbol{N}(hk)\boldsymbol{x} > 0\}, & k < K, \\ \mathbb{R}^{n_\mathrm{x}}, & k = K, \end{cases} \tag{10}$$
$$\boldsymbol{N}(hk) \coloneqq \begin{bmatrix} \boldsymbol{M}(hk) \\ \mathbf{I} \end{bmatrix}^\mathsf{T} \boldsymbol{Q} \begin{bmatrix} \boldsymbol{M}(hk) \\ \mathbf{I} \end{bmatrix}.$$

Eq. (10) defines the sets $\mathcal{Q}_k$, containing the states whose deadline is hk. Eq. (9) simply asserts that a state $\boldsymbol{x} \in \mathbb{R}^n$ is related to a state $k_1 k_2 ... k_l$ of the abstraction if the IST sequence that it generates for the next l samples is $hk_1, hk_2, ..., hk_l$ when the PETC triggering rule is used, i.e., triggers occur at deadlines. Note that $\mathcal{Q}_k$ is a conjunction of quadratic inequalities; likewise, denoting $\sigma \coloneqq k_1 k_2 ... k_l$, we can define the set $\mathcal{Q}_\sigma \coloneqq \{\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}} \mid \boldsymbol{x} \text{ satisfies (9)}\}$, which is also given by a conjunction of quadratic inequalities. Then, a transition from some state in $\mathcal{Q}_\sigma$ to a state in $\mathcal{Q}_{\sigma'}$ exists if $\exists \boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}}$ such that

$$\boldsymbol{x} \in \mathcal{Q}_\sigma \text{ and } \boldsymbol{M}(hu)\boldsymbol{x} \in \mathcal{Q}_{\sigma'}, \tag{11}$$

for some u respecting the deadline, i.e., $u \leq k_1$. We can now define the following abstraction:

Definition 5: Given an integer $l \geq 1$, the *l-predictive traffic model* of $\mathcal{S}$ is the system $\mathcal{S}_l \coloneqq (\mathcal{X}_l, \mathcal{X}_l, \mathcal{U}, \mathcal{E}_l, \mathcal{Y}, H_l, \gamma_l)$, with

- $\mathcal{X}_l \coloneqq \pi_{\mathcal{R}_l}(\mathcal{X})$,
- $\mathcal{E}_l \coloneqq \{(\sigma, u, \sigma') \in \mathcal{X}_l \times \mathcal{U} \times \mathcal{X}_l \mid u \leq \sigma(1), \exists \boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}} :$ Eq. (11) holds$\}$,
- $H_l(k_1 k_2 ... k_l) \coloneqq k_1$.
- $\gamma_l(\sigma, u, \sigma') \coloneqq hu$.

The model above partitions $\mathbb{R}^{n_\mathrm{x}}$ into subsets associated with the next l deadlines times that the related states would generate under the PETC triggering rule. The transition set is determined by verifying Eq. (11), which is a reachability verification; both the state set and transition set can be determined by solving (non-convex) quadratic inequality satisfaction problems (Eqs. (9) and (11)) which can be done exactly using a satisfiability-modulo-theories (SMT) solver[3] such as Z3 [23], or approximately through convex relaxations as proposed in [15]. This system is alternatingly weight simulated by $\mathcal{S}$, as desired:

Proposition 4: The relation $\mathcal{R}$ from Def. 4 is a simulation relation from $\mathcal{S}$ (Eq. (8)) to $\mathcal{S}_l$ (Def. 1), and $\mathcal{R}_l^{-1}$ is a alternating simulation relation from $\mathcal{S}_l$ to $\mathcal{S}$.

Proof: (Sketch) The proof of alternating simulation is obtained by checking the conditions of Def 3: (i) is trivially satisfied, and so is (ii) with $U_l(k_1 k_2 ... k_l) = U(\boldsymbol{x}) = \{1, 2, ..., k_l\}$. Condition (iii) is ensured by construction of $\mathcal{E}_l$ and thanks to the fact that $\gamma(\boldsymbol{x}, u, \boldsymbol{x}') = \gamma_l(\sigma, u, \sigma') = hu$. The simulation then follows from Prop. 1. □

Obtaining the strategy and ϵ. Now that we have a method to abstract System (8) into a finite system, we can use the methods from Section III-B to build a near-optimal SDSS for the abstraction $\mathcal{S}_l$, then refine it for $\mathcal{S}$. The main question is how to define l. Given the results in [10], we suggest the following approach: (i) use [10] to compute the exact PETC SAIST, or a close enough under-approximation of it; denote this value by $V(\mathcal{S}_l | s_{\mathrm{PETC}})$. Set $l = 1$; Then, (ii) compute $\mathcal{S}_l$ (Def. 5) and solve the mean-payoff game for it, obtaining the strategy s_l and the game value estimate v_l. After that, with s_l' being a refinement of s_l to $\mathcal{S}$, (iii) compute $V_{\mathrm{U}}(\mathcal{S}_l)$ and verify, using a similar approach to [10], (a sufficiently close under-approximation of) $V_{\mathrm{adv}}(\mathcal{S}|s_l')$ [4] Finally, (iv) compute $\epsilon = V_{\mathrm{U}}(\mathcal{S}_l) - V(\mathcal{S}_l)$ and (v) if the improvement $V_{\mathrm{adv}}(\mathcal{S}|s_l') - V(\mathcal{S}_l|s_{\mathrm{PETC}})$ is large enough, ϵ is small enough, or l is too large,[5] stop; otherwise, increment l and redo steps (ii) to (v).

IV. Numerical example

Consider a plant and controller of the form (1) from [2]

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \ \boldsymbol{K} = \begin{bmatrix} 1 & -4 \end{bmatrix}$$

with the predictive Lyapunov-based triggering condition [24], [7] of the form $V(\boldsymbol{\zeta}(t)) > -\rho \boldsymbol{\zeta}(t)^\mathsf{T} \boldsymbol{Q}_{\mathrm{Lyap}} \boldsymbol{\zeta}(t)$, where $\boldsymbol{\zeta}(t) \coloneqq \boldsymbol{A}_{\mathrm{d}}(h)\boldsymbol{\xi}(t) + \boldsymbol{B}_{\mathrm{d}}(h)\boldsymbol{K}\hat{\boldsymbol{\xi}}(t)$ is the next-sample prediction of the state, $V(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T} \boldsymbol{P}_{\mathrm{Lyap}} \boldsymbol{x}$, and $0 < \rho < 1$ is the triggering parameter. The Lyapunov matrices were taken from [2] as $\boldsymbol{P}_{\mathrm{Lyap}} = \left[\begin{smallmatrix} 1 & 0.25 \\ 0.25 & 1 \end{smallmatrix}\right]$, $\boldsymbol{Q}_{\mathrm{Lyap}} = \left[\begin{smallmatrix} 0.5 & 0.25 \\ 0.25 & 1.5 \end{smallmatrix}\right]$, and we set $h = 0.1$ and $K = 20$, but the largest K that the system actually exhibits is equal to 11. A minimum-average-cycle-equivalent simulation [10] is found for the PETC, giving a SAIST of approx. 0.233 (obtained with $l = 8$). The strategy obtained with $l = 1$ already gives massive improvements: it increases SAIST to $V_{\mathrm{adv}}(\mathcal{S}|s_1') = 0.5$, which is more than twice the PETC's SAIST. Essentially, the obtained strategy simply limits $s(\boldsymbol{x})$ to 5 for any $k \geq 5$ satisfying $(\boldsymbol{x}, k) \in \mathcal{R}_1$. This surprisingly shows that limiting the maximum IST can actually have long-term benefits. With $l = 2$, the SAIST is improved further to 0.6; a simulation

[3] For that, the query is, e.g., $\exists \boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}}$: Eq. (9) holds.

[4] Even though [10] was proposed for the PETC strategy, the same approach can be used for any fixed sampling strategy, as its essential feature is verifying cycles in the concrete system.

[5] The complexity of the online part of the algorithm value is proportional to l, as the controller must predict the next l deadlines of the current state $\boldsymbol{x}$ under PETC. This involves simulating the PETC forward l steps, which takes at most lK operations of quadratic inequalities. Hence, a maximum l may be needed given online computational constraints.

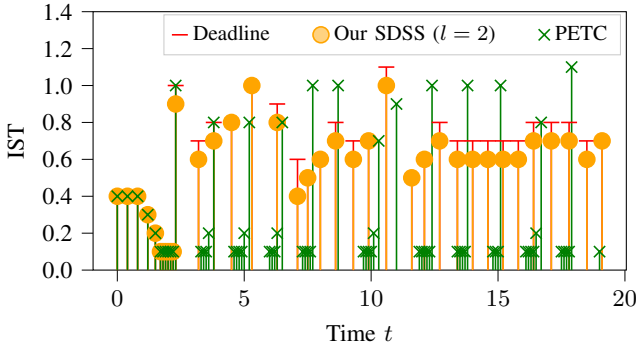

Fig. 1. Comparison between simulated traces of our SDSS ($l = 2$) and of the PETC, both with the same initial state.

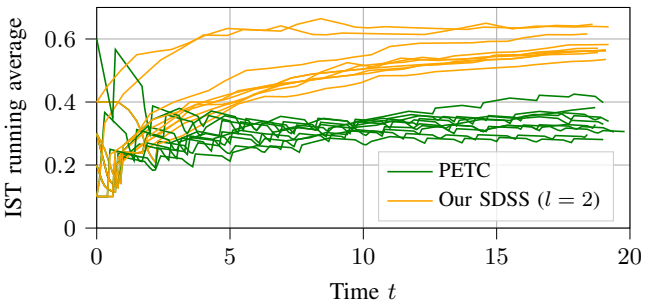

Fig. 2. Running average of the ISTs generated from 10 different initial conditions under PETC and our near-optimal SDSS using $l = 2$.

comparing this strategy and PETC is depicted in Fig. 1. One can see that only around $t = 2.5$ the SDSS samples before the deadline for the first time; doing so prevents the bursts of IST equal to 0.1 that happen recurrently with the PETC. The difference in (simulated) running averages between PETC and our SDSS is displayed in Fig. 2. The SAIST for $l = 3$ does not improve, and the upper bounds are $V_{\mathrm{U}}(\mathcal{S}_1) = V_{\mathrm{U}}(\mathcal{S}_2) = V_{\mathrm{U}}(\mathcal{S}_3) = \bar{\tau} = 1.1$.

V. DISCUSSION AND CONCLUSIONS

In this paper we have presented an abstraction-based approach to build aperiodic sampling strategies for LTI systems in order to maximize their average inter-sample time. For this we rely on the properties of PETC strategies, which ensure stability and performance of the closed loop whenever their "deadlines" are respected. This makes our abstraction inherently safe from a control perspective, but one could relax this condition by allowing "late" samplings as long as the abstraction can incorporate some information about the control performance. To that end, one could, e.g., incorporate Lyapunov-based costs to the abstractions.

As with most abstraction-based approaches, our method suffers from the curse of dimensionality. Even though we shift complexity to an offline phase, computing the abstractions $\mathcal{S}_l$ can easily be intractable as n_{x} gets large enough. Approximate methods to solve the satisfiability problems involved in building $\mathcal{S}_l$ (e.g., as in [15]) are subject of current investigation. Finally, considering external disturbances is an important next step; as dynamic triggering mechanisms [4] have been shown to perform better in this case, dynamic SDSSs could also be considered for that purpose.

REFERENCES

[1] M. Velasco, J. Fuertes, and P. Marti, "The self triggered task model for real-time control systems," in *Work-in-Progress Session of the 24th IEEE Real-Time Systems Symposium (RTSS03)*, vol. 384, 2003.

[2] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[3] X. Wang and M. D. Lemmon, "Event design in event-triggered feedback control systems," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 2105–2110, IEEE, 2008.

[4] A. Girard, "Dynamic triggering mechanisms for event-triggered control," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1992–1997, 2015.

[5] W. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pp. 3270–3285, IEEE, 2012.

[6] M. Mazo Jr., A. Anta, and P. Tabuada, "An ISS self-triggered implementation of linear controllers," *Automatica*, vol. 46, no. 8, pp. 1310–1314, 2010.

[7] A. Szymanek, G. A. Gleizer, and M. Mazo Jr., "Periodic event-triggered control with a relaxed triggering condition," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 1656–1661, IEEE, 2019.

[8] D. Antunes, W. Heemels, and P. Tabuada, "Dynamic programming formulation of periodic event-triggered control: Performance guarantees and co-design," in *2012 IEEE 51st IEEE conference on decision and control (CDC)*, pp. 7212–7217, IEEE, 2012.

[9] Y. Xu and J. P. Hespanha, "Optimal communication logics in networked control systems," in *2004 43rd IEEE Conference on Decision and Control (CDC)*, vol. 4, pp. 3527–3532, IEEE, 2004.

[10] G. de A. Gleizer and M. Mazo Jr., "Computing the sampling performance of event-triggered control," in *Proc. of the 24th Int'l Conf. on Hybrid Systems: Computation and Control*, HSCC '21, ACM, 2021.

[11] W. P. M. H. Heemels, M. C. F. Donkers, and A. R. Teel, "Periodic event-triggered control for linear systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 847–861, 2013.

[12] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[13] K. Chatterjee, L. Doyen, and T. A. Henzinger, "Quantitative languages," *ACM Transactions on Computational Logic (TOCL)*, vol. 11, no. 4, pp. 1–38, 2010.

[14] A. S. Kolarijani, D. Adzkiya, and M. Mazo Jr., "Symbolic abstractions for the scheduling of event-triggered control systems," in *IEEE 54th Annual Conference on Decision and Control (CDC)*, pp. 6153–6158, IEEE, 2015.

[15] G. de A. Gleizer and M. Mazo Jr., "Scalable traffic models for scheduling of linear periodic event-triggered controllers," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2726–2732, 2020.

[16] G. Delimpaltadakis and M. Mazo Jr., "Traffic abstractions of nonlinear event-triggered control systems with disturbances and uncertainties," 2020.

[17] A. Ehrenfeucht and J. Mycielski, "Positional strategies for mean payoff games," *International Journal of Game Theory*, vol. 8, no. 2, pp. 109–113, 1979.

[18] C. Fiter, L. Hetel, W. Perruquetti, and J.-P. Richard, "A state dependent sampling for linear state feedback," *Automatica*, vol. 48, no. 8, pp. 1860–1867, 2012.

[19] C. Comin and R. Rizzi, "Improved pseudo-polynomial bound for the value problem and optimal strategy synthesis in mean payoff games," *Algorithmica*, vol. 77, no. 4, pp. 995–1021, 2017.

[20] M. Mazo Jr. and P. Tabuada, "Symbolic approximate time-optimal control," *Systems & Control Letters*, vol. 60, no. 4, pp. 256–263, 2011.

[21] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete mathematics*, vol. 23, no. 3, pp. 309–311, 1978.

[22] T. Moor and J. Raisch, "Supervisory control of hybrid systems within a behavioural framework," *Systems & control letters*, vol. 38, no. 3, pp. 157–166, 1999.

[23] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, Springer, 2008.

[24] G. de A. Gleizer and M. Mazo Jr., "Towards traffic bisimulation of linear periodic event-triggered controllers," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 25–30, 2021.