MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

# Event-Driven Model Predictive Control of Timed Hybrid Petri Nets

Julvez, J.; Di Cairano, S.; Bemporad, A.; Mahulea, C.

TR2013-064 January 2013

**Abstract**

Hybrid Petri nets represent a powerful modeling formalism that offers the possibility of integrating in a natural way continuous and discrete dynamics in a single net model. Usual control approaches for hybrid nets can be divided into discrete-time and continuous-time approaches. Continuous-time approaches are usually more precise but can be computationally prohibitive. Discrete-time approaches are less complex but can entail mode-mismatch errors due to fixed time discretization. This work proposes an optimization-based event-driven control approach that applies on continuous time models and where the control actions change when discrete events occur. Such an approach is computationally feasible for systems of interest in practice and avoids mode-mismatch errors. In order to handle modelling errors and exogenous disturbances, the proposed approach is implemented in a closed-loop strategy based on event-driven model predictive control.

*International Journal of Robust and Nonlinear Control*

**Noname manuscript No.**
(will be inserted by the editor)

# Event-Driven Model Predictive Control of Timed Hybrid Petri Nets

**J. Júlvez · S. Di Cairano · A. Bemporad · C. Mahulea**

**Abstract** Hybrid Petri nets represent a powerful modeling formalism that offers the possibility of integrating in a natural way continuous and discrete dynamics in a single net model. Usual control approaches for hybrid nets can be roughly divided into discrete time and continuous time approaches. Continuous time approaches are usually more precise but can be computationally prohibitive. Discrete time approaches are less complex but can entail *mode-mismatch* errors. This work proposes an optimization-based *event-driven* control approach that applies on continuous time models and whose control actions change only when events occur. Such an approach is computationally feasible for systems of interest in practice and avoids *mode-mismatch* errors. In order to handle modelling errors and exogenous disturbances, the proposed approach is implemented in a closed-loop strategy based on model predictive control.

**Keywords** Hybrid Petri nets · event-driven control · model predictive control

## 1 Introduction

Petri nets represent a widely spread formalism for modeling discrete event systems [29, 32]. Similarly to other formalisms for discrete systems, Petri nets suffer from the well-known state explosion problem, i.e., the number of states increases exponentially with respect to the size of the system.

An interesting approach to avoid the state explosion problem is to approximate the discrete variables that reach high values by continuous variables. Such variables typically correspond to raw parts, produced items, capacity of buffers, etc. On the other hand, other system variables, such as shared resources or processing machines,

J. Júlvez and C.Mahulea
Dpto. Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain.
E-mail: {julvez,cmahulea}@unizar.es

S. Di Cairano
Mitsubishi Electric Research Laboratories, Cambridge, MA.
E-mail: dicairano@ieee.org

A. Bemporad
Dip. Ingegneria Meccanica e Strutturale, Università di Trento, Italy.
E-mail: bemporad@ing.unitn.it

might maintain small values for any potential system evolution. Hence, they should be maintened as discrete. The above considerations lead to hybrid Petri nets [13], a modeling formalism in which the Petri net structure is the same as in a classical Petri net, but the amount of tokens in the subset of *continuous places* and the firings of the subset of *continuous transitions* are real numbers.

As in timed discrete Petri nets, several semantics can be associated to the firing of continuous transitions in timed hybrid Petri nets. In this paper, we will consider single-server semantics. Under this semantics the firing rate of a continuous transition remains constant as long as no place gets empty [4]. When a place gets empty, the firing rate changes and remains constant again until another place gets empty. in this way, the continuous-time evolution of the marking of a continuous Petri net is piecewise-linear.

The autonomous behavior of a hybrid Petri net can be modified by introducing control actions on the net transitions. By using a common analysis where a continuous transition is seen as a valve through which a liquid flows, the control action on the transition determines how much such valve is open. The introduction of control actions allows one to define control problems in the framework of hybrid Petri nets.

In this paper we propose a framework for optimization-based control of timed hybrid Petri nets, based on their piecewise linear dynamics. The task of solving control problems for continuous-time piecewise-linear systems is, in general, a challenging problem [23, 33]. A common approach to overcome this difficulty is to consider a discrete-time representation of the system [28]. However, time discretization leads to mode mismatch errors [15], mode changes that occur during the intersampling, and hence are lost or delayed in the discrete-time representation, leading to possibly large differences between the discrete-time and continuous-time trajectories. Clearly, the smaller the time step, the smaller the effects of the mode mismatch are. Unfortunately, in the case of finite horizon optimal control, reducing the sampling period usually increases the complexity of the problem to solve [8, 15].

The framework proposed in this paper is event-driven, and, by considering mode switches as included in the events, mode-mismatch is avoided. In such an approach, the control input is parametrized by a piecewise constant function where the time-duration of the different step is not assumed constant. As a consequence, the control signal is defined by tuples $(v(k), q(k))$ defining the integral of the control signal during the application period, and the application period, respectively. A tuple is produced when an event occurs, i.e., for a hybrid Petri net, when a places gets empty or when a discrete transition fires. Given that the marking evolution is piecewise-linear, the full system trajectory is defined by the sequence of tuples. Preliminary results of this method for optimal control of continuous Petri nets were proposed in [24]. This paper improves such preliminary results by considering hybrid Petri nets, and further extending the optimal control framework to a model predictive one, in order to provide a closed-loop strategy that corrects external disturances.

Model predictive control (MPC) [11, 25] is an optimization based receding horizon closed-loop control strategy, where at each control cycle a (constrained) finite horizon optimal control problem is solved, and only the first part of the computed optimal input profile is applied to the system. However, differently from open-loop optimal control, when fresh information on the system state becomes available, by measurements or estimators, the optimal sequence is recomputed. In this way, feedback is taken into account and MPC results to be a closed-loop control strategy.

Due to the improved performance achieved by using optimization algorithms, to the capability of handling multiple input, and to the possibility of enforcing constraints,

model predictive control has found several applications, for instance in process industry [30], automotive (e.g., [17,19]), aerospace (e.g., [21]), and supply chains (e.g., [10]). A previous application of model predictive control to a particular class of discrete-event systems is found in [14].

Several approaches to control continuous Petri nets exist in the literature. In [34] an algorithm to track control of Petri nets without joins is suggested. In [4] an optimal mode of operation for hybrid Petri nets is obtained by means of linear programming problems. Classical model predictive control has been applied to continuous Petri nets in [20,26]. However, classical model predictive control requires time discretization that may lead to mode mismatch errors when places of the net become empty during the intersampling. As such, the sampling period must be kept small enough to minimize the problems due to such errors, which however increases the complexity of the MPC controller. With respect to previous approaches, the present paper provides improvements in the class of models considered -hybrid Petri nets instead of continuous Petri nets- and on the controller properties, since the event-driven MPC does not require time-discretization nor oversampling to avoid mode mismatch problems.

The rest of the paper is organized as follows. Section 2 introduces hybrid Petri nets. A technique to express the behavior of hybrid Petri nets in an event-driven fashion is discussed in Section 3. Section 4 presents two methods for the event-driven control of continuous Petri nets. A finite horizon open-loop optimal control problem is introduced first, then used to implement a model predictive control strategy. Two control scenarios are shown in section 5. The conclusions are summarized in section 6.

### 1.1 Notation

$\mathbb{R}$, ($\mathbb{R}_{0+}$, $\mathbb{R}_{+}$) is the set of (nonnegative, positive) real numbers and $\mathbb{N}$ is the set of natural numbers. Inequalities between vectors are intended componentwise and when a number $c$ is used in the place of a vector, it indicates a vector where all the components have value $c$. For a time-dependent vector $x$, $x[i](k)$ denotes the value of component $i$ at step $k$, and $x(k)$ denotes the whole vector at step $k$. The step $(k)$ will be omitted if clear from the context. For a vector $\mu \in \mathbb{R}^n$, $\mu(h|\tau)$ is the $h$-steps ahead predicted value starting from time $\tau$. Since this paper discusses event-driven control, the steps start at the occurrence of events and time duration of the steps is not constant.

## 2 Hybrid Petri nets

In the following we assume the reader is familiar with the basic concepts of Petri nets (PNs), see [18,29] for an extensive overview. This section introduces the basic concepts related to *hybrid* Petri nets.

### 2.1 Untimed hybrid Petri nets

In contrast to conventional (i.e., discrete) PNs, the arc weights of hybrid PNs are real-valued.

**Definition 1 (HPN)** A *Hybrid Petri Net* (HPN) is a tuple $N = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ where:

- $P$ is a set of $|P|$ places, and $T$ is a set of $|T|$ transitions.
- $Pre : P \times T \to \mathbb{R}_{0+}$ and $Post : P \times T \to \mathbb{R}_{0+}$ are the *pre-* and *post-* incidence functions that specify the arc weights.
- $P = P_c \cup P_d$, $P_c \cap P_d = \emptyset$, and $T = T_c \cup T_d$, $T_c \cap T_d = \emptyset$

The set of places $P$ is partitioned into a set of *discrete places* $P_d$ and a set of *continuous places* $P_c$. Similarly, the set of transitions $T$ is partitioned into a set of *discrete transitions* $T_d$ and a set of *continuous transitions* $T_c$. Discrete places are represented as circles and continuous places as double circles, and similarly discrete transitions are represented as rectangles and continuous transitions as double rectangles, see for instance the network in Figure 2(a).

The main difference between HPNs and discrete PNs is in the way the transitions are fired. In discrete PNs the transitions are fired a natural number of times. In HPNs the discrete transitions are also fired a natural number of times, but the continuous transitions can be fired a real number of times which leads to real markings in continuous places.

In order to ensure the integrality of the marking of discrete places, two conditions are required: a) $Pre[p,t] \in \mathbb{N}$ and $Post[p,t] \in \mathbb{N}$ for every $p \in P_d$ and every $t \in T_d$; b) $Pre[p,t] = Post[p,t]$ for every $p \in P_d$ and every $t \in T_c$.

The incidence matrix of the net is $\mathbb{C} = Post - Pre$, $\mathbb{C} \in \mathbb{R}^{|P_c| \times |P_d|}$ and the state of the net is the marking $m \in {\mathbb{R}_{0+}}^{|P_c|} \times \mathbb{N}^{|P_d|}$, which evolves dynamically. The marking can be partitioned into its real and natural components, $m = [m_c'\ m_d']'$, $m_c \in {\mathbb{R}_{0+}}^{|P_c|}$, $m_d \in \mathbb{N}^{|P_d|}$, the marking of continuous places and discrete places, respectively. The *preset* and *postset* of a node $\xi \in P \cup T$ are denoted as ${}^\bullet\xi$ and $\xi^\bullet$.

**Definition 2 (HPN system)** A *Hybrid Petri Net System* is a tuple $\langle N, m_0 \rangle$ where:

- $N$ is a HPN.
- $m_0 : P \to \mathbb{R}_{0+}$ assigns to each place $p$, an initial marking $m_0[p]$. For every $p \in P_d$, it is required $m_0[p] \in \mathbb{N}$.

**Definition 3 (Enabling degree)** Let $\langle N, m_0 \rangle$ be a HPN system. The enabling degree of a transition $t \in T$ at marking $m$ is $enab(t,m) = \min_{p \in {}^\bullet t}\{m[p]/\mathbf{Pre}[p,t]\}$.

**Definition 4 (Firing)** Let $\langle N, m_0 \rangle$ be HPNs system. A transition $t \in T$ can be fired in any amount $\alpha$ such that $0 \leq \alpha \leq \text{enab}(t,m)$, and $\alpha \in \mathbb{N}$ if $t \in T_d$, $\alpha \in \mathbb{R}$ if $t \in T_c$. The firing of $t$ in a certain amount $\alpha$ leads to a new marking $m' = m + \alpha \cdot \mathbb{C}[P,t]$, where $\mathbb{C}[P,t]$ is the column of the incidence matrix corresponding to transition $t$.

Hence, as in discrete PN systems, the state (or fundamental) equation $m = m_0 + \mathbb{C} \cdot \sigma$ summarizes the marking evolution. Similarly to continuous Petri nets, in HPNs the marking of a continuous place can be seen as an amount of fluid being stored, and the firing of a continuous transition can be considered as a flow of this fluid going from a set of places (input places) to another set of places (output places).

### 2.2 Timed hybrid Petri nets

For the timing interpretation of continuous transitions a first order (or deterministic) approximation of the discrete case [31] is used, hence assuming that the delays associated to the firing of the transitions are approximated by their mean values. As a result,

the marking evolution with respect to time $\tau$ is

$$m(\tau) = m(0) + \mathbb{C} \cdot \sigma(\tau). \tag{1}$$

The instantaneous flow of a continuous transition $t \in T_c$ is defined as the derivative of its firing count vector with respect to time, i.e., $f = \dot{\sigma}$, and thus the evolution of a continuous place $p$ is described by

$$\dot{m}[p](\tau) = \sum_{t \in {}^\bullet p} Post[p,t] \cdot f[t](\tau) - \sum_{t \in p^\bullet} Pre[p,t] \cdot f[t](\tau) \tag{2}$$

Different semantics have been defined for the firing of continuous transitions, the most commonly used being *infinite servers* [31] and *finite servers* [1]. In this paper, finite servers semantics is considered. Under finite firing semantics, every continuous transition, $t \in T_c$, of the timed system is associated with a real parameter $\lambda[t] > 0$ that is the maximum flow allowed by $t$, i.e., $f[t] \leq \lambda[t]$.

As for continuous transitions, different time interpretations can be adopted for the firing of discrete transitions. Here, a deterministic firing of discrete transitions under finite server semantics is considered. Conflicts between discrete transitions are solved according to a given established scheduling policy, therefore no preemption can occur.

**Definition 5 (THPN system)** A *Timed Hybrid Petri Net System* (THPN system) is a tuple $\langle N, m_0, \lambda, \vartheta \rangle$ where:

- $\langle N, m_0 \rangle$ is a HPN system.
- $\lambda : T_c \to \mathbb{R}_+$ defines the maximum flow allowed by each continuous transition.
- $\vartheta : T_d \to \mathbb{N}$ defines the time delay of each discrete transition.

Intuitively, if a continuous transition is seen as a valve through which a fluid passes, $\lambda$ can be seen as the maximum flow admitted by the valve. In contrast to [4], we do not impose a lower bound for the flow of the transitions, thus, $0 \leq f[t] \leq \lambda[t]$.

In THPN two types of enabling for continuous transitions are considered [4].

**Definition 6 (Enabling of continuous transitions)** Let $\langle N, m_0, \lambda \rangle$ be a THPN system and $t \in T_c$. Let $m$ be a marking such that $m[p] \geq Pre[p,t]$ for every $p \in {}^\bullet t \cap P_d$.

- $t$ is *strongly enabled* at $m$ if $m[p] > 0$ for every $p \in {}^\bullet t \cap P_c$.
- $t$ is *weakly enabled* at $m$ if there exists $p \in {}^\bullet t \cap P_c$ such that $m[p] = 0$.

A continuous transition is not enabled if there exists $p \in {}^\bullet t \cap P_d$ such that $m[p] < Pre[p,t]$. Notice that in contrast to an untimed HPN, a continuous transition in a THPN having an empty continuous place is still weakly enabled and can fire. This happens when such an input place receives some input flow that is instantaneously consumed by the transition.

The flow of a transition depends on its enabling state:

**Definition 7 (Flow)** Let $\langle N, m_0, \lambda \rangle$ be a THPN system and $t \in T_c$.

- If $t$ is strongly enabled then it has maximum flow, i.e., $f[t] = \lambda[t]$.
- If $t$ is not enabled then it has no flow, i.e., $f[t] = 0$.
- The flow of the weakly enabled transitions must ensure that $m[p] \geq 0$, for all $p \in P_c$.

The computation of an admissible $f$ is non-trivial when several empty places appear. In [2], an iterative algorithm is suggested to compute one admissible $f$. In this paper, $f$ is computed similarly to [4] where the set of admissible $f$ is characterized by a set of linear inequalities. We choose $f$ to fulfil the linear inequalities and maximize $\sum_{t \in T} f[t]$ while assuming no priority on the transitions.

Due to the use of finite server semantics, the continuous transitions flow vector $f$ is piecewise constant, and, due to (2), the trajectory of the marking evolution of the continuous places is piecewise linear. Because of Definition 7, $f$ changes only when an event occurs, where for HPNs, an event occurs only when a place becomes empty, i.e., a transition becomes weakly enabled, or when a discrete transition fires.

Between two consecutive events, the system is said to be at an *invariant behavior state* (IB-state) [1]. Note that since the dynamics are constant in an IB-state, these play a similar role to the modes of a hybrid system [3]. Thus for a given fixed marking of the discrete places, the number of potential IB-states equals the number of sets of continuous places that can be empty. In principle, each place can be empty or not empty, hence the number of IB-states for a given fixed discrete place marking is bounded by $2^{|P_c|}$. However, the real number of IB-states is usually not so large, since initially marked P-semiflows ( [18, 29]) cannot be emptied.

*Example 1* Let us consider the system in Figure 1(a). The only input place of $t_1$ is marked, hence it is strongly enabled and $f[t_1] = \lambda[t_1] = 2$. The evolution of $m[p_1]$ is given by $\dot{m} = \lambda[t_2] - \lambda[t_1] = -1$. At time 1, $p_1$ becomes empty, i.e., an event occurs, and $t_1$ becomes weakly enabled. Now, the maximum flow admitted by $t_1$ is 1, since a greater flow would cause $m[p_1]$ to be negative. Being $f[t_1] = 1$, $p_1$ remains empty. Now $p_1$ can be seen as a tube instead of a deposit and no more events occur. For arbitrary values of $\lambda[t_1]$ and $\lambda[t_2]$, the flow of $t_1$ when $p_1$ is empty is defined as $f[t_1] = \min(\lambda[t_1], \lambda[t_2])$.

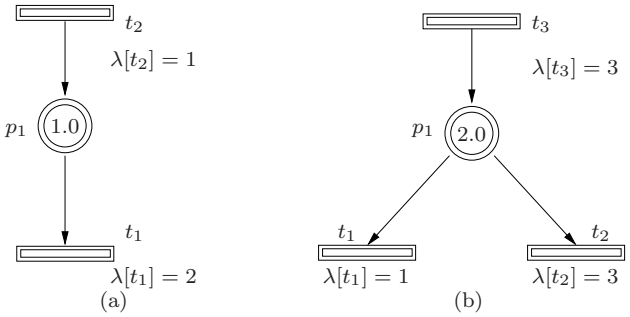



**Fig. 1** (a) Transition $t_1$ becomes weakly enabled at $\tau = 1$. (b) Transitions $t_1$ and $t_2$ become weakly enabled at $\tau = 2$.

*Example 2* Transitions $t_1$ and $t_2$ of the system in Figure 1(b) are strongly enabled for the given initial marking $m[p_1] = 2$. After two time units, $p_1$ becomes empty and $t_1$, $t_2$ become weakly enabled. At this point, it must be decided how to split the input flow, $\lambda[t_3]$, coming into $p_1$. Since we are considering that $t_1$ and $t_2$ have the same priority, half of the flow should be routed to $t_1$ and half to $t_2$. Unfortunately, the maximum

flow of $t_1$, $\lambda[t_1] = 1$, is smaller than $\lambda[t_3]/2$. To solve this situation, the flow that cannot be consumed by $t_1$, $\lambda[t_3]/2 - \lambda[t_1]$, is routed to $t_2$. This results in $f[t_1] = 1$ and $f[t_2] = 2$. The explicit analytic expressions for $f[t_1]$ and $f[t_2]$ with arbitrary $\lambda[t_1]$, $\lambda[t_2]$ and $\lambda[t_3]$ when $m[p_1] = 0$ are $f[t_1] = \min(\lambda[t_3]/2 + \max(0, \lambda[t_3]/2 - \lambda[t_2]), \lambda[t_1])$ and $f[t_2] = \min(\lambda[t_3]/2 + \max(0, \lambda[t_3]/2 - \lambda[t_1]), \lambda[t_2])$. The THPNs systems in Figure 1(a) and Figure 1(b) have two IB-states, one takes places when $m[p_1] > 0$ the other one when $m[p_1] = 0$.

### 2.3 Control actions

The hybrid Petri net models introduced so far are autonomous. Control actions can be introduced in THPNs in order to modify the autonomous evolution. In THPNs the controls act on the transitions.

A discrete transition $t \in T_d$ is *controllable* when its firing time is a decision variable for a controller. Obviously, the firing cannot happen before $\vartheta[t]$ time units have elapsed from the enabling of $t$, according to the common definition of discrete transition delay.

A continuous transition $t \in T_c$ is *controllable* when its flow can be reduced by a decision variable $u[t]$ such that

$$0 \leq u[t] \leq \lambda[t], \tag{3}$$

where $u \in \mathbb{R}^m$ is the vector of controls. An action $u[t]$ on the transition $t$ can be seen as if the hypothetical valve associated to $t$ was closed by the amount $u[t]$.

The flow $f$ is still computed as discussed in Section 2.2, where the maximum allowed flow by $t \in T_c$, is $\lambda[t] - u[t]$, i.e., $0 \leq f[t] \leq \lambda[t] - u[t]$. Thus, if $t \in T_c$ is strongly enabled, $f[t] = \lambda[t] - u[t]$, while if $t \in T_d$ is weakly enabled, $f[t] \leq \lambda[t] - u[t]$. Finally, if $t \in T_c$ is not enabled, $f[t] = 0$.

*Example 3* To show how input actions modify the evolution of a system, let us apply the input action $u[t_1] = 0.5$ to the system in Figure 1(a). Since, transition $t_1$ is initially strongly enabled, its flow will be $f[t_1] = \lambda[t_1] - u[t_1] = 1.5$. After two time units $p_1$ becomes empty. Hence, the maximum flow allowed by $t_1$ is the input flow coming to $p_1$, that is 1. Now, every input action on $t_1$ ranging from 0 to 1 has no effect on the system evolution. However, if $u[t_1]$ is greater that 1, the flow of $t_1$ will be slowed down. For example, if $u[t_1] = 1.5$, the flow of $t_1$ will be $f[t_1] = 0.5$, and consequently $p_1$ will start to fill. The analytic expression for $f[t_1]$ with arbitrary $\lambda[t_1]$ and $\lambda[t_2]$ when $p_1$ is empty is $f[t_1] = \min(\lambda[t_1] - u[t_1], \lambda[t_2] - u[t_2])$.

Similarly, for the system in Figure 1(b), if $m[p_1] > 0$ then $f[t_1] = \lambda[t_1] - u[t_1]$ and $f[t_2] = \lambda[t_2] - u[t_2]$. If $m[p_1] = 0$ then $f[t_1] = \min((\lambda[t_3] - u[t_3])/2 + \max(0, (\lambda[t_3] - u[t_3])/2 - (\lambda[t_2] - u[t_2])), \lambda[t_1] - u[t_1])$ and $f[t_2] = \min((\lambda[t_3] - u[t_3])/2 + \max(0, (\lambda[t_3] - u[t_3])/2 - (\lambda[t_1] - u[t_1])), \lambda[t_2] - u[t_2])$.

## 3 Event-driven representation

This section shows how THPNs can be expressed as a particular class of Mixed Logical Dynamical systems where each step represents the occurrence of an event. Section 3.1 introduces Mixed Logical Dynamical systems, and Section 3.2 shows how the mentioned transformation is performed.

### 3.1 Event-driven mixed logical dynamical systems

Mixed Logical Dynamical (MLD) systems [9] are computationally oriented representations of discrete-time hybrid systems. MLDs consist of a set of linear equalities and inequalities involving both real and Boolean ($\{0,1\}$) variables. An MLD system is described by the relations

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) + B_4 \tag{4a}$$
$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_4 \tag{4b}$$
$$E_1u(k) + E_5x(k) \leq E_2\delta(k) + E_3z(k) + E_4 \tag{4c}$$

where $x = [x_c\ x_d] \in \mathbb{R}^{n_r} \times \{0,1\}^{n_b}$ is a vector of continuous and binary states, $u = [u_c\ u_d] \in \mathbb{R}^{m_r} \times \{0,1\}^{m_b}$ are the inputs, $y = [y_c\ y_d] \in \mathbb{R}^{p_r} \times \{0,1\}^{p_b}$ are the outputs, $\delta \in \{0,1\}^{r_b}$, $z \in \mathbb{R}^{r_r}$ represent auxiliary binary and continuous variables, respectively, and $A$, $C$, $B_i$, $D_i$, $i = 1,\ldots,4$, $E_i$, $i = 1,\ldots,5$ are matrices of suitable dimensions. Given the current state $x(k)$ and input $u(k)$, the evolution of (4a)-(4c) is determined by solving (4c) for $\delta(k)$ and $z(k)$, then updating $x(k+1)$ and $y(k)$ from (4a) and (4b). It is assumed that the system (4a)-(4c) is *well-posed* [9], which means that for any value of $x(k)$, $u(k)$ within the range of interest, $\delta(k)$, $z(k)$ are uniquely defined by (4c).

In the recent work [16] the authors have proposed an event-driven MLD model,

$$\chi(k+1) = \chi(k) + B_1\mu(k) + B_2\delta(k) + B_3z(k) + B_4 \tag{5a}$$
$$y(k) = C\chi(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_4 \tag{5b}$$
$$E_1\mu(k) + E_5x(k) \leq E_2\delta(k) + E_3z(k) + E_4 \tag{5c}$$

where $\chi(k) = [x(k)'\ \tau(k)]$, $q \in \mathbb{R}_{0+}$, $\mu(k) = [v(k)'\ u_d(k)'\ q(k)]'$. In (5), the counter $k$ represents the number of events, the additional state variable $\tau(k)$ is the total time elapsed when the $k^{th}$ event occurs, $q(k)$ is the time between the $k^{th}$ and the $(k+1)^{th}$ events, and $v(k)$ is the integral of the continuous control input between the $k^{th}$ and the $(k+1)^{th}$ events, where it is assumed piecewise constant, i.e., $v(k) = q(k)u_c(t(k)^+)$. As discussed in [16], in the eMLD system an event occurs either when the value of $\delta$ changes, due to (5c), or when the input $\mu$ is changed.

In what follows we show how to transform THPNs in eMLD form.

### 3.2 Transforming THPNs to event-driven MLDs

Consider again the THPN in Figure 1(a). According to the defined semantics, the continuous-time marking evolution is described by

$$\begin{aligned} \text{if} \quad m[p_1] > 0 \quad &\text{then} \quad \dot{m}[p_1] = -1 \\ &\text{else} \quad \dot{m}[p_1] = 0\ . \end{aligned} \tag{6}$$

Clearly, if the initial marking of $p_1$ is $m_0[p_1] = 1$, after 1 time unit $p_1$ gets empty. Such an evolution can be described appropriately by a discrete-time model only if the duration of the sampling period $h \in \mathbb{R}$ satisfies $h \cdot k = 1$, for some $k \in \mathbb{N}$. If this is not the case, the marking of $p_1$ will become at some point negative, and the evolution will block. This phenomenon is named mode-mismatch error, where the exact instant

of the mode switch is lost, because it occurs in the intersampling. Mode-mismatch is present in most discrete-time models of hybrid systems [8], and it can be alleviated by imposing a very small sampling period $h$ (oversampling), which however results in unnecessary computations in the control algorithm.

Indeed, mode mismatch is not present if a continuous time of the system is used. However, for continuous time models, the input is an infinite-dimensional decision variables, hence computational tools which require finite dimensional decision variable, such as mathematical programming, cannot be used. To overcome the mode-mismatch while retaining finite dimensionality of the input, an event-driven approach, instead of a discrete-time one, shall be used. In an event-driven the system evolves at events, and the time separation of the events is not constant, but modelled as a variable, e.g., $q(k)$ in (5). By including mode switches in the set of events, the mode-mismatch error is removed.

For the marking evolution of the system in Figure 1(a) we obtain

$$\begin{array}{lll} \text{if} \quad m[p_1] > 0 & \text{then} & m[p_1](k+1) = m[p_1](k) - 1 \cdot q(k) \\ & \text{else} & m[p_1](k+1) = 0 \end{array} \tag{7}$$

In order to guarantee that $q(k)$ does not drive the system to negative markings, the constraint $m[p_1] \geq 0$ must be considered. Such a constraint together with the conditional statement (7) can be easily included in an eMLD system (4), as it will be shown later.

Next, the controls on $t \in T_c$ have to be included. Let us consider again the system in Figure 1(a) with $m_0[p_1] = 1$. Suppose that the continuous transition $t_1$ is controllable, so that the flow of $t_1$ is $f[t_1] = \lambda[t_1] - u[t_1](k)$ and the marking evolution of $p_1$ from event $k$ to $k+1$ is

$$m[p_1](k+1) = m[p_1](k) + q(k) \cdot (f[t_2] - f[t_1]) = m[p_1](k) + q(k) \cdot (\lambda[t_2] - (\lambda[t_1] - u[t_1](k))).$$

Thus, the equation defining $m(k+1)$ is nonlinear. As in eMLD system (5), the control action is re-parametrized by considering the integral between two events

$$v[t_1](k) = q(k) \cdot u[t_1](k), \tag{8}$$

which results in the linear dynamics

$$m[p_1](k+1) = m[p_1](k) + q(k) \cdot \lambda[t_2] - q(k) \cdot \lambda[t_1] + v[t_1](k).$$

The constraints on the control action (3) are also translated to the new parametrization of the control action by $0 \leq v[t_1] \leq q(k) \cdot \lambda[t_1]$ must be added.

The input action on the discrete transitions $t \in T_d$ is expressed as a boolean variable, $u[t]$, which can be straightforwardly included in (4a). If $u[t](k) = 1$, then transition $t$ fires at step $k$, it does not fire otherwise. The constraints in (4b) are used to avoid $u[t](k)$ from being equal to 1 before $\vartheta[t]$ time units have elapsed from the enabling of $t$.

In a THPN the flow of the transitions is constant between events and depends only on the IB-state of the net. By treating each step $k$ as the occurrence of an event in the THPN rather than the elapse of a sampling period, we will transform the THPN into an event-based mixed logical dynamical (eMLD) system. In this way we obtain two major advantages. First, a discrete-event representation where the events include the mode switches avoids the mode-mismatch errors typical of the discrete-time system, while

maintaining the capability of finite parametrization of the control signals. Second, the number of control actions is not increased, since any event causes the request of only one control action. As a consequence, the length of the sampling period is not constant, but depends on the time separation between the events.

The steps to convert a THPN into the aforementioned eMLD system are the following:

1. *Identify the potential IB-states of the THPN*, i.e., the possible dynamics that define the evolution of the system. For example, the system in Figure 1(a) has two potential IB-states: a) $m[p_1] > 0$ and b) $m[p_1] = 0$.
2. *Define the flow of continuous transitions under each IB-state.* The flow of $t_1$ in Figure 1(b) is $f[t_1] = \lambda[t_1]$ if $m[p_1] > 0$, and $f[t_1] = \min(\lambda[t_1], \lambda[t_2])$ if $m[p_1] = 0$.
3. *Define the evolution of the marking,* by including the state equation of the THPN in the eMLD equations. For instance, for the system in Figure 1(b), the equation defining the evolution of the marking is $m(k+1) = m(k) + q \cdot (f[t_3] - f[t_1] - f[t_2])$, where $q$ is the real variable storing the time elapsed between events $k$ and $k+1$.
4. *Specify the remaining time to fire of enabled discrete transitions.* If a discrete transition $t$ becomes enabled at a given IB-state its time to fire is set to $\vartheta[t]$. If $t$ was already enabled in the previous IB-state, its time to fire is decreased in $q$ time units, where $q$ is the time duration of the previous IB-state.
5. *Constrain the system variables to nonnegative numbers.* In order to achieve this, inequalities on the marking of places, e.g., $m[p_1](k) \geq 0$, and on the time to fire of the discrete transitions are added. An event occurs when one of these constraints is activated, modeling a place to become empty or a discrete transition to be fired.

By Step 1 a set of modes $\mathcal{J}_c$ is computed, each corresponding to an IB-state and identified by a set of linear inequalities on the marking of places

$$H_j^m m(k) \leq h_j^m. \tag{9}$$

The inequalities in (9) represents the IB-state conditions on the marking, such as $m[p_2](k) = 0$, or[1] $m[p_1](k) > 0$.

For each $j \in \mathcal{J}_c$ the vector $f_j$ defining the net flow in the corresponding IB-state is computed (Step 2). At Step 3 we compute the marking update dynamics by considering controllable transitions and integrating between two events, hence obtaining

$$m(k+1) = m(k) + B_j v(k) + f_j q(k), \tag{10}$$

where $B_j$ represents the effects of the controls when in IB-state $j \in \mathcal{J}$. The commands are subject to constraints of the type $0 \leq v[t](q) \leq \lambda[t]q(k)$.

Step 4 concerns the modelling of the discrete transitions $t \in T_d$. In order to account for the transitions delay, a vector of timers $\psi$ is added. For a given IB-state $i \in \mathcal{J}_c$, the timer dynamics are defined by

$$\psi(k+1) = \psi(k) + \gamma_{ij} q(k) + \vartheta_{ij} \tag{11a}$$

$$j \in \mathcal{J}_\psi^i \ : \ H_j^\psi \psi(k) \leq h_j^\psi, \tag{11b}$$

where (11b) selects the timer dynamics depending on the current timer value (e.g., $\psi[h](k) > 0$, $\psi[h](k) = 0$, etc.), and (11a) represents the different timer evolutions such as $\psi[h](k+1) = \vartheta[h]$ for a timer that has not started the countdown,

[1] An arbitrarily small numerical constant, e.g., the machine precision, is used to convert strict inequalities into non-strict ones.

$\psi[h](k+1) = \psi[h](k) - q(k)$, for a timer that is counting down, $\psi[h](k+1) = \psi[h](k)$, for a timer that maintains the current value, for instance because it has reached zero.

The discrete transition can be fired when enabled, which depends on the current IB-state and on the timer state. Thus, the firing of discrete transition $j$, occurs by setting to 1 a corresponding Boolean variable $u_d[j] \in \{0, 1\}$, when the conditions on the timer and on the IB-state are met. This results in a set of modes $\mathcal{J}_d$ with evolution defined by

$$m(k+1) = m(k) + \phi_j,\ j \in \mathcal{J}_d \tag{12a}$$
$$\tau(k+1) = \tau(k) \tag{12b}$$
$$u_d[j] \geq 0.5 \tag{12c}$$
$$H_j^{d,m} m(k) \leq h_d^m \tag{12d}$$
$$H_j^{d,\psi} \psi(k) \leq h_d^\psi \tag{12e}$$

where $f_j$, $j \in \mathcal{J}_d$ represents the effect of the transition, (12b) shows the transition firing is instantaneous, (12c) is added to command the transition firing, and (12d), (12e) represents the conditions that have to be met for a discrete transitions to fire, in terms of marking and timers, respectively.

By collecting (9)–(12) we have that the complete set of modes is $\mathcal{J} = \mathcal{J}_d \cup \left(\bigcup_{i \in \mathcal{J}_c} \mathcal{J}_\psi^i\right)$, and the dynamics of the net are formulated as

$$m(k+1) = m(k) + B_j^{pn} v(k) + f_j^{pn} q(k) + \phi_j^{pn} \tag{13a}$$
$$\tau(k+1) = \tau(k) + \Gamma_j^{pn} q(k) \tag{13b}$$
$$\psi(k+1) = \psi(k) + \gamma_j^{pn} q(k) + \vartheta_j^{pn} \tag{13c}$$
$$m(k) \geq 0,\ q(k) \geq 0 \tag{13d}$$
$$j \in \mathcal{J}\ :\ H_j^{pn,m} m(k) + H_j^{pn,\psi} \psi_j(k) \leq h_j^{pn}, \tag{13e}$$
$$u_d(k) \geq \rho_j, \tag{13f}$$
$$0 \leq v(k) \leq \Lambda_j q(k) \tag{13g}$$
$$\Omega_j^m m(k) + \Omega_j^\psi \psi(k) + \Omega_j^v v(k) + \Omega_j^q q(k) \leq \ell_j \tag{13h}$$

where $\gamma_j^{pn}, B_j^{pn}, f_j^{pn}, \phi_j^{pn}, \Gamma_j^{pn}, \vartheta_j^{pn}$ are obtained from (10), (11a), (12a). Note that $\gamma_j^{pn}, \Gamma_j^{pn}, B_j^{pn}, f_j^{pn} = 0$, for $j \in \mathcal{J}_d$, since the discrete transition firings are instantaneous, and $\phi_j = 0$, for $j \notin \mathcal{J}_d$, since the marking evolution is continuous except for the discrete transitions.

Inequalities (13e) are obtained from the marking conditions (9) that select the current IB-state, from conditions (12d) that indicate whether a discrete transition can occur at the current IB-state, from timer conditions (11b), that select how the timer evolves, and from conditions (12e) that check whether a timed discrete transition can occur. Inequalities (13f) fire a discrete transition, where obviously the firing is disabled by setting the components of $\rho_j$ to be larger than 1. The additional set of inequalities (13h) enforces the conditions (13e)–(13g) to hold for $m(k+1)$, $\psi(k+1)$ within an arbitrarily small bound, so that mode switches do not occur during $(t(k), t(k+1))$ but only at $t(k+1)$ (see [16] for details).

Equation (13) define the net dynamics as a piecewise affine system that is easily converted into eMLD form (5) as discussed in [15] by algorithm [6], whose implementation is available in the toolbox [5]. The result is an eMLD (5) whose state vector

$\chi' = [m'\ \psi'\ \tau']$ contains the net marking, the timers values, and the current time, and whose input vector $\mu' = [u'\ u'_d\ q']$ contains the integral of the continuous input to be applied, the discrete transition firing commands, and the duration of the application. By using the language HYSDEL in [5], the net dynamics can be directly obtained from the high-level description in Steps (1)–(5), while the matrices of (13) and of the corresponding eMLD (5) are automatically generated.

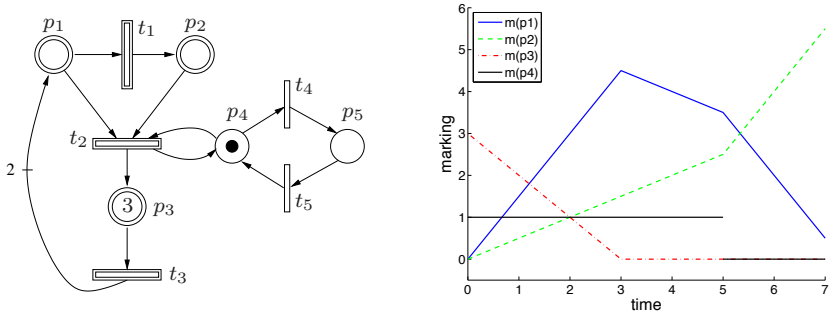
**Proposition 1** *Let* (13) *be used to model the THPN and let* $\tau(k)$, $\tau(k+1)$, $k \in \mathbb{N}$, *be the time instants of two consecutive events. Then, neither IB-state changes nor changes in the ready-to-fire discrete transitions occur during* $(t(k), t(k+1))$.

**Proof:** For given $\bar{\jmath} \in \mathcal{J}$, inequalities (13e)–(13g) identify the the active IB-state and ready-to-fire transitions. By the results of [16], constraint (13h) enforces that inequalities (13e)–(13g) are satisfied at $\tau(k)$ and at $\tau(k+1)-\sigma$, for $\sigma > 0$ arbitrarily small. Thus, the IB-state and the ready-to-fire transitions are the same at $\tau(k)$ and at $\tau(k+1)-\sigma$. Inequalities (13e)–(13g) are linear, hence they describe a polyhedral set, and the state trajectory $\chi(\tau)$ is linear between two events. Given that the extrema of the segment $\chi(\tau)$, $\tau \in [\tau(k), \tau(k+1)-\sigma]$ belongs to the polyhedral set, the whole segment belongs to the set. Hence constraints (13e)–(13g) which describe the active IB-state and ready-to-fire transitions are the satisfied for all $\tau \in [\tau(k), \tau(k+1)-\sigma]$. □

The result of Proposition 1 ensures mode and constraint enforcement continuously-in-time. This is stronger than classical results for discrete-time systems, where constraints and mode enforcement are guaranteed only pointwise-in-time. Proposition 1 ensures the mode-mismatch error is avoided.

*Example 4* Consider the system in Figure 2(a) with $\lambda[t_1] = (1.5)$, $\lambda[t_2] = (1)$, $\lambda[t_3] = (2)$, $\vartheta[t_4] = \vartheta[t_5] = 5$, and $m_0 = (0\ 0\ 3\ 1\ 0)$. The system is transformed into eMLD form following the tasks described above. At time 0 the flow of the continuous transitions is $f[t_1] = 1.5$, $f[t_2] = 1$ and $f[t_3] = 2$. At time instant 3 place $p_3$ becomes empty, i.e., the length of the first interval is $q(1) = 3$ time units. At that instant $t_3$ becomes weakly enabled and its new flow is $f[t_3] = 1$, the other two continuous flows keep the same.

At time instant 5, transition $t_4$ fires. This second event disables transition $t_2$ causing the continuous flows be $f[t_1] = 1.5$, $f[t_2] = 0$ and $f[t_3] = 0$. The time trajectory of the system for the first 7 time units is depicted in Figure 2(b).




(a) (b)

**Fig. 2** (a) A THPN system. (b) Its marking evolution.

## 4 Event-driven control of THPNs

In this section, we show how optimization based control can be applied on THPNs via their eMLDs formulation, and how feedback can be accounted for by a model predictive control strategy. We first propose a set of cost functions and constraints that can be used to formulate open-loop finite horizon optimal control problems for the THPN that can be solved by standard Mixed Integer Linear Programming (MILP) algorithms, then we show how the optimal control problem can be used as the base for a receding horizon control strategy, hence implementing an event-driven model predictive control algorithm for the THPN.

### 4.1 Event-driven optimal control

The advantages of formulating the THPN as an eMLD system (5) is that the dynamics are expressed by mixed-integer equalities and inequalities. As a consequence, the dynamics equations can be used into the mixed integer optimization problem

$$\min_{\bar{\mu}(t)} \quad J(\bar{\mu}(t), \bar{\chi}(t)) \tag{14a}$$

$$\text{s.t.} \quad \chi(k+1|t) = A\chi(k|\tau) + B_1 v(k) + B_2 \delta(k|\tau) + B_3 z(k|\tau) + B_5 \tag{14b}$$

$$E_2\delta(k|\tau) + E_3 z(k|\tau) \leq E_1\mu(k|\tau) + E_4\chi(k|\tau) + E_5 \tag{14c}$$

$$H_2\delta(k|\tau) + H_3 z(k|\tau) \leq H_1\mu(k|\tau) + H_4\chi(k|\tau) + H_5 \tag{14d}$$

$$\chi(0|t) = \chi(t), \quad k = 1, \ldots, N \tag{14e}$$

where $J$ in (14a) is the cost function, $\bar{\mu}(t) = \{\mu(k|\tau)\}_{k=0}^{N}$ are the decision variables, $\bar{\chi} = \{\bar{\chi}(k|\tau)\}_{k=0}^{N}$ is the eMLD state trajectory, (14b) and (14c) define the dynamics, (14d) includes additional constraints, and (14e) defines the initial state used for prediction. Note that once the initial state is fixed by (14e) the trajectory $\bar{\chi}(t)$ and the value of the auxiliary variables is assigned because of the wellposedness of the eMLD.

The decision variables $\mu(t)$ contains the continuous command integral, the corresponding durations, and the discrete commands. From the first ones and the second ones, the flow commands can be easily obtained as $u(h|t) = v(h|t)/q(h|t)$, with application interval $(\tau(h|t), \tau(h|t) + q(h|t))$.

Since the constraints in (14) are linear relations between integer and real variables, by choosing the cost function (14a) to be linear, the resulting problem is a mixed integer linear programming (MILP) problem, which can be solved by using exisiting reliable solvers [12, 22, 27]. Figure 3 sketches the steps that have been followed to obtain an MILP problem from the initial THPN. If (14a) was chosen to be a quadratic function, the resulting problem would be a mixed-integer quadratic problem, which is still solvable, even though computationally more complex, see [16].

The cost function (14a) and the additional constraints (14d) are used to define the objectives of the optimization problem, as shown next.

#### *4.1.1 Final target marking*

In order to enforce the marking to reach a desired target marking $\hat{m}$ after $N$ events, a terminal constraint can be added

$$m(N) = \hat{m}. \tag{15}$$

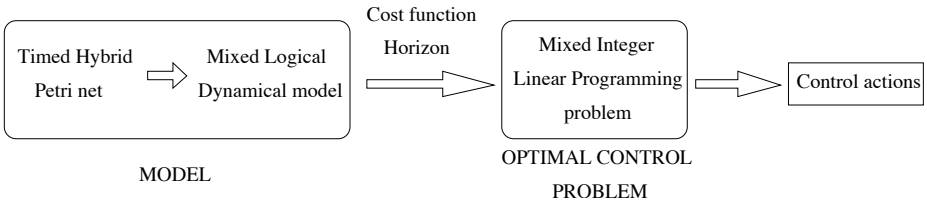



**Fig. 3** Obtaining an MILP problem from a THPN.

The terminal constraint (15) can be softened to preserve feasibility of optimization problem (14), hence adding to $J$ in (14a) the term

$$F\Big(m(N),\tau(N)\Big) = \rho\,\|m(N)-\hat{m}\|_\infty, \tag{16}$$

where $\rho$ is a large weight. A more general case is to consider a desired marking range, for instance a polyhedral set expressed by the constraints $\mathcal{M}_N m(n) \le M_N$, $\mathcal{M}_N \in \mathbb{R}^{q\times|P|}$, $M_N \in \mathbb{R}^q$.

*4.1.2 Cost function*

For THPN control, the general form of the cost function is

$$J(\bar{\chi}(t),\bar{\mu}(t)) = F\Big(m(N|t),\tau(N|t)\Big) + \sum_{k=0}^{N-1} L\Big(m(k|\tau),\tau(k|\tau),\mu(k|\tau)\Big). \tag{17}$$

hence composed of a terminal cost $F$ and a stage cost $L$, usually in the form

$$L(m,\tau,\mu) \triangleq \|m-\hat{m}\|_p^{Q_1} + \|\tau-\hat{\tau}\|_p^{Q_2} + \|v-\hat{v}\|_p^{R_1} + \|q-\hat{q}\|_p^{R_2} \tag{18a}$$

$$F(m,\tau) \triangleq \|m-\hat{m}\|_p^{Q_N} + \|\tau-\hat{\tau}\|_p^{Q_\tau},\ p\in\{1,\infty\}. \tag{18b}$$

where "$\hat{}$" denotes a given reference for the corresponding vector. The following subsections show some of the possible control goals in an event-driven framework. The use of $1,\infty$-norms allows to formulate (17) as a linear function, through auxiliary variables and linear constraints [7].

A case of particular interest is *minimum-time* control, where the minimum time to reach a certain marking is sought. Thus, together with terminal constraint (15) the stage cost and terminal cost are respectively set to

$$L(m(k),\tau(k),\mu(k)) = \sum_{k=1}^{N} q(k),\ \ F\Big(m(N),\tau(N)\Big) = 0. \tag{19}$$

A different criterion to reach the desired marking $\hat{m}$ is *minimum-effort*, which minimizes the intensity of the command input $u(\tau)$, hence letting the THPNs evolve as close as possible to its autonomous behavior. By using the $\ell_1$-norm of the input signal, we obtain $J(m,\tau,q,v) = \int_0^{\tau_N} \|u(\tau)\|\mathrm{d}t = \sum_{k=0}^{N-1}\int_{\tau(k)}^{\tau(k+1)} \|u(\tau)\|_1\mathrm{d}\tau$. Since $u$ is constant in each period $[\tau(k),\tau(k+1))$,

$$L\Big(m(k|\tau),\tau(k|\tau),\mu(k|\tau)\Big) = \|v(k|\tau)\|_1,\ \ F\Big(m(N|t),\tau(N|t)\Big) = 0. \tag{20}$$

A slightly different cost function from (17) can be used to represent the *minimum-displacement* criterion. This criterion looks for the trajectory that minimizes the largest deviation from a desired continuous state trajectory $\hat{m}(\cdot)$, that we assume piecewise linear and continuous (a special case is $\hat{m}(\cdot) \equiv \hat{m}$)

$$J(\bar{\chi}(t), \bar{\mu}(t)) = \max_{\tau \in [\tau(0), \tau(N)]} \|m(\tau) - \hat{m}(\tau)\|_\infty. \tag{21}$$

**Proposition 2** *Let $m_c(\tau)$, $\forall \tau \in [\tau_0, \tau_N]$, be the trajectory of continuous states of a THPN system, $\tau_0 < \tau_1 < \ldots < \tau_N$ be the event instants, assume that $\hat{m}(\tau)$ is linear over each $[\tau_i, \tau_{i+1})$, $i = 0, \ldots, N-1$ and continuous over $[\tau_0, \tau_N]$. Then*

$$\max_{\tau \in [\tau_0, \tau_N]} \|m(\tau) - \hat{m}(\tau)\|_\infty = \max_{k=0,\ldots,\tau_N} \{\|m(k|\tau) - \hat{m}(k|\tau)\|_\infty\}. \tag{22}$$

**Proof:** The marking trajectories of a THPN are continuous, so $\|m(\cdot) - \hat{m}(\cdot)\|_\infty$ is continuous, being the composition of continuous functions ($\|\cdot\|_\infty$, $m(\cdot)$, $\hat{m}(\cdot)$), and therefore the maximum over $[t_0, t_N]$ is well defined. Moreover, function $\|m(\cdot) - \hat{m}(\cdot)\|_\infty$ is a convex function of time $\tau$ on $[\tau(k), \tau(k+1)]$, being the composition of a convex function (the infinity norm) with linear functions (the state trajectory of the THPN and $\hat{m}$ between two consecutive switches), and thus it attains its maximum either at $\tau(k)$ or at $\tau(k+1)$. Hence,

$$\begin{aligned}
&\max_{\tau \in [\tau(0), \tau(N)]} \|m(\tau) - \hat{m}(\tau)\|_\infty \\
&= \max_{0 \le k \le N-1} \left\{ \max_{\tau \in [\tau(k), \tau(k+1)]} \|m(t) - \hat{m}(t)\|_\infty \right\} \\
&= \max_{0 \le k \le N-1} \{\max\{\|m(\tau(k)) - \hat{m}(\tau(k))\|_\infty, \|m(\tau(k+1)) - \hat{m}(\tau(k+1))\|_\infty\}\} \\
&= \max_{0 \le k \le N} \{\|m(\tau(k)) - \hat{m}(\tau(k))\|_\infty\} = \max_{k=0,\ldots,\tau_N} \{\|m(k|\tau) - \hat{m}(k|\tau)\|_\infty\}
\end{aligned}$$

□

Note that cost function (22) still leads to a mixed-integer linear formulation of problem (14).

### 4.2 Event-driven model predictive control

Problem (14) is a finite horizon open-loop optimal control problem, which computes the control profile $u(r)$, $r \in [\tau, \tau + \tau(N|\tau)]$, such that the constraints are satisfied and the cost is minimized. However, the control profile proceeds only for a finite number of events, where more events can be considered only at the price of an increased computational burden for solving (14). Furthermore, disturbances that occur during the execution of the control profile and possible modelling errors are not accounted for. Thus, a receding horizon feedback strategy is more advisable for cases where disturbances are possible. For this reason we incorporate the optimal control problem (14) in an event-driven closed-loop strategy based on Model Predictive Control (MPC) [11,25].

The event-driven Model Predictive Control (eMPC) strategy is defined as follows:

1. Let $N$ be the event horizon; at a generic time $\tau$ set $\chi(\tau) = [m(\tau)' \ \psi(\tau)' \ \tau']'$.
2. Solve problem (14), to obtain the sequence of optimal controls $\mu^*(\chi(\tau)) = [\mu^*(0|\tau), \ldots, \mu^*(N-1|\tau)]$.

3. Compute the input levels profile $\bar{u}_c^*(\chi(\tau)) = [u_c^*(0|\tau), \ldots, u_c^*(N-1|\tau)] = \left[\frac{v^*(0|\tau)}{q^*(0|\tau)}, \ldots, \frac{v^*(N-1|\tau)}{q^*(N-1|\tau)}\right]$,
4. Apply $u(r) = [u_c^*(0|\tau)'\ u_d^*(0|\tau)']'$ for $r \in [\tau, \tau + q^*(0|\tau)]$.
5. Set $\tau = \tau + q^*(0|\tau)$, measure the new value of $\chi(\tau)$ and go to Step 2.

The actual state $m(\tau + q(0|\tau))$ at the end of each control action may be different from the predicted one $m(1|\tau)$ because of external disturbances and modelling errors. In fact, also the time instant at which the optimization problem is repeated may be different from the scheduled instant $\tau + q(0|\tau)$. By the closed-loop nature of the eMPC approach, the current state (and time) are measured or estimated again and a new updated optimal input sequence is computed.

For the nominal case, i.e., the trajectory is not purturbed by external disturbances, the reachability of a desired target marking can be proven.

**Proposition 3** *Consider the event-driven MPC scheme applied to a THPN where the cost function is the minimum-time criterion* (19)*, and where the terminal constraint* (15) *is applied on the desired target marking $\hat{m}$. If the problem is feasible at time $\tau_0$ with finite cost, then it is recursively feasible and the desired marking is reached in finite time, i.e., $m(\tau) = \hat{m}$ for $\tau < \infty$.*

**Proof:** The result follows from the convergence of eMLD scheme proved in [16]. Since at time $\tau_0$ problem (14) is feasible with finite cost, due to terminal constraint and minimum-time criterion, the command sequence $\mu^*(\tau_0)$ brings the marking to the target in finite time $J^*(\tau_0)$. A time $\tau_1 = \tau_0 + q^*(0|\tau_0)$, a new optimization problem is solved, where the sequence $[\mu^*(1|\tau_0), \ldots, \mu(N|\tau_0), \mu(N|\tau_1)]$ where $\mu(N|\tau_1) = [0\ 0\ 0]'$ is feasible, and brings the marking to the target in time $J(\tau_1) = J^*(\tau_0) - q^*(0|\tau_0)$. Hence, $J^*(\tau_1) \leq J^*(\tau_0) - q^*(0|\tau_0)$, and by recursive application, $J^*(\tau_k) + \sum_{i=0}^{k-1} q^*(0|\tau_0) \leq J^*(\tau_0)$, which means that the time computed at the first step it is always a lower bound for the time to reach the marking. Since $J^*(\tau_0) < \infty$ the time to reach the marking is finite. □

Similar reachability results can be proved for the other type of criteria, where however the target may be reached only asymptotically in time, because convergence time is not explicitly accounted for in the cost function.

## 5 Control scenarios

This section presents two manufacturing systems modeled with hybrid Petri nets to which an event-driven MPC approach has been applied. The first system is a multiclass machine, the second one is a production network.

### 5.1 Multiclass machine

The hybrid Petri net in Figure 4 models a production system consisting of two lines and a single machine that processes the items in both lines. The first (second) line is modeled by transitions $t_1$, $t_2$ ($t_3$, $t_4$), and places $p_1$, $p_3$ ($p_2$, $p_4$), and has a capacity of $c_1$ ($c_2$) items. The input flows of the first and second lines are given by the flows of $t_1$ and $t_3$, respectively. Places $p_1$ and $p_2$ are the buffers to store the incoming parts

from $t_1$ and $t_3$ before being processed. The output flow of the first (second) class is represented by $t_2$ ($t_4$). The processing machine is modeled by transitions $t_5$, $t_6$, $t_7$, $t_8$ and places $p_5$, $p_6$, $p_7$, $p_8$. Since the number of items in the lines is expected to be high, the places and transitions for the lines are continuous. On the other hand, since only one machine is available, the subnet that models the machine is discrete.

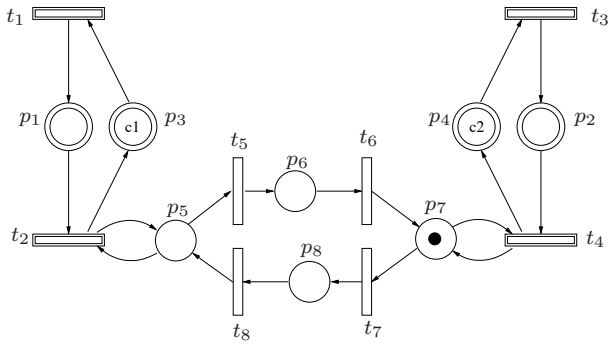



**Fig. 4** Two production lines and a multiclass machine.

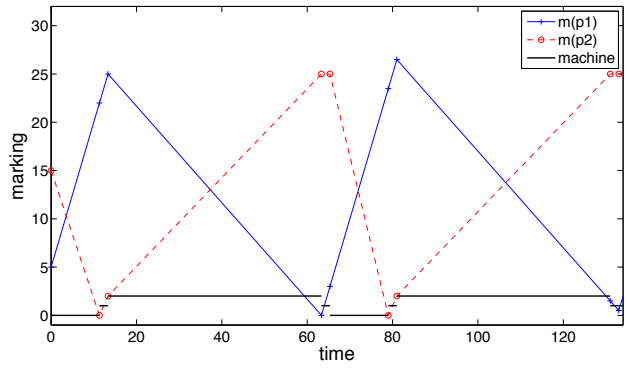
Let the capacity of the buffers be $c_1 = 30$, $c_2 = 25$, and $\lambda[t_1] = 1.5$, $\lambda[t_2] = 2$, $\lambda[t_3] = 1$, $\lambda[t_4] = 3$. It is assumed that the processing machine needs 2 time units to change from one line to the other. During such 2 time units, none of the lines is processed. This is modeled by a deterministic delay of 2 units in transitions $t_6$ and $t_8$ , i.e., $\vartheta[t_6] = \vartheta[t_8] = 2$, and by immediate transitions $t_5$ and $t_7$, i.e., $\vartheta[t_5] = \vartheta[t_7] = 0$. Let the initial marking of the system be $m_0[p_1] = 5$, $m_0[p_2] = 15$ and $m_0[p_7] = 1$. The marking of the remaining places is uniquely defined by these, since the invariants $m[p_1] + m[p_3] = c_1$, $m[p_2] + m[p_4] = c_2$, $m[p_5] + m[p_6] + m[p_7] + m[p_8] = 1$), hold.

Assume that it is required to compute a control law that maximizes the number of items produced over a given time interval. This is equivalent to maximizing the sum of the integral flows, $v$, of transitions $t_2$ and $t_4$. Hence, the cost function associated to (17) is

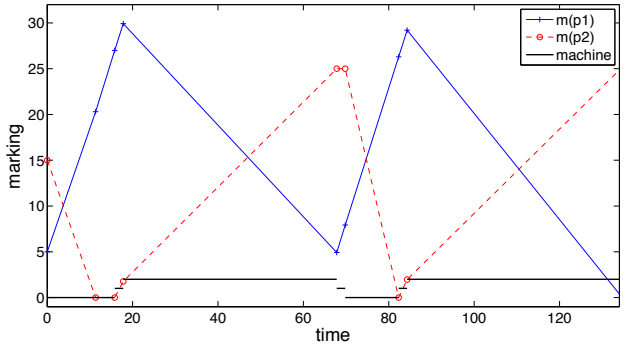
$$F(m(N|\tau), \tau(N|\tau)) = 0, \quad L\Big(m(k|\tau), t(k|\tau), \mu(k|\tau)\Big) = -\sum_{k=1}^{N-1} (v[t_2](k|\tau) + v[t_4](k|\tau)) \tag{23}$$

The control actions for this control problem are obtained by applying the MPC approach with prediction horizon $N = 8$ steps, and imposing a maximum time $\tau(N|\tau_0) \leq 200$, added to (14d). The time evolution of the system is shown in Figure 5(a). The state of the machine is shown by the line associated to *machine*: when the value is 2, the machine is processing line 1, i.e., $m[p_5] = 1$, when the value is 0, it is processing line 2, i.e., $m[p_7] = 1$, when the value is 1, it is swapping from one line to the other, i.e., either $m[p_6] = 1$ or $m[p_8] = 1$. It can be observed that the machine starts swapping as soon as one of the buffers become empty, i.e., for this particular set of parameters, the maximum production is obtained when the production of both lines is alternated. Note that, as expected for an event-driven formulation, a step only takes place when an event happens, and thus, in general, the duration of the steps is variable.

Let us now assume that input flows are subject to external uncontrollable factors that can modify the controlled flow up to 10%. The proposed eMPC strategy reacts to these perturbations by recomputing its control actions at each step. Figure 5(b) shows the resulting trajectory of the system. At the first step (around time instant 12), the marking of place $p_1$ is not as high as expected. Hence, in order to maximize the items produced over the specified period of time, the controller decides not to swap the machine to line 1 in order to let buffer of line 1 fill completely. It can be seen, that although the trajectory of the disturbed controlled system is slightly different to the nominal one (Figure 5(a)), the controller manages to fill and empty buffers appropriately in order to maximize the performance.




(a)




(b)

**Fig. 5** Time trajectory of the controlled system in Figure 4 without disturbances (a); with disturbances in flows of $t_1$ and $t_3$ (b).

## 5.2 Production network

In this section we consider a production network system described in [4]. The model of the system consists of continuous places and transitions representing buffers and

flows of items, and two discrete places and transitions modeling a single machine, see Figure 6. In contrast to the model in [4] and in order to model the system more realistically, the net in Figure 6 includes complementary places for every buffer, so that the system is structurally bounded, and models the existing machine with a discrete subnet. In this model we assume that the time spent by the machine to swap from one line to the other is negligible, i.e., $\vartheta[t_7] = \vartheta[t_8] = 0$.

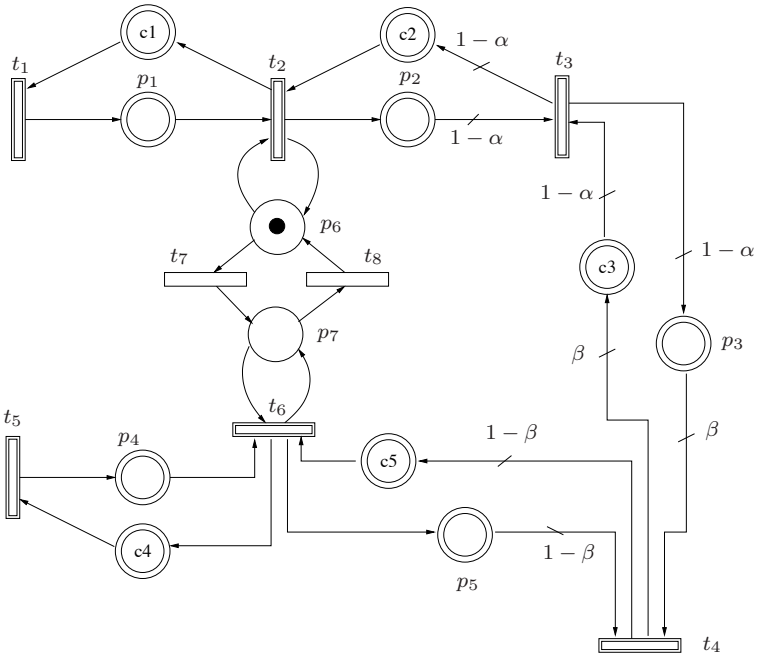



**Fig. 6** A production network.

Let the capacity of the buffers be $c_1 = 8$, $c_2 = 6$, $c_3 = 4$, $c_4 = 8$, $c_5 = 6$, the $\lambda$ of transitions be $\lambda[t_1] = 0.5$, $\lambda[t_2] = 1.5$, $\lambda_[t_3] = 0.6$, $\lambda[t_4] = 1$, $\lambda[t_5] = 0.8$, $\lambda[t_6] = 1.5$, and $\alpha = 0.1$, $\beta = 0.4$. Assume that the initial marking of the system is $m_0[p_1] = 1$, $m_0[p_2] = 3$, $m_0[p_3] = 5$, $m_0[p_4] = 2$, $m_0[p_5] = 4$ and $m_0[p_6] = 1$.

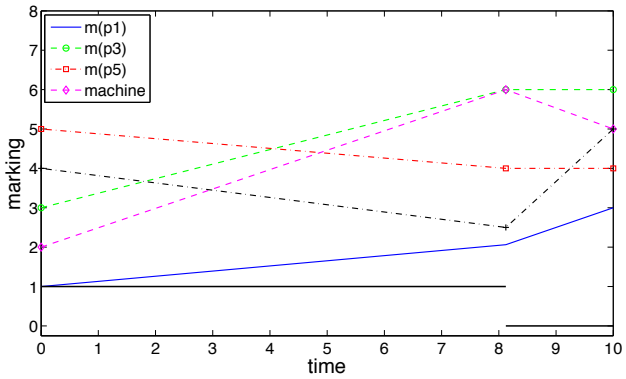
We first focus on the minimum time problem to reach the target marking $m[p_1] = 3$, $m[p_2] = 6$, $m[p_3] = 4$, $m[p_4] = 5$, $m[p_5] = 5$ and no target marking is specified for the machine. After adding the constraint for the target marking (15), the objective function of the control problem is set to minimum time criterion

$$F(m(N|\tau), \tau(N|\tau)) = 0, \quad L\Big(m(k|\tau), t(k|\tau), \mu(k|\tau)\Big) = \sum_{k=1}^{N-1} q(k|\tau), \tag{24}$$

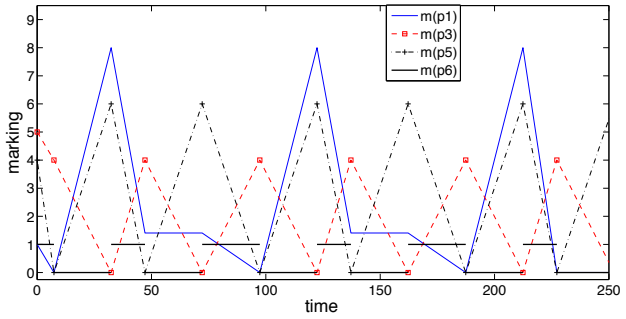
where we have set $N = 3$.

Figure 7(a) shows the time evolution of the system under the obtained control actions. The value associated to the label 'machine' indicates in which place the machine is located: if the value is 1 then the token is in $p_6$, if the value is 0 then the token is $p_7$. The target marking is reached at the second step. During the first interval of time,

which lasts 8.1 time units, the machine is processing the items in buffer $p_1$, and in the second time interval it is processing the items in buffer $p_4$.




(a)




(b)

**Fig. 7** Time trajectory of the controlled system in Figure 6 to reach a target marking (a); to maximize the number of items produced by $t_4$ (b).

Let us now assume that it is desired to maximize the number of items produced during the first 250 time units. The objective function associated to such control problem is: $L\Big(m(k|\tau), t(k|\tau), \mu(k|\tau)\Big) = -\sum_{k=1}^{N-1} v[t_4](k)$. The trajectory of the system under the computed control actions for a control horizon of 3 steps is shown in Figure 7(b). It can be observed that after the first step a repetitive pattern develops in order minimize the objective function.

## 6 Conclusions

With the aim of maintaining the performance of continuous-time approaches and the computability of discrete-time ones, in this paper we have introduced an event-driven scheme for controlling timed hybrid Petri nets. While the control action is finitely parametrized, as in discrete-time models, hence allowing the application of

optimization-based control algorithms, by selecting the events to include mode switches and constraints activation, the event-driven strategy enforces constraints and mode switches continuously in time, hence avoiding intersampling constraint violation and mode-mismatch errors.

By representing the hybrid Petri net in the proposed event-driven formalism, we have proposed a finite horizon open-loop optimal control problem that, using different control objectives, optimizes the dynamic behavior of the net. The problem has been used as the base of an event-driven model predictive control strategy that is a closed-loop control strategy and hence able to counteract the effect of external disturbances.

We have evaluated the behavior of the proposed algorithms on two examples obtained from the literature.

## References


1. H. Alla and R. David. Continuous and Hybrid Petri Nets. *Journal of Circuits, Systems, and Computers*, 8(1):159–188, 1998.
2. H. Alla and R. David. A modeling and analysis tool for discrete event systems: Continuous Petri net. *Performance Evaluation*, 33:175–199, 1998.
3. P. Antsaklis. A brief introduction to the theory and applications of hybrid systems. *Proc. IEEE, Special Issue on Hybrid Systems: Theory and Applications*, 88(7):879–886, July 2000.
4. F. Balduzzi, A. Giua, and G. Menga. First-Order Hybrid Petri Nets: a Model for Optimization and Control. *IEEE Trans. on Robotics and Automation*, 16(4):382–399, 2000.
5. A. Bemporad. *Hybrid Toolbox – User's Guide*. Dec. 2003. `http://www.dii.unisi.it/hybrid/toolbox`.
6. A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Trans. Automatic Control*, 49(5):832–838, 2004.
7. A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming- the explicit solution. *IEEE Trans. Automatic Control*, 47(12):1974–1985, 2002.
8. A. Bemporad, S. Di Cairano, and J. Júlvez. Event-driven optimal control of integral continuous-time hybrid automata. In *Proc. 44th IEEE Conf. on Decision and Control*, pages 1409–1414, Seville, Spain, 2005.
9. A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.
10. M. Braun, D. Rivera, M. Flores, W. Carlyle, and K. Kempf. A model predictive control framework for robust management of multi-product, multi-echelon demand networks. *Annual Reviews in Control*, 27(2):229–245, 2003.
11. E. Camacho and C. Bordons. *Model Predictive Control*. Springer-Verlag, 2004.
12. Dash Associates. *XPRESS-MP User Guide*, 2003. `http://www.dashoptimization.com`.
13. R. David and H. Alla. *"Discrete, Continuous, and Hybrid Petri Nets"*. Springer, 2010.
14. B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, 2001.
15. S. Di Cairano. *Model Predictive Control of Hybrid Dynamical Systems: Stabilization, Event-driven, and Stochastic Control*. PhD thesis, Dip. Ing. Informazione, Universita' di Siena, Italy, 2008.
16. S. Di Cairano, A. Bemporad, and J. Júlvez. Event-driven optimization-based control of hybrid systems with integral continuous-time dynamics. *Automatica*, 45(5):1243–1251, 2009.
17. S. Di Cairano, D. Yanakiev, A. Bemporad, I. Kolmanovsky, and D. Hrovat. An MPC design flow for automotive control and applications to idle speed regulation. In *Proc. 47th IEEE Conf. on Decision and Control*, pages 5686–5691, Cancun, Mexico, 2008.
18. F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman & Hall, 1993.
19. P. Falcone, F. Borrelli, J. Asgari, H. Tseng, and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Trans. Contr. Systems Technology*, 15(3):566–580, 2007.

20. A. Giua, C. Mahulea, L. Recalde, C. Seatzu, and M. Silva. Optimal control of timed continuous Petri nets via explicit MPC. *Lecture notes in control and information sciences*, 341:383–390, 2006.
21. Ø. Hegrenæs, J. Gravdahl, and P. Tøndel. Spacecraft attitude control using explicit model predictive control. *Automatica*, 41(12):2107–2114, 2005.
22. ILOG, Inc. *CPLEX 9.0 User Manual*. Gentilly Cedex, France, 2004.
23. M. Johansson and A. Rantzer. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automatic Control*, 43(4):555–559, 1998.
24. J. Júlvez, A. Bemporad, L. Recalde, and M. Silva. Event-Driven Optimal Control of Continuous Petri Nets. In *43rd IEEE Conference on Decision and Control (CDC)*, pages 69–74, Paradise Island. Bahamas, 2004.
25. J. Maciejowski. *Predictive control with constraints.* Englewood Cliffs, NJ: Prentice Hall., 2002.
26. C. Mahulea, A. Giua, L. Recalde, C. Seatzu, and M. Silva. Optimal model predictive control of Timed Continuous Petri nets. *IEEE Trans. Automatic Control*, 53(7):1731–1735, 2008.
27. A. Makhorin. *GLPK (GNU Linear Programming Kit) User's Guide*, 2003.
28. D. Mignone, G. Ferrari-Trecate, and M. Morari. Stability and stabilization of piecewise affine and hybrid systems: An LMI approach. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 504–509, Dec. 2000.
29. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
30. S. Qin and T. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 93, no. 316:733–764, 2003.
31. L. Recalde and M. Silva. Petri Nets Fluidification revisited: Semantics and Steady state. *APII-JESA*, 35(4):435–449, 2001.
32. M. Silva. Introducing Petri Nets. *Practice of Petri Nets in Manufacturing, Chapman & Hall*, pages 1–62, 1993.
33. E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automatic Control*, 26(2):346–358, April 1981.
34. J. Xu, L. Recalde, and M. Silva. Tracking control of join-free timed continuous petri net systems under infinite servers semantics. *Discrete Event Dynamic Systems*, 18(2):263–283, 2008.