Scalable Traffic Models for Scheduling of Linear Periodic Event-Triggered Controllers

de Albuquerque Gleizer, G.; Mazo, M.

DOI
10.1016/j.ifacol.2020.12.2525

Publication date
2020

Document Version
Final published version

Published in
IFAC-PapersOnLine

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 2726–2732

Scalable Traffic Models for Scheduling of Linear Periodic Event-Triggered Controllers

Gabriel de A. Gleizer * **Manuel Mazo Jr.** *

* *Delft Center for Systems and Control, TU Delft, The Netherlands (e-mail: {g.gleizer, m.mazo}@tudelft.nl)*

Abstract:
This paper addresses the problem of modeling and scheduling the transmissions generated by multiple event-triggered control (ETC) loops sharing a network. We present a method to build a finite-state similar model of the traffic generated by periodic ETC (PETC), which by construction mitigates the combinatorial explosion that is typical of symbolic models. The model is augmented with early triggering actions that can be used by a scheduler. The complete networked control system is then modeled as a network of timed game automata, for which existing tools can generate strategies that avoids communication conflicts, while keeping early triggers to a minimum. Our proposed model is relatively fast to build and is the first to constitute an exact simulation. Finally, we demonstrate modeling and scheduling for a numerical example.

Keywords: Control systems, digital control, linear systems, event-triggered control, networked control systems, formal methods, scheduling.

1. INTRODUCTION

Networks have become prevalent as the communication media for control devices. Despite the cost and implementability benefits brought by such Networked Control Systems (NCSs), the lack of dedicated communication lines has introduced a challenge for practitioners: managing the transmissions generated by each control loop without compromising control performance itself. In this context, aperiodic sampling methods such as Event-Triggered Control (ETC, Tabuada, 2007) and Self-Triggered Control (Anta and Tabuada, 2008, STC,) have been proposed. These methods significantly decrease network usage when compared to standard periodic sampling. ETC communications are triggered by state-dependent events, while STC communication times are determined by the controller after every new data acquisition, generally by predicting when an ETC would trigger. [1] Since then, many studies have focused on designing sampling strategies to reduce communication even further (see, e.g., Wang and Lemmon, 2008; Girard, 2015; Dolk et al., 2017), among which there is periodic event-triggered control (PETC, Heemels et al., 2013), which provides more practical implementations. Other researchers have proposed co-designing the controller and triggering mechanism to achieve the desired control performance (e.g., Peng and Yang, 2013; Donkers et al., 2014). We do not consider co-design in this work in order to separate the concerns of control design from those of its digital implementation.

Despite the communication savings achieved by ETC and STC, little research has addressed the coordination of data transfers from multiple controllers in a single network; scheduling is particularly difficult for ETC, since its triggering times vary immensely. Few exceptions are Kolarijani and Mazo Jr (2016); Mazo Jr et al. (2018); Fu and Mazo Jr. (2018), who propose conflict-avoiding scheduler design by means of symbolic abstractions of the ETC traffic. Using timed game automata (TGA) for approximately simulating ETC traffic, they demonstrate that a scheduling strategy can be computed by composing multiple traffic TGAs with a network TGA and solving a safety game. The major drawback of the abstractions presented in Kolarijani and Mazo Jr (2016) is the curse of dimensionality: their proposed isotropic partitioning creates a model with the number of locations that depend exponentially on the state-space dimension of the plant. For PETC, a traffic model was also proposed in Fu and Mazo Jr. (2018), but it also suffers from the same dimensionality issue due to the use of isotropic partitioning.

In this paper, we follow the same philosophy of Mazo Jr et al. (2018) for scheduling, but propose a different way of creating the traffic models: instead of partitioning space, we partition time, and determine the states associated with a given triggering time *a posteriori*. For PETC this allows to construct a quotient model (Tabuada, 2009), which provides an exact simulation relation with the actual traffic generated. The resulting regions are intersections of quadratic non-convex cones that, despite being easy to check membership online, make the problem of computing transitions a non-convex quadratic constrain satisfaction problem, which is in general NP-hard (Park and Boyd, 2017). We propose using semidefinite relaxations (Boyd and Vandenberghe, 2004; Park and Boyd, 2017), which are fast and reliable, but add extra conservativeness to the resulting abstraction. After having constructed the traffic model, we augment it to allow for controllable early triggers, which can be used by the scheduler to avoid conflicts. Finally, we follow the steps in Mazo Jr et al. (2018) to compose the scheduling problem, with some

[1] For an introduction on ETC and STC, see Heemels et al. (2012).

10.1016/j.ifacol.2020.12.2525

minor modifications to keep the number and earliness of scheduling interventions small. For testing it, we generate strategies using UPPAAL Tiga (Behrmann et al., 2007) and provide simulation results for an NCS with two ETC loops. This demonstrates the usage of our method, which can support implementation of PETC in real NCSs, while helping realize the full potential of event-triggered control.

1.1 Notation

We denote $\mathbb{N}_0$ the set of natural numbers including zero, $\mathbb{N} \coloneqq \mathbb{N}_0 \setminus \{0\}$, and $\mathbb{R}_+$ the set of non-negative reals. For a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, we write $\mathrm{Tr}(\boldsymbol{A})$ to denote its trace, and $\boldsymbol{A} \succ \boldsymbol{0}$ ($\boldsymbol{A} \succeq \boldsymbol{0}$) if $\boldsymbol{A}$ is positive definite (semi-definite). The sets $\mathbb{S}, \mathbb{S}_+$ and $\mathbb{S}_{++}$ are the sets of symmetric, positive definite, and positive semi-definite matrices, respectively. For a set $\mathcal{X}$, we denote by $\bar{\mathcal{X}}$ its complement; when $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{X}$ is an equivalence relation on $\mathcal{X}$, we denote by $\mathcal{X}/\mathcal{R}$ the set of all equivalent classes.

2. PRELIMINARIES

2.1 Transition systems

For formally establish a relation between (finite and infinite) systems, we use the framework of Tabuada (2009):

Definition 1. (Transition System (Tabuada, 2009)). A system $\mathcal{S}$ is a tuple $(\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{E}, \mathcal{Y}, H)$ 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 inputs,
- $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{U} \times \mathcal{X}$ 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.

A system is called finite (infinite) state if the cardinality of $\mathcal{X}$ is finite (infinite). A system is called autonomous if $\mathcal{U} = \emptyset$, in which case a transition is denoted by a pair $(x, x') \in \mathcal{X} \times \mathcal{X}$ instead of a triplet.

We aim at constructing an Automaton model of the timing of an ETC by using the notion of simulation relation:

Definition 2. (Simulation Relation (Tabuada, 2009)). Consider two systems $\mathcal{S}_a$ and $\mathcal{S}_b$ with $\mathcal{Y}_a = \mathcal{Y}_b$. 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:

- for every $x_{a0} \in \mathcal{X}_{a0}$, there exists $x_{b0} \in \mathcal{X}_{b0}$ with $(x_{a0}, x_{b0}) \in \mathcal{R}$;
- for every $(x_a, x_b) \in \mathcal{R}, H_a(x_a) = H_b(x_b)$;
- 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 $(x'_a, x'_b) \in \mathcal{R}$.

A simulation relation from $\mathcal{S}_a$ to $\mathcal{S}_b$ is denoted by $\mathcal{S}_a \subseteq \mathcal{S}_b$. Essentially, a simulation relation $\mathcal{R} \subseteq \mathcal{X}_a \times \mathcal{X}_b$ captures which states of $\mathcal{S}_a$ are simulated by which states of $\mathcal{S}_b$: for the right state selection, their outputs are the same; and every transition in $\mathcal{S}_a$ leads to a state whose output can also be attained in $\mathcal{S}_b$ after a single transition. It is important to notice, however, that there might be transitions in $\mathcal{S}_b$ that lead to states that are not related to the ones attained in $\mathcal{S}_a$. When using simulation relations to model the behavior of a system, these transitions are called spurious transitions.

Finally, we introduce the notion of quotient system:

Definition 3. (Quotient System (Tabuada, 2009)). Consider a system $\mathcal{S} = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{E}, \mathcal{Y}, H)$ and let $\mathcal{R}$ be an equivalence relation on $\mathcal{X}$ such that $(x, x') \in \mathcal{R} \implies H(x) = H(x')$. The quotient of $\mathcal{S}$ by $\mathcal{R}$, denoted by $\mathcal{S}_{/\mathcal{R}}$, is the system $(\mathcal{X}_{/\mathcal{R}}, \mathcal{X}_{/\mathcal{R}0}, \mathcal{U}, \mathcal{E}_{/\mathcal{R}}, \mathcal{Y}, H_{/\mathcal{R}})$ consisting of

- $\mathcal{X}_{/\mathcal{R}} = \mathcal{X}/\mathcal{R}$;
- $\mathcal{X}_{/\mathcal{R}0} = \{x_{/\mathcal{R}} \in \mathcal{X}_{/\mathcal{R}} : x_{/\mathcal{R}} \cap \mathcal{X}_0 \neq \emptyset\}$;
- $(x_{/\mathcal{R}}, u, x'_{/\mathcal{R}}) \in \mathcal{E}_{/\mathcal{R}}$ if there exists $(x, u, x') \in \mathcal{E}$ with $x \in x_{/\mathcal{R}}$ and $x' \in x'_{/\mathcal{R}}$;
- $H_{/\mathcal{R}}(x_{/\mathcal{R}}) = H(x)$ for some $x \in x_{/\mathcal{R}}$.

Building a quotient system is fundamentally aggregating states of the original system that produce the same output, and then determining the transitions so that every possible transition of the original system is reproduced in the quotient (symbolic) system. By construction, $\mathcal{S} \subseteq \mathcal{S}_{/\mathcal{R}}$.

2.2 Timed automata

Timed Automata are regular Automata that make use of clocks, which are resettable real-valued variables measuring the passage of time. Let $\mathcal{C}$ be a finite set of said clocks, and consider $\bowtie \in \{<, \leq, =, \geq, >\}$. A clock constraint g is a conjunctive formula of atomic constraints $c \bowtie k, c \in \mathcal{C}, k \in \mathbb{N}$. We denote by $\mathcal{B}(\mathcal{C})$ the set of all clock constraints.

Definition 4. (Timed Safety Automaton, (Bengtsson and Yi, 2004)). A Timed Safety Automaton is a tuple $\mathcal{A} = (\mathcal{L}, \mathcal{L}_0, \mathcal{U}, \mathcal{C}, \mathcal{E}, I)$ where:

- $\mathcal{L}$ is the finite set of locations (or discrete states),
- $\mathcal{L}_0 \subseteq \mathcal{L}$ is the set of initial locations,
- $\mathcal{U}$ is the finite set of actions,
- $\mathcal{C}$ is the finite set of clocks,
- $\mathcal{E} \subseteq \mathcal{L} \times \mathcal{B}(\mathcal{C}) \times \mathcal{U} \times 2^{\mathcal{C}} \times \mathcal{L}$ is the set of edges (or transitions), and
- $I : \mathcal{L} \to \mathcal{B}(\mathcal{C})$ assigns invariants to locations.

A TSA is a system with both discrete (the locations) and continuous states (the clocks). All clocks increase value at the same rate, but transitions can reset the value of certain clocks. The system can change locations through edges, depending on the action taken and the clocks' values. We denote by $l \xrightarrow{g,a,r} l'$ the transition from $l \in \mathcal{L}$ to $l' \in \mathcal{L}$ under action $a \in \mathcal{U}$, with $r \subseteq \mathcal{C}$ as the set of clocks reset when this transition is taken, and g over $\mathcal{C}$ as the guards that enabled the transition. *Invariants* of a location are the sufficient clock conditions for a transition to happen; in other words, the system is forced to leave the place l if a clock c violates any invariant $I(l)$. Symmetrically, a *guard* is a necessary condition for a transition to occur.

TGA extend TSA by partitioning the set of actions into controllable and uncontrollable. Controllable actions are decisions that the system operator can choose, while uncontrollable actions are taken independently of the system operator (e.g., by the environment or an opponent).

Definition 5. (Timed Game Automaton, (Bengtsson and Yi, 2004)). A Timed Game Automaton is a tuple $\mathcal{A} = (\mathcal{L}, \mathcal{L}_0, \mathcal{U}_\mathrm{c}, \mathcal{U}_\mathrm{u}, \mathcal{C}, \mathcal{E}, I)$ where:

- $(\mathcal{L}, \mathcal{L}_0, \mathcal{U}_\mathrm{c} \cup \mathcal{U}_\mathrm{u}, \mathcal{C}, \mathcal{E}, I)$ is a TSA,
- $\mathcal{U}_\mathrm{c}$ is the set of controllable actions,
- $\mathcal{U}_\mathrm{u}$ is the set of uncontrollable actions, and
- $\mathcal{U}_\mathrm{c} \cap \mathcal{U}_\mathrm{u} = \emptyset$.

The distinction between controllable and uncontrollable is paramount in our case. The scheduler can control when to sample, but not how the system will react to this choice.

To define a strategy, let $\mathcal{A}$ be a TGA, and $\mathcal{L}_\mathrm{c} \subseteq \mathcal{L}$ be its set of locations, for which a controllable action exists. A strategy $S : \mathcal{L}_\mathrm{c} \times \mathcal{C} \to 2^{\mathcal{U}_\mathrm{c}}$ determines which actions can be taken depending on the TGA states. A deterministic strategy outputs a single action.

Finally, TGAs can be combined into a *network of timed game automata* (NTGA), which allows for modularity (Bengtsson and Yi, 2004). An NTGA consists of n TGAs $\mathcal{A}_i = (\mathcal{L}_i, \mathcal{L}_{i0}, \mathcal{U}_\mathrm{c}, \mathcal{U}_\mathrm{u}, \mathcal{C}, \mathcal{E}_i, I_i)$, where 1) uncontrollable actions take precedence over controllable actions, and 2) a location of the network, denoted as $\bar{l} := (l_1, ..., l_n)$, has its invariant $I(\bar{l}) = \wedge_i I_i(l_i)$. Most importantly, TGAs within an NTGA can have transitions influence each other through *synchronization channels*: for a channel `a`, the initiating transition is labeled `a!` and, when fired, all transitions labeled `a?` have to fire simultaneously.

2.3 Periodic event-triggered control

Consider the plant with a sample-and-hold state-feedback control below:

$$\begin{aligned} \dot{\boldsymbol{\xi}}(t) &= \boldsymbol{A}\boldsymbol{\xi}(t) + \boldsymbol{B}\boldsymbol{K}\hat{\boldsymbol{\xi}}(t), \\ \boldsymbol{\xi}(0) &= \hat{\boldsymbol{\xi}}(0) = \boldsymbol{\xi}_0, \end{aligned} \tag{1}$$

where $\boldsymbol{\xi}(t) \in \mathbb{R}^{n_\mathrm{x}}$ is the state with initial value $\boldsymbol{\xi}_0$, $\hat{\boldsymbol{\xi}}(t) \in \mathbb{R}^{n_\mathrm{x}}$ is the available measurement of the state, $\boldsymbol{K}\hat{\boldsymbol{\xi}}(t) \in \mathbb{R}^{n_\mathrm{u}}$ is the control input, n_x and n_u are the state- and input-space dimensions, respectively, and $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{K}$ are matrices of appropriate dimensions. The controller is of zero-order hold type; i.e., consider a sequence of sampling times $t_i \in \mathbb{R}_+$, 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 event-triggered control, the sequence of times t_i is generated by a *triggering condition*, which is generally a function of the states of the system. In periodic ETC, such a condition is checked periodically, with a fundamental checking period h:

$$t_{i+1} = \inf \left\{ t = kh > t_i, k \in \mathbb{N} \,\middle|\, \begin{bmatrix} \boldsymbol{\xi}(t) \\ \boldsymbol{x} \end{bmatrix}^\mathsf{T} \boldsymbol{Q} \begin{bmatrix} \boldsymbol{\xi}(t) \\ \boldsymbol{x} \end{bmatrix} > 0 \;\vee\; t - t_i \leq \bar{k}h \right\}, \tag{2}$$

where $\boldsymbol{x} = \boldsymbol{\xi}(t_i)$, $\boldsymbol{Q} \in \mathbb{S}^{2n_\mathrm{x}}$ is the designed triggering matrix, and $\bar{k}$ is a chosen maximum inter-event time. Many of the triggering conditions available in the literature can be written as in Eq. (2). We kindly refer the interested reader to Heemels et al. (2013) for the list of conditions and their formulations.

In-between t_i and t_{i+1}, the value of $\boldsymbol{\xi}(kh)$ is

$$\boldsymbol{\xi}_{\boldsymbol{x}}(kh) = \boldsymbol{M}(k)\boldsymbol{x}, \;\; \boldsymbol{M}(k) := \mathrm{e}^{\boldsymbol{A}kh} + \int_0^{kh} \mathrm{e}^{\boldsymbol{A}\tau} \mathrm{d}\tau \boldsymbol{B}\boldsymbol{K}, \tag{3}$$

where $\boldsymbol{\xi}_{\boldsymbol{x}}(t)$ is used to denote the value of $\boldsymbol{\xi}$ at t when $\boldsymbol{\xi}(0) = \hat{\boldsymbol{\xi}}(t) = \boldsymbol{x}$. One can determine the discrete inter-event $\kappa := (t_{i+1} - t_i)/h$ time as a function of the currently held state by combining Equations (2) and (3):

$$\begin{aligned} \kappa(\boldsymbol{x}) &= \min \left\{ k \in \{1, 2, ...\bar{k}\} \middle| \boldsymbol{x}^\mathsf{T} \boldsymbol{N}(k) \boldsymbol{x} > 0 \vee k = \bar{k} \right\} \\ \boldsymbol{N}(k) &:= \begin{bmatrix} \boldsymbol{M}(k) \\ \mathbf{I} \end{bmatrix}^\mathsf{T} \boldsymbol{Q} \begin{bmatrix} \boldsymbol{M}(k) \\ \mathbf{I} \end{bmatrix}, \end{aligned} \tag{4}$$

where $\mathbf{I}$ denotes the identity matrix.

3. PROBLEM FORMULATION

The starting point for scheduling ETC traffic is modeling it, for which we use symbolic abstractions as in Kolarijani and Mazo Jr (2015); Mazo Jr et al. (2018); however, we aim to build a quotient model, obtaining an exact simulation relation. More than that, we want to mitigate the curse of dimensionality that is typical of such abstractions:

Problem 6. Build a quotient model $\mathcal{S}_{/\mathcal{R}}$ for the traffic generated by system (1) using triggering condition (2) such that the cardinality of $\mathcal{X}_{/\mathcal{R}}$ does not directly depend on n_x.

A traffic model alone is not sufficient for scheduling. System (1) is autonomous, and a scheduler needs to be able to alter the traffic pattern in some way to avoid communication conflicts. We choose to allow the scheduler to request data before the ETC triggers. Thus, we need to enrich the traffic model with controllable actions that represent this early triggering:

Problem 7. Enhance $\mathcal{S}_{/\mathcal{R}}$ with transitions that capture the evolution of system (1) when inter-event times smaller than $\kappa(x)$ are chosen.

Finally, we need to pose the scheduling problem:

Problem 8. Design an NTGA that forms the scheduling problem, for which a strategy serves as a scheduler for the NCS with multiple event-triggered loops. In doing so, try to keep the number of communications to a small level.

4. PETC TRAFFIC MODEL

Constructing a similar model of the traffic generated by (1)–(2) requires two steps: 1) gathering the states that share the same output in a single quotient state, and 2) computing the transition relations between them. First, we must define the actual, infinite-state, traffic model: it is the system $\mathcal{S} = (\mathcal{X}, \mathcal{X}_0, \emptyset, \mathcal{E}, \mathcal{Y}, H)$ where

$$\begin{aligned} &\mathcal{X} = \mathcal{X}_0 = \mathbb{R}^{n_\mathrm{x}}; \\ &\mathcal{E} = \{(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{X} \times \mathcal{X} | \boldsymbol{x}' = \boldsymbol{\xi}_{\boldsymbol{x}}(h\kappa(\boldsymbol{x}))\}; \\ &\mathcal{Y} = \{1, 2, ..., \bar{k}\}; \\ &H = \kappa. \end{aligned} \tag{5}$$

4.1 Quotient state set

Gathering states that share the same output is in a sense straightforward in PETC. From Eq. (4), we can determine the set $\mathcal{K}_k \subseteq \mathbb{R}^{n_\mathrm{x}}$ of states that will certainly have triggered by time k:

$$\mathcal{K}_k = \begin{cases} \{\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}} | \boldsymbol{x}^\mathsf{T} \boldsymbol{N}(k) \boldsymbol{x} > 0\}, & k < \bar{k}, \\ \mathbb{R}^{n_\mathrm{x}}, & k = \bar{k}. \end{cases} \tag{6}$$

To determine the state set whose output k is the *minimum* that satisfies $\boldsymbol{x}^\mathsf{T}\boldsymbol{N}(k)\boldsymbol{x} > 0$, one must remove from $\mathcal{K}_k$ all states that could have triggered before, i.e., that belong to some $\mathcal{K}_j$ with $j < k$. This is expressed as

$$\mathcal{Q}_k = \mathcal{K}_k \cap \bigcap_{j=1}^{k-1} \bar{\mathcal{K}}_j. \tag{7}$$

By construction, $\mathcal{Q}_k, k \in \{1, 2, ..., \bar{k}\}$ constitutes a partition of $\mathbb{R}^{n_\mathrm{x}}$; also, $H(\boldsymbol{x}) = k, \forall \boldsymbol{x} \in \mathcal{Q}_k$. Therefore, $\mathcal{X}_{/\mathcal{R}} = \{\mathcal{Q}_1, \mathcal{Q}_2, ...\}$ is a good candidate for a quotient state set of the system $\mathcal{S}$. Finally, different from Kolarijani and Mazo Jr (2016), we have that $|\mathcal{X}_{/\mathcal{R}}| = \bar{k}$, i.e., the cardinality of the quotient state space does not depend explicitly on n_x. This in part accomplishes solving Problem 6; however, for completing the model, we need to establish the transitions between these quotient states.

Remark 9. Matrices $\boldsymbol{N}(k)$ can be computed offline. Online determination of which region the current state $\boldsymbol{x}$ belongs to requires at most $\bar{k}$ quadratic operations.

Remark 10. Unperturbed state-feedback ETC has an intrinsic positive minimum inter-event time (MIET), which, in the case of PETC, can be bigger than $k = 1$. In this case, for all $k < \underline{k}$, where $\underline{k}$ is such MIET, all $\boldsymbol{N}(k) \preceq \boldsymbol{0}$. This can be checked offline, and the corresponding matrices may be discarded. Likewise, a maximum inter-event time $\bar{k}$ can naturally show up if, for some k^*, $\boldsymbol{N}(k^*) \succ \boldsymbol{0}$, which can also be checked offline. In this case, take $\bar{k} = k^*$.

4.2 Quotient transition relations

The problem of determining the transition relation between two quotient states $\mathcal{Q}_i$ and $\mathcal{Q}_j$ is, from Eq. (5),

$$\exists \boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}} : \boldsymbol{x} \in \mathcal{Q}_i, \boldsymbol{\xi}_{\boldsymbol{x}}(ih) = \boldsymbol{M}(i)\boldsymbol{x} \in \mathcal{Q}_j, \tag{8}$$

where the last equality uses Eq. (3). Expanding $\mathcal{Q}_i, \mathcal{Q}_j$ with Eqs. (7) and (6) arrives in the following *non-convex quadratic constraint satisfaction problem*:

$$\begin{aligned} \exists\ & \boldsymbol{x} \in \mathbb{R}^{n_\mathrm{x}} \\ \text{s.t.}\ & \boldsymbol{x}^\mathsf{T}\boldsymbol{N}(i)\boldsymbol{x} > 0, \\ & \boldsymbol{x}^\mathsf{T}\boldsymbol{N}(i')\boldsymbol{x} \leq 0, \forall i' \in \{1, ..., i-1\}, \\ & \boldsymbol{x}^\mathsf{T}\boldsymbol{M}(i)^\mathsf{T}\boldsymbol{N}(j)\boldsymbol{M}(i)\boldsymbol{x} > 0, \\ & \boldsymbol{x}^\mathsf{T}\boldsymbol{M}(i)^\mathsf{T}\boldsymbol{N}(j')\boldsymbol{M}(i)\boldsymbol{x} \leq 0, \forall j' \in \{1, ..., j-1\}. \end{aligned} \tag{9}$$

The non-convexity of this problem can be easily checked using the facts that both $>$ and $\leq$ inequalities are present, and that the matrices $\boldsymbol{N}(i)$ are non-definite.[2] We solve it by means of semi-definite relaxations (SDR, Boyd and Vandenberghe, 2004),[3] which take the form

$$\begin{aligned} \exists\ & \boldsymbol{X} \in \mathbb{S}_+^{n_\mathrm{x}} \\ \text{s.t.}\ & \mathrm{Tr}(\boldsymbol{X}^\mathsf{T}\boldsymbol{N}(i)) \geq 0, \\ & \mathrm{Tr}(\boldsymbol{X}\boldsymbol{N}(i')) \leq 0, \forall i' \in \{1, ..., i-1\}, \\ & \mathrm{Tr}(\boldsymbol{X}\boldsymbol{M}(i)^\mathsf{T}\boldsymbol{N}(j)\boldsymbol{M}(i)) \geq 0, \\ & \mathrm{Tr}(\boldsymbol{X}\boldsymbol{M}(i)^\mathsf{T}\boldsymbol{N}(j')\boldsymbol{M}(i)) \leq 0, \forall j' \in \{1, ..., j-1\}, \\ & \mathrm{Tr}(\boldsymbol{X}) = 1, \end{aligned} \tag{10}$$

where the last equation was added to avoid the trivial solution $\boldsymbol{X} = 0$; the value 1 was chosen arbitrarily, since Eq. (9) is homogeneous. To determine (offline) the complete transition set $\mathcal{E}_{/\mathcal{R}}$, one requires solving $\bar{k}^2$ semidefinite problems. The final model follows:

Model 11. (PETC Traffic Model). The model is the system $\mathcal{S}_{/\mathcal{R}} = (\mathcal{X}_{/\mathcal{R}}, \mathcal{X}_{/\mathcal{R}0}, \emptyset, \mathcal{E}_{/\mathcal{R}}, \mathcal{Y}, H_{/\mathcal{R}})$ with

- $\mathcal{X}_{/\mathcal{R}} = \mathcal{X}_{/\mathcal{R}0} = \{\mathcal{Q}_1, \mathcal{Q}_2, ..., \mathcal{Q}_{\bar{k}}\}$;
- $\mathcal{E}_{/\mathcal{R}} = \{(\mathcal{Q}_i, \mathcal{Q}_j) | \text{Eq. (10) is satisfied}\}$;
- $H_{/\mathcal{R}}(\mathcal{Q}_k) = k$.

By construction, we obtain the following result:

Theorem 12. Model 11 is a quotient system of $\mathcal{S}$ from Eq. (5), and, therefore, $\mathcal{S}_{/\mathcal{R}}$ simulates $\mathcal{S}$.

In other words, all sequences of triggering times generated by system (1)–(4) can be generated by our model $\mathcal{S}_{/\mathcal{R}}$. This solves Problem 6.

Remark 13. A relaxation generally provides conservative solutions. In our case, it may generate spurious transitions. If such transitions do occur, this does not change the fact that the constructed symbolic model simulates $\mathcal{S}$.

5. SCHEDULING OF PETC SYSTEMS

5.1 Early triggering and TGA

As stated earlier, for the traffic model to be applicable for scheduling, we need to augment it with controllable transitions that correspond to early triggering. From a quotient state $\mathcal{Q}_i$, one can allow early triggers for any $k \in \mathbb{N} : k < i$; for simplicity we choose to label the corresponding actions by k. It remains necessary to verify which transitions exist for such actions. Obviously, this can be done by solving the SDR problem (10) as before, replacing j by k. We denote the set of early triggering transitions by $\mathcal{E}^*$ and the resulting system as $\mathcal{S}^*_{/\mathcal{R}}$. Computing all of its transitions requires solving $\bar{k} + 2\bar{k} + ... + \bar{k}(\bar{k}-1) = \bar{k}^2(\bar{k}-1)/2$ semidefinite problems.

Finally, we transform the quotient system into a TGA. For the game part, we set the early triggering actions in $\mathcal{S}^*_{/\mathcal{R}}$ as controllable, and the event triggers as uncontrollable. All that is left is defining the clock set, the guards, and the invariants, resulting in the following TGA:

Model 14. (PETC Traffic Timed Game). The model is the TGA $\mathcal{A} = (\mathcal{X}_{/\mathcal{R}}, \mathcal{X}_{/\mathcal{R}0}, \mathcal{U}_\mathrm{c}, \mathcal{U}_\mathrm{u}, \mathcal{C}, \mathcal{E}_\mathrm{c} \cup \mathcal{E}_\mathrm{u}, I)$ where

- $\mathcal{U}_\mathrm{c} = \{\texttt{early}\}$;
- $\mathcal{U}_\mathrm{u} = \{\texttt{trigger}\}$;
- $\mathcal{C} = \{c\}$;
- $\mathcal{E}_\mathrm{c} = \{(\mathcal{Q}_i, c = k, \texttt{early}, \{c\}, \mathcal{Q}_j) : (\mathcal{Q}_i, k, \mathcal{Q}_j) \in \mathcal{E}^*\}$;
- $\mathcal{E}_\mathrm{u} = \{(\mathcal{Q}_i, c = i, \texttt{trigger}, \{c\}, \mathcal{Q}_j) : (\mathcal{Q}_i, \mathcal{Q}_j) \in \mathcal{E}_{/\mathcal{R}}\}$;
- $I(\mathcal{Q}_i) = (c \leq i)$.

Model 14 uses one clock, that is reset at every transition. The invariant of a quotient state $\mathcal{Q}_i$ is $c \leq i$, because i is the time that a trigger is sure to occur; hence $c = i$ is the clock constraint associated with this uncontrolled action. For the controlled, early triggering actions, the transition is enabled at discrete instants satisfying $c = k$, for $k < i$.

[2] See Remark 10: the definite cases are discarded.

[3] Additionally, we relaxi the strict inequalities with non-strict ones, so that it can fit the semi-definite programming formulation.

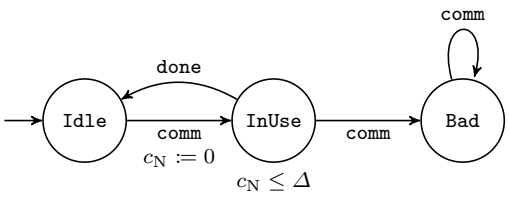

Fig. 1. TGA of a shared network.

5.2 Network and NCS models

For scheduling, we follow the same strategy as described in Mazo Jr et al. (2018), using the same network model as theirs, with a minor technical change [4]:

Model 15. (Network TGA, adapted from Mazo Jr et al. (2018)). The model is the TGA $\mathcal{N} = (\mathcal{L}, l_0, \mathcal{U}_{cN}, \emptyset, \mathcal{C}_N, \mathcal{E}_N, I_N)$ where

- $\mathcal{L} = \{\texttt{Idle}, \texttt{InUse}, \texttt{Bad}\}$;
- $\mathcal{U}_{cN} = \{\texttt{comm}, \texttt{done}\}$;
- $\mathcal{C} = \{c_N\}$;
- $\mathcal{E}_N = \{(\texttt{Idle}, \texttt{true}, \texttt{comm}, \{c_N\}, \texttt{InUse}),$ $(\texttt{InUse}, c_N = \Delta, \texttt{done}, \emptyset, \texttt{Idle}),$ $(\texttt{InUse}, \texttt{true}, \texttt{comm}, \emptyset, \texttt{Bad}),$ $(\texttt{Bad}, \texttt{true}, \texttt{comm}, \emptyset, \texttt{Bad})\}$;
- $I_N(\texttt{InUse}) = (c_N \leq \Delta)$,

where Δ is the maximum channel occupancy time.

Model 15 is represented in Fig. 1. The state `Bad` is reached if a second communication happens while the channel is still occupied by the first.

To model the NCS, we build an NTGA of the two or more traffic models $\mathcal{A}_i$ with the network model $\mathcal{N}$. What remains to be done is synchronizing the correct actions. For this, we add a synchronization channel called `up`, which is used as follows:

- every `early` and `trigger` actions of each traffic model $\mathcal{A}_i$ fires the synchronizing action `up!`;
- every `comm` action of the network model $\mathcal{N}$ takes the synchronizing action `up?`.

While avoiding the `Bad` state is necessary, we also want that the number of early triggers is small, so as to benefit from the communication savings of ETC. For that, we introduce an integer variable $e, 0 \leq e \leq E$, representing an accumulated "earliness" of communications, with E as the maximum allowed earliness. It is essentially a bounded integrator that increases every time an early trigger is done and decreases when a natural trigger happens. It starts at zero and is updated as

$$e \leftarrow \max(0, \min(E, e + r(k - i) - \bar{e})) \tag{11}$$

for every `trigger` or `early` transition from any traffic model, from quotient state $\mathcal{Q}_i$ when $c = k$. The parameters $r \in \mathbb{N}_+$ and $\bar{e} \in \mathbb{N}_+$ represent the cost of a time unit and a reference value for e, respectively. The earlier the trigger is, the higher the cost incurred. Parameter $\bar{e}$ is necessarily positive so that natural triggers discount e. Like any arithmetic on bounded integers, the evolution of e can be represented as an automaton itself. [5]

As a final note, remember that the time in model $\mathcal{A}$ is normalized w.r.t. the check time h. When composing the NTGA, one needs to put the clocks and their constraints in the same time scale.

5.3 Strategies for schedulers

In UPPAAL Tiga, strategies can be generated so as to guarantee certain specifications. We refer the reader to the manual of UPPAAL Tiga (Behrmann et al., 2007) for the complete list. In our case, we want that the NTGA never enters state `Bad` of $\mathcal{N}$, while keeping the earliness below a certain threshold E. This can be achieved by setting the specification `strategy safe = control: A[] not network.Bad and e < E`. The resulting strategy maps the locations of each automaton and their clock valuations into the decision of whether to trigger early or not. Therefore, a scheduler that implements such strategy needs to determine online the regions $\mathcal{Q}_i$ that the state of each system belongs to, and keep track of how much time elapsed since the last communication of each plant.

6. NUMERICAL RESULTS

Consider two copies of a linearized batch reactor, taken from Donkers (2011), of the form (1) with

$$\begin{aligned} \boldsymbol{A}_i &= \begin{bmatrix} 1.38 & -0.208 & 6.715 & -5.676 \\ -0.581 & -4.29 & 0 & 0.675 \\ 1.067 & 4.273 & -6.654 & 5.893 \\ 0.048 & 4.273 & 1.343 & -2.104 \end{bmatrix}, \\ \boldsymbol{B}_i &= \begin{bmatrix} 0 & 0 \\ 5.679 & 0 \\ 1.136 & -3.146 \\ 1.136 & 0 \end{bmatrix}, \quad i \in \{1, 2\}. \end{aligned} \tag{12}$$

Two different controllers $\boldsymbol{K}_i$ were designed for this plant using LQR with matrices $\boldsymbol{Q}_{\mathrm{LQR},1} = \boldsymbol{Q}_{\mathrm{LQR},2} = \mathbf{I}$ and $\boldsymbol{R}_1 = 0.2\mathbf{I}, \boldsymbol{R}_2 = 0.1\mathbf{I}$. The Lyapunov function chosen was the LQ cost, that is, setting $\boldsymbol{Q}_{\mathrm{lyap},i} = \boldsymbol{Q}_{\mathrm{LQR},i} + \boldsymbol{K}_i^\mathsf{T} \boldsymbol{R}_i \boldsymbol{K}_i$ and solving the continuous-time Lyapunov equation for $\boldsymbol{P}_i$. We used a triggering condition based on the Lyapunov function, so as to guarantee that

$$\dot{V}_i(t) \leq -\rho_i \boldsymbol{\xi}_i(t)^\mathsf{T} \boldsymbol{P}_i \boldsymbol{\xi}_i(t),$$

for some $0 < \rho_i < 1$. We set $\rho_1 = \rho_2 = 0.8$. This triggering condition can be expressed in quadratic form (2) with

$$\boldsymbol{Q}_i = \begin{bmatrix} \boldsymbol{A}_i^\mathsf{T} \boldsymbol{P}_i + \boldsymbol{P}_i \boldsymbol{A}_i + \rho_i \boldsymbol{Q}_{\mathrm{lyap},i} & \boldsymbol{P}_i \boldsymbol{B}_i \boldsymbol{K}_i \\ \boldsymbol{K}_i^\mathsf{T} \boldsymbol{B}_i^\mathsf{T} \boldsymbol{P}_i & \mathbf{0} \end{bmatrix}.$$

In both cases, $h_1 = h_2 = h = 0.01$; following Remark 10, we obtained natural maximum inter-event times at $\bar{k}_1 = 19$ and $\bar{k}_2 = 16$ by imposing that $\boldsymbol{N}(k)$ have its largest eigenvalue bigger than 10^{-3}. Likewise, both have MIETs greater than 1: $\underline{k}_1 = 6$, $\underline{k}_2 = 4$.

To build Model 14 for each control loop, we used Python with Numpy, Scipy and control packages, and CVXPY

[4] The difference of this model with respect to Mazo Jr et al. (2018) is that, here, all actions are controlled. We do this because of how NTGA are composed in UPPAAL Tiga: if an uncontrolled edge is synchronized with a controlled edge, the composed edge is uncontrolled. When we compose the traffic models with the network model, we want the `early` communications to be controlled, and the `trigger` ones not to.

[5] UPPAAL Tiga allows one to use integer variables, and it performs the necessary operations automatically.

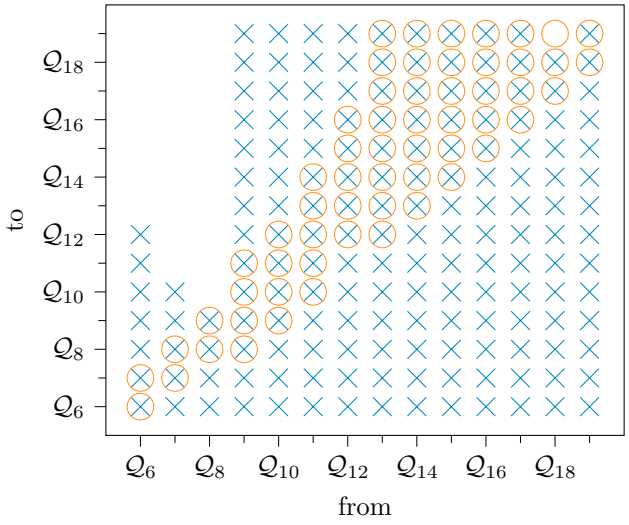

Fig. 2. Transition relations of $\mathcal{S}^*_{/\mathcal{R}}$ of loop 1, for `trigger` actions (x) and `early` actions (o) with $k = 1$.

(Diamond and Boyd, 2016) with solver SCS (O'Donoghue et al., 2017) to solve the semidefinite problems involved. The whole process of computing matrices $\boldsymbol{N}(k)$ and solving the semidefinite problems took 46.64 seconds for loop 1 and 31.51 seconds for loop 2. The computer used is a MacBook Pro with a 3.1 GHz Intel Core i5 CPU and memory of 8 GB, 2133 MHz LPDDR3. The resulting transition relation for closed-loop system 1 is represented in Figure 2. As one can see, there is a significant amount of nondeterminism introduced by this model, especially for high triggering times.

A series of scripts was used to generate the XML files that are used for TGA models in UPPAAL Tiga. We used all times in the NTGA relative to h, and set $\Delta = 1$. The earliness parameters for Eq. 11 were $r = 2, \bar{e} = 1, E = 2$. These parameters allow the scheduler to trigger one step earlier at every two communications.

The strategy was solved in UPPAAL STRATEGO (David et al., 2015) version 4.1.20-5, which includes all functionalities of UPPAAL Tiga. It took 0.864 s to find a solution. The generated strategy is too long to be reproduced in this paper, but we give below one example of when an early trigger has to occur:

If System 1 is in $\mathcal{Q}_6$, System 2 is in $\mathcal{Q}_4$, and $e = 0$,
when $c_1 = 5$ and $c_2 \in \{1, 2, 3\}$, do `early` on System 1;
when $c_2 = 3$ and $c_1 \in \{3, 4, 5\}$, do `early` on System 2,

where c_i represents the clock valuation of system i. As one can see, the strategy is not deterministic. In the example above, the early trigger can be executed on any of the loops when $(c_1, c_2) = (5, 3)$. In such case, the scheduler must arbitrate who triggers.

Figures 3 and 4 show the results of a simulation of the two control loops executing in parallel with the communication managed by the synthesized scheduler. The initial conditions are $\boldsymbol{\xi}_1(0) = [1\ -1\ 1\ -1]^\mathsf{T}$ and $\boldsymbol{\xi}_2(0) = [1\ 2\ 3\ 4]^\mathsf{T}$. The first pair of communications were arbitrated on a round-robin fashion. Figure 5 shows the communication pattern of the NCS. As we can see, both systems' states converge to zero, while there is no conflict in communications. As

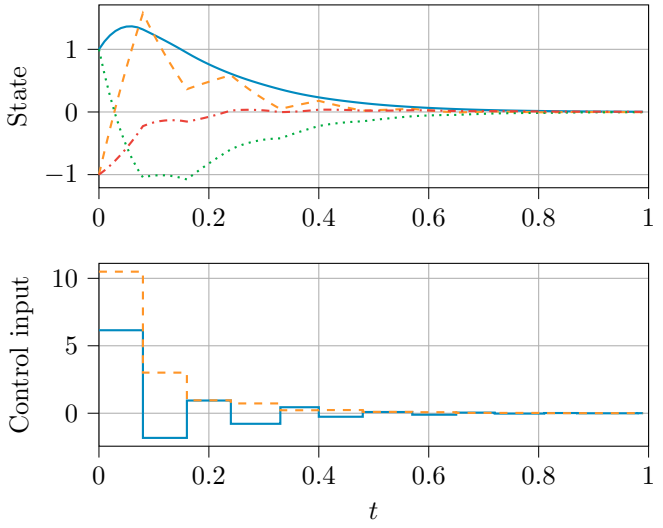

Fig. 3. Trajectories of $\boldsymbol{\xi}_1(t)$ (top) and $\boldsymbol{K}_1\hat{\boldsymbol{\xi}}_1(t)$ (bottom).

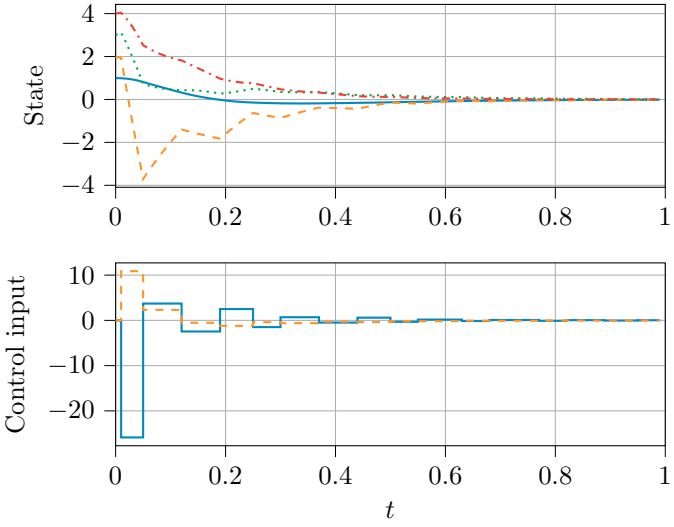

Fig. 4. Trajectories of $\boldsymbol{\xi}_2(t)$ (top) and $\boldsymbol{K}_2\hat{\boldsymbol{\xi}}_2(t)$ (bottom).

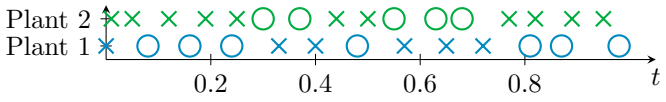

Fig. 5. Communication pattern of the simulated NCS: 'x' marks represent event triggers, while 'o' marks represent early triggers.

designed through the earliness mechanism, about half of the communications are early triggers, and half are natural, event triggers.

7. CONCLUSIONS

In this paper, we presented a method to build a quotient model of the traffic generated by PETC, and how to augment it and use it for scheduling of multiple PETC loops. The quotient model has many advantages with respect to related work: first, it is a (exact) simulation instead of an approximate simulation; and second, it avoids the combinatorial explosion created by isotropic partitioning of the state space. The state space and output

map of the quotient model can be easily created straight from the PETC and system matrices, requiring no solution of LMIs or other optimization problems. The transition relations do require semidefinite problems to be solved, but only one per transition, with no reachability tools required. It is relatively fast to compute, and the models generated are reasonably small. The use of TGA models for scheduling of ETC had already been demonstrated in Mazo Jr et al. (2018); here, we demonstrate that they can also be done for PETC, and argue that it is in fact simpler to do so.

Among the disadvantages of our solution is the high nondeterminism of the generated models. The state-space partitions are based solely on the output function, and each region seems to be large enough that, after some time, many regions can be reached. A highly nondeterministic traffic model can hamper the generation of strategies, as the predictability of the model after multiple steps gets smaller. One solution we are exploring is partitioning the regions further using backwards reachability. A second disadvantage of this approach, shared with Mazo Jr et al. (2018), is that the size of the NTGA state space grows exponentially with the number of control loops. This can make solving the scheduling problem impracticable. Solving strategies for TGA is EXPTIME-complete (Asarin et al., 1998), so controlling the size of the (N)TGA is paramount. Methods to do so are subject of future research. A third point of attention is addressing optimality of these schedulers. Parameterizing the earliness function (11) is not always trivial. Even so, finding a scheduler that minimizes the interventions is still an open problem. Priced TGA could be used, but their undecidability for games with three clocks has been proven by Bouyer et al. (2006), putting a roadblock in that direction. Approximate solutions using stochastic priced TGA (David et al., 2015) are currently being explored.

REFERENCES

Anta, A. and Tabuada, P. (2008). Self-triggered stabilization of homogeneous control systems. In *American Control Conference, 2008*, 4129–4134. IEEE.

Asarin, E., Maler, O., Pnueli, A., and Sifakis, J. (1998). Controller synthesis for timed automata. *IFAC Proceedings Volumes*, 31(18), 447–452.

Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K.G., and Lime, D. (2007). Uppaal tiga user-manual. *Aalborg University*.

Bengtsson, J. and Yi, W. (2004). Timed automata: Semantics, algorithms and tools. In *Advanced Course on Petri Nets*, 87–124. Springer.

Bouyer, P., Brihaye, T., and Markey, N. (2006). Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98(5), 188–194.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

David, A., Jensen, P.G., Larsen, K.G., Mikučionis, M., and Taankvist, J.H. (2015). Uppaal stratego. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 206–211. Springer.

Diamond, S. and Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83), 1–5.

Dolk, V., Borgers, D.P., and Heemels, W. (2017). Output-based and decentralized dynamic event-triggered control with guaranteed lp-gain performance and zeno-freeness. *IEEE Transactions on Automatic Control*, 62(1), 34–49.

Donkers, M. (2011). *Networked and event-triggered control systems*. Ph.D. thesis, TU Eindhoven.

Donkers, M., Tabuada, P., and Heemels, W. (2014). Minimum attention control for linear systems. *Discrete Event Dynamic Systems*, 24(2), 199–218.

Fu, A. and Mazo Jr., M. (2018). Traffic models of periodic event-triggered control systems. *IEEE Transactions on Automatic Control*.

Girard, A. (2015). Dynamic triggering mechanisms for event-triggered control. *IEEE Transactions on Automatic Control*, 60(7), 1992–1997.

Heemels, W.P.M.H., Donkers, M.C.F., and Teel, A.R. (2013). Periodic event-triggered control for linear systems. *IEEE Transactions on Automatic Control*, 58(4), 847–861.

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

Kolarijani, A.S. and Mazo Jr, M. (2015). Traffic characterization of lti event-triggered control systems: a formal approach. *arXiv preprint arXiv:1503.05816*.

Kolarijani, A.S. and Mazo Jr, M. (2016). A formal traffic characterization of LTI event-triggered control systems. *IEEE Transactions on Control of Network Systems*.

Mazo Jr, M., Kolarijani, A.S., Adzkiya, D., and Hop, C. (2018). Abstracted models for scheduling of event-triggered control data traffic. In *Control Subject to Computational and Communication Constraints*, 197–217. Springer.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. (2017). SCS: Splitting conic solver, version 2.1.1. `https://github.com/cvxgrp/scs`.

Park, J. and Boyd, S. (2017). General heuristics for nonconvex quadratically constrained quadratic programming. *arXiv preprint arXiv:1703.07870*.

Peng, C. and Yang, T.C. (2013). Event-triggered communication and h-∞ control co-design for networked control systems. *Automatica*, 49(5), 1326–1332.

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

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

Wang, X. and Lemmon, M.D. (2008). Event design in event-triggered feedback control systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 2105–2110. IEEE.