Timed Automata as synchronization abstractions for aperiodic control systems

Manuel Mazo Jr, Gabriel Gleizer, Giannis Delimpaltadakis, Paul Schaalwijk.
Delft University of Technology

June 10, 2019

1 Introduction

Control systems impose stringent real-time requirements on the communication systems that support them. Delays and jitter need to be confined to precise levels in order to not break the stability properties of the systems being controlled. In the quest to efficiently share communication channels among diverse applications, much attention has been put on the control community to reduce the required *amount* of bandwidth that a control task demands. Following this goal aperiodic control techniques have been proposed to either adapt to the available bandwidth [1] or to close the loop when strictly necessary to guarantee some level of performance. In the latter category fall a great amount of diverse techniques under the umbrella of event-based control. In particular event-triggered control techniques rely on letting the sensors decide when fresh measurements are required at the controller side. Sensors do this by checking appropriately designed *triggering conditions*, and thus the name of these techniques.

In the present work we argue that reducing the amount of required transmissions at the cost of unpredictable (aperiodic) traffic does not solve any of the targeted problems by ETC: scarcity of bandwidth or energy usage on wireless control systems. Real-time task scheduling is as good as the models of the timing (traffic) of the tasks at hand. In order to provide a timely response to events, when no models of the ETC traffic are available, overly conservative bandwidth allocations are employed. These allocations are often even periodic [2] thus defeating the purpose of ETC techniques. Furthermore, in wireless settings the diverse network devices need to schedule listening time on their radios to react to events generated at other points of the network [3]. Again, not having a model of the traffic, or having a too conservative one, results in these devices spending large amounts of time listening, which entails a large energy consumption[1]. These limitations of ETC can be observed in e.g. [6] where a tenfold reduction of communication resulted in not even doubling the battery savings with respect to a periodic counterpart.

Thus, it seems that in order to extract the true potential of ETC systems one needs to obtain models of the generated traffic that could be used for scheduling purposes. In the current work I'll present our latest advances on addressing that challenge. The approach we take is to construct Timed Game Automata (TGA)models of the traffic of ETC systems. Composing such models results in what is called Networks of TGA, in which one can solve diverse type of games to automatically synthesize schedulers for the network of ETC controllers sharing a communication channel.

[1]Note that radio chips consume, in general, a similar amount of power for transmission and reception of data [4, 5], but in the latter the radio typically spends a longer time 'on' with the subsequent larger energy impact than transmission has.

2 Technical details

In order to study the communication traffic generated by an NCS, consider a continuous-time control system:

$$\dot{\xi}(t) = f(\xi(t), \upsilon(t)), \;\; \xi(t) \in \mathbb{R}^n, \; \upsilon(t) \in \mathbb{R}^m \qquad (1)$$

and a given state-feedback controller[2] $\upsilon(t) = k(\xi(t))$ designed to guarantee e.g. asymptotic stability of the closed loop system: $\dot{\xi} = f(\xi, k(\xi(t))$. Nowadays, most controllers are implemented in a sampled-data fashion, which can be modelled by:

$$\Sigma : \dot{\xi}(t) = f(\xi(t), k(\xi(t_k)), \quad t \in [t_k + \Delta_k, t_{k+1} + \Delta_{k+1}[\qquad (2)$$

Denote from here on the state of system (2) at time $t_0 + \tau$ for $\xi(t_0) = x$ by $\xi_x(t_0 + \tau)$. The computed control signal at time instant t_k is held constant until the next controller update t_{k+1}, when the control action is recomputed. The measurements may be delayed, from generation at the sensors to application in the actuation, by a certain amount of time Δ_k, typically assumed to be bounded, i.e. $\Delta_k < \bar{\Delta}$, $\forall k \in \mathbb{N}$. The selection of the update times sequence $\mathrm{T}_\Sigma := \{t_k\}_{k\in\mathbb{N}}$, together with $\mathrm{T}_\Sigma^\Delta := \{\Delta_k\}_{k\in\mathbb{N}}$ is critical to guarantee any prescribed performance of the controller implementation. Note that the sequence T_Σ determines for an NCS the times at which the traffic would be generated, and T_Σ^Δ (or simply $\bar{\Delta}$) imposes the maximum delay that transmission and manipulation of measurements can introduce. Therefore, these two sequences determine the occupancy of the communication channel by the control task.

In practice, ETC implementations require some computation power at the sensors in order to check the triggering condition. Whenever the sensors trigger a new controller update, measurements are sent to the controller which updates the control action υ. Formally, an ETC system is the combination of a sampled-data system as (2) with a triggering condition $\Phi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ determining the controller update sequence:

$$\mathrm{T}_\Sigma^{etc} := \{t_k\}_{k\in\mathbb{N}}, \qquad (3)$$
$$\text{where } t_{k+1} = \min\{t > t_k | \Phi(\xi(t), \xi(t_k)) \geq 0\}.$$

A widely used Φ function is e.g. $||\xi(t_k) - \xi(t)|| - \sigma||\xi(t)||$. Usually, the triggering conditions are designed so that the system can tolerate a certain maximum delay $\bar{\Delta}$. Observe how, for ETC systems, the sequence T_Σ^{etc} is only determined in run-time depending on the initialization of the system. Furthermore, if disturbances are present, e.g. process noise, the update times also depend on the particular realization of the disturbance.

2.1 Efficient scheduling of ETC to improve energy and bandwidth use

Having the sequence of update times a priori, as is the case with T_Σ^T, enables tight scheduling of communications [8]. As T_Σ^{etc} is unknown in advance, conservative bandwidth allocations are employed to guarantee the delivery of measurements according to the tolerable deadlines imposed by $\bar{\Delta}$ [2]. Furthermore, in wireless settings, not knowing $\mathrm{T}_{\Sigma,j}^{etc}$ in advance forces the wireless devices to spend a prohibitive amount of time listening for possible events, with the associated high energy cost[3] [9]. As a result, while ETC techniques reduce drastically the communication traffic generated, the effective bandwidth freed for other applications and the impact on energy savings is much more limited, see e.g. [10].

2.1.1 ETC Timing models

Symbolic abstractions. As in the field of *symbolic/formal methods for control* [11] we propose to replace a *concrete* system of large complexity, in particular an infinite system, by a simpler finite one amenable to automatic algorithmic manipulation: the *abstraction* of the concrete one. We embed

[2]The choice of state-feedback is made for simplicity of the presentation. Extensions to output-feedback and even dynamic controllers can similarly be formulated, see e.g. [7].

[3]Note that radio chips consume, in general, a similar amount of power for transmission and reception of data [4, 5], but in the latter the radio typically spends a longer time "on" with the subsequent larger energy impact than transmission has.

both the concrete and abstracted systems under the formalism of a *generalized transition system*:

Definition 1 *(System [11]) A system is a sextuple* $S = (X, X_0, U, \longrightarrow, Y, H)$ *consisting of:*

- *a set of states* X*;*
- *a set of initial states* $X_0 \subseteq X$*;*
- *a set of inputs* U*;*
- *a transition relation* $\xrightarrow{S} \subseteq X \times U \times X$*;*
- *a set of outputs* Y*;*
- *an output map* $H : X \to Y$*.*

The term finite-state (or infinite-state) system is thus determined by X being a finite (or infinite) set. Furthermore, a system $\mathcal{S}$ is said to be a metric system if the set of outputs Y is equipped with a metric $d : Y \times Y \to \mathbb{R}_0^+$. By external behaviour $\mathcal{B}_S$ of a system S, it is understood the set of all possible (potentially infinite) sequences $\mathbf{y} := y_1, y_2, \ldots$ allowed by the system S, i.e. $y_k \to y_{k+1} \in \xrightarrow{S}$. Sequences $\mathbf{y}$ are often denoted as the *traces* of the system. To make these abstractions useful, one needs to establish formal relationships between the possible external behaviours of the concrete system, in our case S_Σ, and those of the abstract system $\hat{S}_\Sigma$. We make use of the notion of ϵ-approximate simulation relation, denoted as $\mathcal{S} \preceq^\epsilon \hat{\mathcal{S}}$ when $\hat{S}$ approximately simulates S, which (under some technical assumptions) implies an approximate behaviour containment, c.f. [11]:

$$\forall \mathbf{y} \in \mathcal{B}_S,\ \exists\, \hat{\mathbf{y}} \in \mathcal{B}_{\hat{S}} \text{ such that } d(\mathbf{y}, \hat{\mathbf{y}}) \leq \epsilon.$$

Symbolic abstractions of timing systems. Going back to the original problem, consider now the generalized transition system, modelling *exactly* the timing of an ETC system: $\mathcal{S}_\Sigma = (X, X_0, U, \longrightarrow, Y, H)$ where

- $X = \mathbb{R}^n$ is the set of states;
- $X_0 \subseteq \mathbb{R}^n$ is the set of initial states;
- $U = \varnothing$ is the set of inputs, i.e. the system is autonomous;
- $\longrightarrow \in X \times U \times X$ such that $\forall x, x' \in X : (x, x') \in \longrightarrow$ iff $\xi_x(\tau(x)) = x'$, defines the evolution (possible transitions of states) of the system;
- $Y \subset \mathbb{R}^+$ is the (observable) output of the system;
- $H : \mathbb{R}^n \to \mathbb{R}^+$ where $H(x) = \tau(x)$ defines the mapping from states to outputs.

In the previous, we denote by $\tau(x)$ the time to the next event when the current measurement was x, i.e.

$$\tau(x) := t_{k+1} - \min\{t > t_k | \Phi(\xi_x(t), x) \geq 0\}.$$

The system $\mathcal{S}$ generates as traces all possible sequences of inter-event intervals that the system (2) with triggering condition (3) can exhibit [12]. Therefore, $\mathcal{S}_\Sigma$ is the ideal model to predict exactly T_Σ^{etc}. However, two obstacles impede the use of $\mathcal{S}_\Sigma$:

1. the system (2) may not be analytically integrable and thus $\mathcal{S}_\Sigma$ may be impossible to be constructed;
2. in the few cases in which one can integrate the systems' dynamics (e.g. for LTI systems) the resulting model is still infinite ($X = \mathbb{R}^n$ includes an infinite number of points[4]).

Having an infinite model hampers the application of automated and/or optimal approaches to synthesize schedulers with the said models. We construct finite abstractions of $\mathcal{S}_\Sigma$ in order to approximately predict T_Σ^{etc}, and employ the resulting models as the stepping stone to reach efficient energy and bandwidth use of ETC systems. In [12] we proposed the use of a modified notion of *quotient system* particularly suited to construct abstractions of infinite timing models as S_Σ:

Definition 2 *(Power Quotient System) [12] Let* $\mathcal{S} = (X, X_0, U, \longrightarrow, Y, H)$ *be a system and* R *be an equivalence relation on* X*. The power quotient of* $\mathcal{S}$ *by* R*, denoted* $\mathcal{S}_{/R}$*, is the system* $(X_{/R}, X_{/R0}, U_{/R}, \xrightarrow[/R]{}, Y_{/R}, H_{/R})$ *with:*

[4]Even if one considers $X \subset \mathbb{R}^n$ a compact subset of the possible states of system (2).

- $X_{/R} = X/R$;
- $X_{/R0} = \{x_{/R} \in X_{/R} | x_{/R} \cap X_0 \neq \varnothing\}$;
- $U_{/R} = U$;
- $Y_{/R} \subset 2^Y$;
- $H_{/R}(x_{/R}) = \bigcup_{x \in x_{/R}} H(x)$,
- $(x_{/R}, u, x'_{/R}) \in \underset{/R}{\rightarrow}$ *if* $\exists (x, u, x') \in \rightarrow$ *in* $\mathcal{S}$ *with* $x \in x_{/R}$ *and* $x' \in x'_{/R}$;

where X/R denotes the quotient set of X determined by the equivalence relation R. The main difference with respect to the traditional definition of a quotient system is that this new notion allows for outputs to be infinite sets. Then, in order to compare with behaviours from S_Σ (whose outputs are singletons) we extend the Hausdorff distance d_H between sets to compare traces of S_Σ and $\mathcal{S}_{\Sigma/R}$. The following lemma, establishes the formal relationship between a power quotient system and the original system it is abstracting.

Lemma 1 *[12] Let $\mathcal{S}$ be a metric system, R be an equivalence relation on X, and let the metric system $\mathcal{S}_{/R}$ be the power quotient system of $\mathcal{S}$ by R. For any $\epsilon \geq \max_{x \in x_{/R}, x_{/R} \in X/R} d(H(x), H_{/R}(x_{/R}))$, with d the Hausdorff distance over the set 2^Y, $\mathcal{S}_{/R}$ ϵ-approximately simulates $\mathcal{S}$, i.e. $\mathcal{S} \preceq^\epsilon \mathcal{S}_{/R}$.*

This Lemma indicates that constructing a power quotient system, the resulting abstraction $\hat{S}_\Sigma$ approximately simulates the exact timing model S_Σ.

The proposed abstractions of the ETC timing behaviour $\mathcal{S}_{\Sigma/\mathcal{P}}$, following the approach just sketched, take the form $\mathcal{S}_{\Sigma/\mathcal{P}} = (X_{/\mathcal{P}}, X_{0/\mathcal{P}}, U_{/\mathcal{P}}, \underset{/\mathcal{P}}{\longrightarrow}, Y_{/\mathcal{P}}, H_{/\mathcal{P}})$ where:

- $X_{/\mathcal{P}} = \mathbb{R}^n_{/\mathcal{P}} := \{\mathcal{R}_1, \ldots, \mathcal{R}_q\}$;
- $X_{0/\mathcal{P}} = \{\mathcal{R}_i \,|\, X_0 \cap \mathcal{R}_i \neq \emptyset\}$;
- $U_{/\mathcal{P}} = \varnothing$, i.e. the system is autonomous;
- $Y_{/\mathcal{P}} \subset 2^Y \subset \mathbb{IR}^+$ (positive intervals of real numbers);
- $H_{/\mathcal{P}}(x_{/\mathcal{P}}) = [\min_{x \in x_{/\mathcal{P}}} H(x), \max_{x \in x_{/\mathcal{P}}} H(x)] := [\underline{\tau}_{x_{/\mathcal{P}}}, \bar{\tau}_{x_{/\mathcal{P}}}]$.
- $(x_{/\mathcal{P}}, x'_{/\mathcal{P}}) \in \underset{/\mathcal{P}}{\longrightarrow}$ if $\exists x \in x_{/\mathcal{P}}$, $\exists x' \in x'_{/\mathcal{P}}$ such that $\xi_x(H(x)) = x'$;

In that model:

- $\mathcal{R}_i$ bundles of states of the original system, subsets of $\mathbb{R}^n$, defining an equivalence class of $\mathcal{P}$;
- $\underline{\tau}_{x_{/\mathcal{P}}}$ denotes the smallest of the inter-event times $\tau(x)$ among all the points of the equivalence class $x_{/\mathcal{P}} \ni x$;
- $\bar{\tau}_{x_{/\mathcal{P}}}$ denotes the largest of the inter-event times $\tau(x)$ among all the points of the equivalence class $x_{/\mathcal{P}} \ni x$;

Intuitively, the state of $\mathcal{S}_{\Sigma/\mathcal{P}}$ represents the location of the last state measurement received, i.e. to which $\mathcal{R}_i$ region it belongs, and produces as output the future time interval in which the next event must occur. This model let us also predict all the possible future states[5] of the system, and consequently the possible future intervals of event occurrences.

To construct such an abstraction three remaining questions must be addressed:

(i) how to select an appropriate equivalence relation $\mathcal{P}$,

(ii) how to compute the respective intervals $[\underline{\tau}_{x_{/\mathcal{P}}}, \bar{\tau}_{x_{/\mathcal{P}}}]$, and

(iii) how to determine if there is a transition between a pair of abstract states $(x_{/\mathcal{P}}, x'_{/\mathcal{P}})$.

TPGA as models for ETC traffic. The abstractions $\mathcal{S}_{\Sigma/\mathcal{P}}$ are semantically equivalent to *timed automata* (TA) [13], a class of (decidable) hybrid systems that have been studied in depth on the theoretical computer science literature. TA are essentially an extension of classical automata [14] in which transitions between states (locations) are enabled and/or

[5]Note that in general the set of possible states $\mathbf{Post}(x) := \{x' \,|\, (x, x') \in \underset{/\mathcal{P}}{\longrightarrow}\}$ may contain more than one possible (future) state.

triggered by conditions depending on clocks. For a set of clocks $\mathcal{C}$, denote by $\mathcal{B}(\mathcal{C})$ the set of all possible constraints imposed on them. Consider a slight modification of the original TA definition, denoted Timed Safety Automata (TSA), formally described as follows:

Definition 3 *(Timed Safety Automata [15]) A* timed safety automata *is a tuple* $\mathcal{A} = (L, L_0, \mathsf{Act}, \mathcal{C}, E, \mathsf{Inv})$[6] *where*

- L *is a finite set of locations (or discrete states);*
- $L_0 \subseteq L$ *is a set of initial locations*[7]*;*
- Act *is the set of actions;*
- $\mathcal{C}$ *is the set of clocks;*
- $E \subseteq L \times \mathcal{B}(\mathcal{C}) \times \mathsf{Act} \times 2^{\mathcal{C}} \times L$ *is the set of transitions.*
- $\mathsf{Inv} : L \longrightarrow \mathcal{B}(\mathcal{C})$ *assigns invariants to locations.*

In layman terms, the state of a TSA is defined by a pair (l, u) where l is a discrete location and u the current value of a clock. The TSA can move from one state to the following either by transiting to a new discrete location (discrete transition) or letting time run (flow). A discrete transition $(l, \mathbf{b}, \sigma, \mathbf{c}, l')$ is allowed if it belongs to E, action σ has been issued, and the transition is enabled, i.e. whenever all the clocks c_i satisfy the constraint (*guard*): $\mathbf{b} \in \mathcal{B}(\mathcal{C})$. After the transition (some of) the clocks are reset to the values specified by $\mathbf{c}$. The system may flow only for as long as the *invariants*, clock conditions, assigned to the current discrete location hold. Note that the *guards* of a command assert necessary conditions for the transitions to take place, while *invariants* assert sufficient conditions for transitions to take place, and must not be violated by letting time advance. Thus, invariants establish upper bounds for the time to the next transition (to a different location) [16].

[6]Although technically not necessary, for clarity we extend the original definition to explicitly state the actions' and clocks' sets.

[7]Note that for convenience, and without any impact on the expressivity of the model, we slightly modify the definition of [15] to allow several possible initial locations as in the original works of [13, 16].

So far, the models proposed are not making use of any input set U (in the generalized transition system formalism) or set of actions Act (in the TSA formalism). However, in order to be able to "control" the ETC timing (e.g. for scheduling) with these models we need to introduce additional elements in the set Act. The new set of actions Act_c that we propose allow the scheduler to either: (1) change the type of triggering condition that will determine the next event, or (2) force the time of the next event. With these two types of "knobs" a scheduler can, in order to achieve a collision free schedule, stretch/compress and shift the time intervals at which an ETC loop may request a shared communication channel. Additionally, one may need a mechanism to incorporate discrete actions that are not controllable Act_u. Incorporating uncontrollable actions enables the possibility of describing games between competing TSA (as they do, e.g. when trying to access a shared resource). *Timed game automata* (TGA) [17] provide such an extension:

Definition 4 *(TGA [17]) A* timed game automaton TGA *is a septuple* $(L, \ell_0, \mathsf{Act}_c, \mathsf{Act}_u, C, E, \mathsf{Inv})$ *where:*

- $(L, \ell_0, \mathsf{Act}_c \cup \mathsf{Act}_u, C, E, \mathsf{Inv})$ *is a timed automaton;*
- Act_c *is a set of controllable actions;*
- Act_u *is a set of uncontrollable actions; and*
- $\mathsf{Act}_c \cap \mathsf{Act}_u = \varnothing$.

Without changing the applied controller, event-triggered controllers can, simply by changing the triggering condition employed, trade performance for frequency of events. Thus, it seems natural to capture (as a cost) the impact of the scheduling actions on the control performance. *Timed Priced Game Automata* (TPGA) [18] let us include both the control performance via cost rates, and the cost of communications via discrete costs:

Definition 5 *(TPGA [18]) A* timed priced game automaton TPGA *is a tuple* $(L, \ell_0, \mathsf{Act}_c, \mathsf{Act}_u, C, E, \mathsf{Inv}, P)$ *where* $(L, \ell_0, \mathsf{Act}_c \cup \mathsf{Act}_u, C, E, \mathsf{Inv})$ *is a timed game automaton and* $P : L \cup E \to \mathbb{N}_0$ *assigns cost rates and costs to locations and edges, respectively.*

2.1.2 Automatic synthesis of schedulers

Scheduling with TPGA. The main idea to address the scheduling of ETC loops, together with other potential real-time tasks, is to bring all the elements of the problem into the TPGA formalism:

1. models of the ETC loops;
2. models of other real-time tasks, for which there's already work available, see e.g. the examples in [19];
3. a model of the channel access, as sketched in what follows.

The following very simple model of a shared communication network, is used to illustrate the type of modelling as a TGA proposed. TGA^{net}, depicted graphically in figure 1 and formalized as follows [20], represents such a channel:

Definition 6 *Let* Δ *represent the maximum channel occupancy time, a timed game automaton associated with the communication network is given by* $\mathsf{TGA}^{net} = (L^{net}, \ell_0^{net}, \mathsf{Act}_c^{net}, \mathsf{Act}_u^{net}, C^{net}, E^{net}, \mathsf{Inv}^{net})$ *where*

- $L^{net} = \{Idle, InUse, Bad\}$;
- $\ell_0^{net} = Idle$;
- $\mathsf{Act}_c^{net} = \{*\}$;
- $\mathsf{Act}_u^{net} = \{up?\}$;
- $C^{net} = \{c\}$;
- $E^{net} = \{(Idle, true, up?, \{c\}, InUse), (InUse, c = \Delta, *, \emptyset, Idle), (InUse, true, up?, \emptyset, Bad), (Bad, true, up?, \emptyset, Bad)\}$;
- $\mathsf{Inv}^{net}(InUse) = \{c \mid 0 \leq c \leq \Delta\}$, $\mathsf{Inv}^{net}(Idle) = \{c \mid c \geq 0\}$, $\mathsf{Inv}^{net}(Bad) = \{c \mid c \geq 0\}$.

The guard "true" represents a condition that is always satisfied, for example $c \geq 0$.

This model TGA^{net} has only three locations *Idle*, *InUse* and *Bad*, where the initial location is *Idle* (cf. Fig. 1). These locations are used to represent the three main possible states of the network:

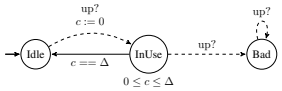

Figure 1: A timed game automaton modeling the shared communication network. The solid and dashed arrows represent controllable and uncontrollable edges, respectively.

- *Idle* represents the network being available,
- *InUse* represents the network being used by a real-time task, and
- *Bad* represents a conflict occured (two tasks tried to access the shared network at the same time).

The active location changes from *Idle* to *InUse* when a task requests access to the channel (*up* action), which forces the reset of the clock variable c. The channel is occupied for Δ time units before the network is freed again, changing location to *Idle*. When in *InUse* and another task requests access, the active location moves to *Bad* (a blocking/absorbing location). This simple model is somewhat conservative, as it considers that every control loop occupies the channel the whole time Δ.

Once these different components are modelled as TPGA, the interconnection of the different models is attained by synchronizing (connecting) the controllable and uncontrollable actions from each of the TPGA. Finally, in this unified model of the NCS components, solutions to timed games [21] (as we have preliminary shown in [22]) or priced timed games [19] can be leveraged to prevent medium access collisions and to optimize the performance or energy consumption, respectively.

3 Current state of the program

In our quest to solve the described problems we are constructing tools that allow to:

- Construct Timed Automata from ETC designs, for both linear and non-linear systems

- Compose several abstractions for the same control system (e.g. with different triggering conditions) to construct Timed Game Automata
- Compose a set of TGA into an NTGA and synthesize schedulers automatically (relying on the UPPAAL Suite)
- Approximately solve optimization problems in which one can trade the convergence of controllers vs the amount of transmissions required to achieve such goal.

We would like to present and publicize the advances on all those different fronts, showcasing some of our currently developed tools.

References

[1] R. Bhattacharya and G. Balas, "Anytime control algorithm: Model reduction approach," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 5, pp. 767–776, 2004.

[2] L. L. Bello, E. Bini, and G. Patti, "Priority-driven swapping-based scheduling of aperiodic real-time messages over ethercat networks," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 741–751, 2015.

[3] M. Mazo Jr and P. Tabuada, "Decentralized event-triggered control over wireless sensor/actuator networks," *Automatic Control, IEEE Transactions on*, vol. 56, no. 10, pp. 2456–2461, 2011.

[4] A. Prayati, C. Antonopoulos, T. Stoyanova, C. Koulamas, and G. Papadopoulos, "A modeling approach on the TelosB WSN platform power consumption," *Journal of Systems and Software*, vol. 83, no. 8, pp. 1355 – 1363, 2010, performance Evaluation and Optimization of Ubiquitous Computing and Networked Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121210000087

[5] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 171–180. [Online]. Available: http://doi.acm.org/10.1145/958491.958512

[6] J. Araújo, M. Mazo Jr, A. Anta, P. Tabuada, and K. H. Johansson, "System architectures, protocols and algorithms for aperiodic wireless control systems," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 1, pp. 175–184, 2014.

[7] M. Donkers and W. Heemels, "Output-based event-triggered control with guaranteed L_∞-gain and improved and decentralized event-triggering," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1362–1376, 2012.

[8] G. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2011, vol. 24.

[9] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2002, pp. 1567–1576 vol.3.

[10] J. Araújo, M. Mazo Jr, A. Anta, P. Tabuada, and K. H. Johansson, "System architectures, protocols and algorithms for aperiodic wireless control systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 175–184, feb 2014.

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

[12] A. S. Kolarijani and M. Mazo Jr, "A Formal Traffic Characterization of LTI Event-triggered Control Systems," *Early Access: IEEE Transactions on Control of Network Systems.*, 2016. [Online]. Available: http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=6730648

[13] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[14] J. E. Hopcroft, R. Motwani, and J. D. Ullman, "Introduction to automata theory, languages, and computation, 2nd edition," *SIGACT News*, vol. 32, no. 1, pp. 60–65, Mar. 2001. [Online]. Available: http://doi.acm.org/10.1145/568438.568455

[15] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," in *Lectures on Concurrency and Petri Nets*. Springer, 2004, pp. 87–124.

[16] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," *Information and computation*, vol. 111, no. 2, pp. 193–244, 1994.

[17] A. Pnueli, E. Asarin, O. Maler, and J. Sifakis, "Controller synthesis for timed automata," in *Proc. IFAC Symposium on System Structure and Control. Elsevier*, Elsevier, Ed., 1998, pp. 469–474.

[18] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, "Priced timed automata: Algorithms and applications," in *International Symposium on Formal Methods for Components and Objects*. Springer, 2004, pp. 162–182.

[19] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen, "Optimal strategies in priced timed game automata," in *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 2004, pp. 148–160.

[20] D. Adzkiya and M. Mazo Jr, "Scheduling of event-triggered networked control systems using timed game automata," arXiv:1610.03729. [Online]. Available: https://arxiv.org/pdf/1610.03729.pdf

[21] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Efficient on-the-fly algorithms for the analysis of timed games," in *International Conference on Concurrency Theory*. Springer, 2005, pp. 66–80.

[22] A. S. Kolarijani, D. Adzkiya, and M. Mazo Jr, "Symbolic Abstractions for the Scheduling of Event-Triggered Control Systems," in *IEEE Conference on Decision and Control*, feb 2015, pp. 6153–6158.