Observer-based state-feedback control of timed Petri nets with deadlock recovery: theory and implementation

Francesco Basile
Dip. Ing. dell'Informazione e Ing. Elettrica, Università di Salerno
Via Ponte don Melillo, 84084 Fisciano (Salerno), Italy
Phone: +39-089-96-4400, Fax: +39-06-233 227 957, E-mail: fbasile@unisa.it

Alessandro Giua, Carla Seatzu
Dip. di Ing. Elettrica ed Elettronica, Università di Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
Phone: +39-70-675 5751, Fax: +39-70-675 5782, E-mail: {giua,seatzu}@diee.unica.it

Abstract— **When a timed Petri net is controlled using an observer in the control loop, the use of crude marking estimates may reduce the system performance and often leads to partial or total deadlocks. Here we show an efficient procedure that, using the knowledge of timing delays associated to transitions, may improve the marking estimate and allow the net to recover from partial or total deadlocks. A software developed for implementing this procedure is also described in the paper.**

I. Introduction

In this paper we deal with the issue of controlling a Petri net whose marking cannot be measured. The approach we follow in this paper is based on the classical system theory notion of a state-feedback controller that uses an observer to estimate the plant state.

In previous works [GIU, 02] we have shown how it is possible to estimate the actual marking of the net based on the observation of a word of events (i.e., transition firings) and an algorithm was given for computing the marking estimate and error bound. In particular, the set $\mathcal{C}$ of *markings consistent with an observed word*, i.e., the set of markings in which the system may actually be given the observed word, can easily be described in terms of the observer estimate and can be characterized as the integer solutions of a linear constraint set. Other approaches to the design of Petri net observers can also be found in [MED, 98], [RAM, 00].

In [GIU, 02] we have also shown how the estimate generated by the observer may be used to design a state feedback controller, that ensures that the controlled system never enters a set of forbidden states. We considered a special class of safeness specifications that limit the weighted sum of markings in subsets of places called generalized mutual exclusion constraints (GMEC). The problem of controlling a Petri net under incomplete information has also been discussed by Zhang and Holloway [ZHA, 95].

Published in the *Proc. of the Symp. on Discrete Events in Industrial and Manufacturing Systems* at the *IMACS/IEEE Multiconference CESA' 2003* (Lille, France), July 2003.

Clearly, the use of marking estimates, as opposed to the exact knowledge of the actual marking of the plant, leads to a worse performance of the closed-loop system. In fact, in a safeness problem the aim of the controller is that of preventing all those transition firings that lead to a forbidden marking. If the actual marking is not exactly known, but is only known to belong to a given consistent set $\mathcal{C}$, the controller must forbid all transitions firing that from "any" marking in $\mathcal{C}$ may lead to a forbidden marking and the controller becomes usually more restrictive as the cardinality of this set increases. Because of this it may be the case that the controlled system reaches a deadlock, i.e., a blocking condition.

In two previous papers [BAS, 01], [BAS, 02] we have shown that using siphon analysis, the set of deadlock markings $\mathcal{M}_b$ of a structurally bounded net can be characterized as the integer solution of a linear constraint set. (Similar techniques have also been used for deadlock analysis and avoidance by Barkaoui [BAR, 95], [BAR, 97], Chu and Xie [XIE, 97], Ezpeleta et *al.* [EXP, 95], and Park and Reveliotis [PAR, 01].)

In [BAS, 02] we considered timed Petri nets, i.e., Petri nets where a delay is associated to each transition. The delay represents the time that must elapse from the enabling of the transition until it fires. In a timed net, let us assume that a transition has been control enabled for a period of time longer that its delay without firing (we say that the transition has timed out): one can conclude that the transition was not marking enabled during that same period. In this case a procedure that uses this additional information to improve the marking estimate and that we call transition time-out (TTO) procedure, is invoked.

In Section 2 and Section 3 we recall the background material on Petri nets and on their control using observers following the procedure presented in [BAS, 02].

The new contribution of this paper with respect to [BAS, 02] is threefold.

Firstly, in Section 4 we describe the architecture of a software based on the Xpress Optimizer C/C++ li-

brary that we have developed for the control of Petri nets using observers. It consists of a timed Petri net simulator, of a Petri net observer that computes the estimate, of a controller that given the marking estimate and bound computes a safe control pattern, and finally of a module that implements the recovery procedure invoked whenever a transition times out.

Secondly, in Section 5 we use the software to solve an application example that is different from that one discussed in [BAS, 02].

Thirdly, in Section 6 we show with an example that unlike the procedure presented in [BAS, 01] — that could only be invoked when a total deadlock has occured — the TTO procedure based on the net time out may allow one to detect partial deadlocks as well, and in general it improves and accelerates the convergence of the marking estimation procedure.

II. Background on Petri nets

In this section we recall the formalism used in the paper. For more details on Petri nets we address to [MUR, 89].

A *Place/Transition net* (P/T net) is a structure $N = (P, T, Pre, Post)$, where P is a set of m places; T is a set of n transitions; $Pre : P \times T \rightarrow \mathbb{N}$ and $Post : P \times T \rightarrow \mathbb{N}$ are the *pre–* and *post–* incidence functions that specify the arcs; $C = Post - Pre$ is the incidence matrix. The *preset* and *postset* of a node $X \in P \cup T$ are denoted ${}^\bullet X$ and $X^\bullet$ while ${}^\bullet X^\bullet = {}^\bullet X \cup X^\bullet$.

A *marking* is a vector $M : P \rightarrow \mathbb{N}$ that assigns to each place of a P/T net a non–negative integer number of tokens, represented by black dots. In the following we denote $M(p)$ the marking of place p.

A transition t is enabled at M if $M \geq Pre(\cdot, t)$ and may fire yielding the marking $M' = M + C(\cdot, t)$. We write $M\ [w\rangle\ M'$ to denote that the enabled sequence of transitions w may fire at M yielding M'. Finally, we denote w_0 the sequence of null length.

A marking M is *reachable* in N from M_0 iff there exists a firing sequence w such that $M_0\ [w\rangle\ M$. The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0\rangle$ and is denoted $R(N, M_0)$.

A nonnegative integer vector $\vec{x} \neq \vec{0}_m$ such that $\vec{x}^{\,T} \cdot C = \vec{0}_n{}^T$ is called a *P–invariant* (here $\vec{0}_k$ denotes a $k \times 1$ vector of zeros).

A transition t is said to be *live* if for any $M \in R(N, M_0)$, there exists a sequence of transitions firable from M which contains t. A Petri net is said to be live if all transitions are *live*. A Petri net is said to be *deadlock–free* if at least one transition is enabled at every reachable marking.

A place p is said to be *bounded* if there exists a constant k such that $M(p) \leq k$ for all $M \in R(N, M_0)$. A net system is bounded if all places are bounded. A net is *structurally bounded* if it is bounded for all initial markings.

Definition 1: Given a net $N = (P, T, Pre, Post)$, and a subset $T' \subseteq T$ of its transitions, we define the *$T'-$induced subnet of N* as the new net $N' = (P, T', Pre', Post')$ where $Pre', Post'$ are the restriction of $Pre, Post$ to T'. The net N' can be thought as obtained from N removing all transitions in $T \setminus T'$. We also write $N' \prec_{T'} N$. ■

A deterministic *timed* P/T net is a pair (N, δ), where $N = (P, T, Pre, Post)$ is a standard P/T net, and $\delta(t) : T \rightarrow \mathbb{R}_0^+$, called release delay, assigns a non-negative fixed firing duration to each transition. A transition with a release delay equal to 0 is said to be immediate. The value of $\delta(t)$ represents the time that must elapse, starting from the time at which the transition t is enabled, until it fires. We use single server-semantics, i.e., no concurrent firings of the same transition are possible.

Finally, we conclude this section recalling a linear algebraic characterization of deadlock markings derived in [BAS, 01] that will be used in the paper. Such a characterization is valid for ordinary and structurally bounded Petri nets. Note that similar linear characterizations have been independently proposed in [BAR, 97], [XIE, 97], [PAR, 01].

Theorem 2 ([BAS, 01]) Given a structurally bounded net N with m places, a marking $M \in \mathbb{N}^m$ is a deadlock marking if and only if there exists a vector $\vec{s} \in \{0,1\}^m$ such that the following set of linear equations is satisfied:

$$\mathcal{D}(N) := \begin{cases} K_1 \cdot Pre^T \cdot \vec{s} \geq Post^T \cdot \vec{s} & (a) \\ K_2 \cdot \vec{s} + M \leq K_2 \cdot \vec{1}_m & (b) \\ \vec{s} + M \geq \vec{1}_m & (c) \\ Pre^T \cdot \vec{s} \geq \vec{1} & (d) \\ M \in \mathbb{N}^m & (e) \\ \vec{s} \in \{0,1\}^m & (f) \end{cases} \quad (1)$$

where $K_1 = \max_{t \in T} Post^T(\cdot, t) \cdot \vec{1}$ and K_2 is any positive integer greater or equal to the maximum structural bound of p, for any $p \in P$. ■

By virtue of the linear characterization above, we define the set of blocking markings of a net N as:

$$\mathcal{M}_b(N) = \{M \ \mid\ \exists\ \vec{s} \in \{0,1\}^m : (M, \vec{s}) \in \mathcal{D}(N)\}. \quad (2)$$

III. Control with marking estimation and time-out

In this paper we assume that partial information about the initial marking is available. In particular, we assume that the initial marking is given in the form of a *macromarking*.

Definition 3 ([GIU, 02]) The *macromarking* defined by $V \in \mathbb{N}^{m \times r}$ and $\vec{b} \in \mathbb{N}^r$ is the set of markings $\mathcal{V}(V, \vec{b}) = \{M \in \mathbb{N}^m \mid V^T M = \vec{b}\}$. ■

The notion of macromarking occurs frequently when describing systems containing a known set of resources (e.g., parts, machines) whose actual conditions (e.g.,

exact location of parts within the plant, state of a machine) is unknown.

We make the following assumptions.

A1) The structure of the net $N = (P, T, Pre, Post)$ is known, while the initial marking M_0 is not.
A2) The event occurrences (i.e., the transition firings) can be observed.
A3) The initial marking M_0 belongs to the macromarking $\mathcal{V}(V, \vec{b})$, i.e., it satisfies the equation $V^T M_0 = \vec{b}$.

We also introduce the following notation.

Definition 4 ([GIU, 02]) After the word w has been observed we define the set of $w-$*consistent markings* as

$$\mathcal{C}(w) = \{M \in \mathbb{N}^m \mid \exists M_0 \in \mathcal{V}(V, \vec{b}),\ M_0[w\rangle M\}.$$

i.e., as the set of all markings in which the system may be, given the observed behaviour and the initial marking. ■

In a previous work [GIU, 02] was provided a simple algorithm[1] to compute the estimate μ and the bound B of each actual marking M based on the observation of a word of events and on the knowledge of the initial macromarking $\mathcal{V}(V, \vec{b})$.

The following important result was also proved.

Theorem 5 ([GIU, 02]) Let us consider a net with initial macromarking $\mathcal{V}(V, \vec{b})$. Let w be an observed word, and μ and B be the corresponding estimate and bound computed using the estimation algorithm in [GIU, 02]. We define the set of (μ, B)-*consistent markings*

$$\mathcal{M}(\mu, B) = \{M \in \mathbb{N}^n \mid M \geq \mu,\ V^T \cdot M = V^T \cdot \mu + B\} \quad (3)$$

and it holds that the set of w-consistent markings coincides with the set of (μ, B)-consistent markings, i.e., $\mathcal{C}(w) = \mathcal{M}(\mu, B)$. ■

In [GIU, 02] we showed how the marking estimate can be used by a control agent to enforce a given specification on the plant behaviour. In particular, we made several assumptions that are briefly summarized here.

- We considered a special type of state specifications called *generalized mutual exclusion constraints* (GMEC) that have been considered by various authors [GIU, 92], [LI, 94], [YAM, 96].
Given an integer matrix $L = [\vec{l}_1 \cdots \vec{l}_q]$ with $\vec{l}_j \in \mathbb{Z}^m$ and a vector $\vec{k} = [k_1, \cdots, k_q]$ with $k_j \in \mathbb{Z}$, a GMEC $(L, \vec{k})$ defines the set of legal states

$$\mathcal{L} = \{M \in \mathbb{N}^m \mid L^T \cdot M \leq \vec{k}\}.$$

- The controller may disable transitions to prevent the plant from entering a forbidden marking, computing a control pattern $f(t, M) : T \times \mathbb{N}^m \to \{0, 1\}$. If $f(t, M) = 0$ then t is disabled by the controller at M.

[1]This algorithm is not reported here: it consists in steps 7.(a)-(c) of the more general Algorithm 7 presented in the following.

- All transitions are controllable, i.e., can be disabled by the controller.

When an observer is used in the control loop, the actual marking M is not known and only the set of consistent markings $\mathcal{C} \subseteq \mathbb{N}^m$ is available to the controller. The control law thus becomes a function $f(t, \mathcal{C}) : T \times 2^{\mathbb{N}^m} \to \{0, 1\}$ and can be given as follows.

Definition 6: [GIU, 02] **[Optimal state feedback with observer]** Given a GMEC $(L, \vec{k})$ and a set of consistent markings $\mathcal{C} \subseteq \mathbb{N}^m$, the firing of transition t should be prevented if and only if there exists a legal consistent marking M such that the firing of t from M leads to a forbidden marking, i.e.,

$$f(t, \mathcal{C}) = \begin{cases} 0 & \text{if } (\exists M)\ M \in \mathcal{C},\ L^T \cdot M \leq \vec{k}, \\ & \quad M[t\rangle M',\ (\exists j)\ \vec{l}_j \cdot M' > k_j \\ 1 & \text{otherwise.} \end{cases}$$

The computation of the control pattern may be carried out solving a number of linear integer programming problems (IPP) [GIU, 02]. ■

A. Control and estimate updating after transition time-out

Now, let us recall a general approach to exploit available information on the timing structure of the net so as to obtain a better estimate of the set of consistent markings. The approach has been firstly proposed by the authors in [BAS, 02] and is essentially based on the linear algebraic characterization of deadlock markings given by the system of inequalities (1).

Let us assume that a known delay $\delta(t) : T \to \mathbb{R}$ is associated to each transition. We say that a transition t has *timed-out* at time *now* if it has been control enabled without firing during the time interval $[now - \delta(t), now]$ and the marking of its input places ${}^\bullet t$ has not increased during this interval. Thus, we can be sure that at time *now* the actual marking M is such that $\neg M[t\rangle$, or equivalently t is not marking enabled. The set of timed-out transitions is denoted T_{to}.

The procedure that we describe in Algorithm 7 considers two types of events that modify the marking estimate. The first type of events occurs when the firing of a transition $\hat{t}$ is detected, while the second type of events occurs when a new transition times-out.

Algorithm 7: **Control and Estimate Updating After Transition Time-Out**

In this algorithm the variable *now* represents the current value of the time. At each instant of time it is possible to partition the set of transitions T into three subsets:

T_n is the set of transitions that are *not control enabled* given the current set of consistent markings. A transition t belongs to this set if $f(t, \mathcal{C}) = 0$.

T_{to} is the set of *control enabled* transitions that have *timed-out*. A transition t belongs to this set if during

the time interval $[now - \delta(t), now]$ has continuously been control enabled and the marking of all its input places ${}^\bullet t$ has not increased during this same interval.

T_e is the set of those control *enabled* transitions that do not belong to T_{to}.

These are the steps of the algorithm.

1. Let $\mu = \mu_{w0}$ and $B = B_{w0}$ be the initial estimate and bound, and let $\mathcal{C} = \mathcal{M}(\mu_{w0}, B_{w0})$ be the initial set of consistent markings.

2. Compute for all transitions $t \in T$ the control pattern $f(t, \mathcal{C})$ and let

$$\begin{aligned} T_{to} &= \emptyset, \\ T_n &= \{t \in T \mid f(t, \mathcal{C}) = 0\}, \\ T_e &= \{t \in T \mid f(t, \mathcal{C}) = 1\}. \end{aligned}$$

3. Set for all $t \in T_e$ the current clock to $\omega(t) = \delta(t)$.

4. Let $\delta = \min\{\omega(t) \mid t \in T_e\}$ the time to wait (step 6).

5. Let $\tau = now$ and $f_{old}(t) = f(t, \mathcal{C})$ (keeps track of the previous control pattern).

6. Wait until

(a) EITHER a transition $\hat{t}$ fires and THEN go to 7

(b) OR $now = \tau + \delta$ and THEN go to 8.

If one event of type (a) and one event of type (b) occur simultaneously, then condition 6.a takes priority.

7. Activate the observer update procedure.

(a) Update the estimate to μ' with $\mu'(p) = \max\{\mu(p), Pre(p, \hat{t})\}$.

(b) Let the current estimate and bound be $\mu = \mu' + C(\cdot, \hat{t})$ and $B = B - V^T \cdot (\mu' - \mu)$.

(c) Let the current set of consistent markings be $\mathcal{C} = \mathcal{M}(\mu, B)$.

(d) Compute for all transitions $t \in T$ the control pattern $f(t, \mathcal{C})$ and let

$$\begin{aligned} T_{to} &= T_{to} \setminus \{t \in T_{to} \mid {}^\bullet t \cap \hat{t}^\bullet \neq \emptyset\}, \\ T_n &= \{t \in T \mid f(t, \mathcal{C}) = 0\}, \\ T_e &= \{t \in T \mid f(t, \mathcal{C}) = 1,\ t \notin T_{to}\}. \end{aligned}$$

(e) Update the clocks of enabled transitions.

IF a transition $t \in T_e$ satisfies at least one of the following three conditions

i. $f_{old}(t) = 0$ {newly control enabled}

ii. ${}^\bullet t \cap \hat{t}^\bullet \neq \emptyset$ {may have become marking enabled by the firing of $\hat{t}$}

iii. $t = \hat{t}$ {it is the transition that has just fired}

THEN $\omega(t) = \delta(t)$ {reset the clock}
ELSE $\omega(t) = \omega(t) - (now - \tau)$.

(f) Go to 4.

8. Activate the time-out procedure.

(a) Let $T_{to} = T_{to} \cup \{t \in T_e \mid \omega(t) = \delta\}$.

(b) Let $N_{to} \prec_{T_{to}} N$ be the $T_{to}-$induced subnet N.

(c) Compute for all transitions $t \in T$ the control pattern $f(t, \mathcal{C} \cap \mathcal{M}_b(N_{to}))$ and let

$$\begin{aligned} T_n &= \{t \in T \mid f(t, \mathcal{C}) = 0\}, \\ T_e &= \{t \in T \mid f(t, \mathcal{C}) = 1,\ t \notin T_{to}\}. \end{aligned}$$

(d) Improve the previous estimate μ. This simply requires the solution of m linear integer programming problems (IPP), one for each place $p_i \in P$:

$$\begin{cases} \min M(p_i) \\ s.t. \\ M \in \mathcal{M}(\mu, B) \\ M \in \mathcal{M}_b(N_{to}) \end{cases} \tag{4}$$

Now, let $\mu^* = [\mu_1^* \ \cdots \ \mu_m^*]^T$, where μ_i^* is the solution of the i–th IPP, and let $B^* = B - V^T(\mu^* - \mu)$.

(e) Update the estimate and bound to $\mu = \mu^*$ and $B = B^*$, and compute the new set of consistent markings $\mathcal{C} = \mathcal{M}(\mu, B)$.

(f) If $T_e = \emptyset$ exit (the net is deadlocked and the time-out procedure fails to recover from the deadlock), else goto 4. ■

The main idea behind the algorithm is the following. If T_{to} is the set of transitions that have timed out at time *now* we can be sure that the actual marking M must also belong to the set of blocking markings for the net N_{to} obtained from N removing all transitions not in T_{to}. Thus in step 8.c we can compute a (possibly) less restrictive control pattern using as set of consistent markings $\mathcal{C} \cap \mathcal{M}_b(N_{to})$.

This set, even if defined by a set of linear inequalities — namely, the constraint set of IPP (4) — is not in the simple form given by eq. (3) that is required in the following step of the algorithm. Thus at step 8.d we approximate it with a set of the form given by eq. (3) computing new estimates and bounds.

IV. Simulation and implementation of the procedure

In this section we focus on the main aspects of the simulation or implementation of the procedure presented in the Algorithm 7.

We observe that two sets of IPP problems have to be solved: the first one, is the set of optimization problems required for the computation of $f(t, \mathcal{C})$ at step 2; the second one is the set of m optimization problems (4) required to improve the previous estimate μ at step 8. In addition, at step 8 again, once that an induced subnet N_{to} has been defined the computation of K_2 requires the solution of an LP problem. Thus, a number of calls to an IPP/LP solver library is necessary at each step of the algorithm. We have developed a simulator using the Xpress Optimizer C/C++ library.

As shown in Figure 1 we implement in C language the controller algorithm. The main program of the controller can communicate with an open timed PN simulator or can be properly interfaced with the plant. Every time an event is generated (i.e. a transition $\hat{t}$

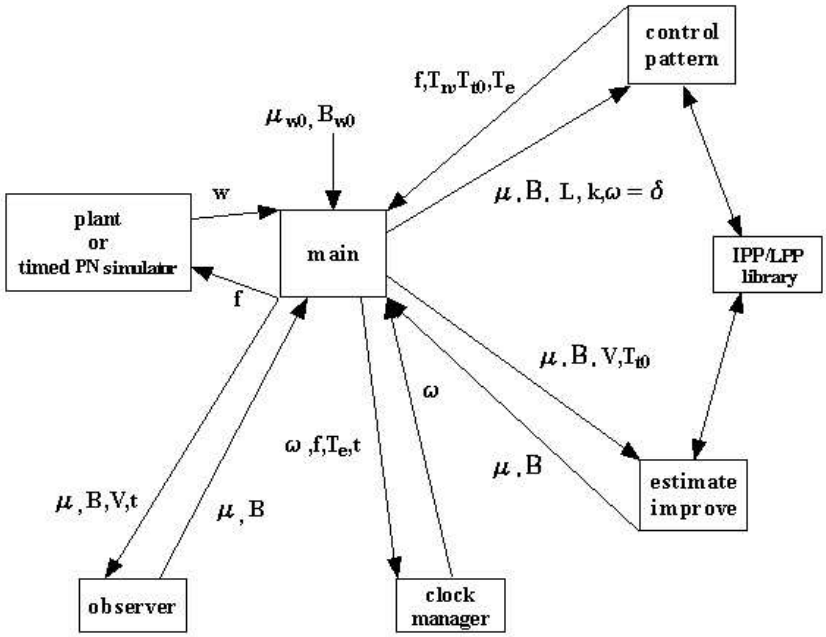

Fig. 1. A possible architecture for an observer based feedback-control of timed PN modeled plant.

fires) a new control pattern is sent to the plant or to the simulator.

We found useful in its development to write a main program and four procedures described in the following:

1. $[f,T_n,T_{to},T_e]$=`control pattern`$(\mu,B,L,k,\omega = \delta)$ - this procedure refers to step 2, steps 7.c-d and steps 8.a-c; it first computes T_{to} (that is initially an empty set) and after $f(t,\mathcal{C})$ from the solution of an IPP problem where $\mathcal{C} = \mathcal{M}(\mu,B) = \{M \in \mathbb{N}^n \mid M \geq \mu,\ V^T \cdot M = V^T \cdot \mu + B\}$ and $\mathcal{M}_b(N_{to}) = \{M \mid \exists\ \vec{s} \in \{0,1\}^m : (M,\vec{s}) \in \mathcal{D}(N_{to})\}$ in the case that the logical condition $\omega = \delta$ is true; finally, the sets T_n, T_e are computed.
2. $[\omega]$=`clock manager`$(\omega,f,T_e,\hat{t})$ - this procedure updates the clocks according to step 7.e after the firing of a transition $\hat{t}$.
3. $[\mu,B]$=`observer`$(\mu,B,V,\hat{t})$ - this procedure refers to steps 7.a-b; it is invoked when a transition $\hat{t}$ fires so that the current estimate μ and bound B are updated according to the macromarking constraint $\mathcal{V}(V,V^T M_0)$.
4. $[\mu,B]$=`estimate improve`(μ,B,V,T_{to}) - this procedure refers to steps 8.d-e; on the basis of T_{to} the current estimate μ and bound B are updated by solving m IPP problems (4) and then by imposing the macromarking constraint.

By using the four procedure above the main program results to be the following.

[f]=main(w)

1. $\mu = \mu_{w0}$; $B = B_{w0}$;
2. $[f,T_n,T_{to},T_e]$=`control pattern`$(\mu_{w0},B_{w0},L,k,0)$
3. `if` $t \in T_e$ `then` $\omega(t) = \delta(t)$.
4. $\delta = \min\{\omega(t) \mid t \in T_e\}$ (* time to wait for a transition firing *)
5. $\tau = now$; $f_{old} = f$; (* keeps track of the previous control pattern *).
6. Wait until
   (a) EITHER a transition $\hat{t}$ fires and THEN go to 7
   (b) OR $now = \tau + \delta$ and THEN go to 8.

If one event of type (a) and one event of type (b) occur simultaneously, then condition 6.a takes priority.
7. Activate the observer update procedure.
   (a) $[\mu,B]$=`observer`$(\mu,B,V,\hat{t})$
   (b) $[f,T_n,T_{to},T_e]$=`control pattern`$(\mu,B,L,k,0)$
   (c) $[\omega]$=`clock manager`$(\omega,f,T_e,\hat{t})$
   (d) Goto 4
8. Activate the time-out procedure.
   (a) $[f,T_n,T_{to},T_e]$=`control pattern`$(\mu,B,L,k,1)$
   (b) $[\mu,B]$=`estimate improve`(μ,B,V,T_{to})
   (c) `if` $T_e = \emptyset$ exit (* the net is deadlocked and the time-out procedure fails to recover from the deadlock*) `else` goto 4 ■

V. A manufacturing example

Now, let us apply the above methodology to a manufacturing system whose Petri net model is shown in Figure 2.

This assembly system, that is similar to the one described in [PRO, 93], consists of five machines, $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$ and $\mathcal{M}_5$ whose operational process is modeled by the firing of transitions t_1, t_2, t_3, t_4 and t_5, respectively. Two principal types of operations are involved in this manufacturing system: *regular operations* and *assembly operations*. Regular operations (modeled by transitions t_1, t_2 and t_5) just transform a component of the intermediate product. Assembly

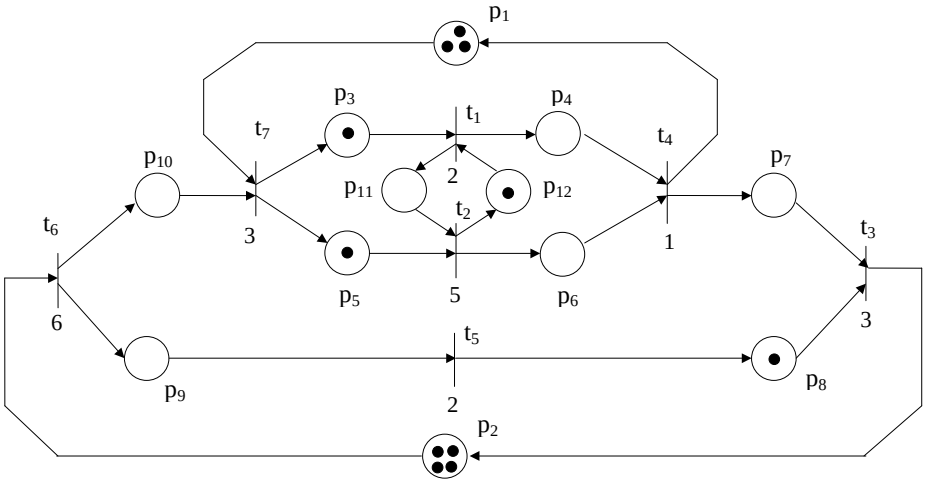

Fig. 2. Petri net model of the assembly system

operations (modeled by transitions t_3 and t_4) put components together to obtain a more complex component of a final product or the final product itself. Note that this model uses transitions (t_6 and t_7) which do not represent operations but the beginning of the manufacturing of components which are required to assemble a more complex component or the final product. In this example there are two manufacturing levels, the primary one, performed by $\mathcal{M}_3$, leads to finite product, the secondary one, performed by $\mathcal{M}_4$, leads to semi–finished (in–working) product.

The markings of places p_1 and p_2 represent the number of assembly servers for t_4 and t_3 respectively. The marking of places p_3, p_5, and p_9 represent the availability of parts to be processed (raw materials), while the marking of places p_4, p_6, p_7 and p_8 represent the availability of semi–finished products. Places p_{11} and p_{12} ensure that machines t_1 and t_2 work alternatively.

The Petri net model in Figure 2 is a strongly connected event graph with $m = |P| = 12$ and $n = |T| = 7$. There exist ten elementary circuits, that correspond to an equal number of P-invariants. If we assume that the initial marking of the net is $M_0 = [3\ 4\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1]^T$, we have (here to avoid a heavy notation we denote as M_i the marking of place p_i)

$$\begin{cases} M_2 + M_4 + M_5 + M_7 + M_{10} + M_{12} = 6 & (1) \\ M_2 + M_3 + M_6 + M_7 + M_{10} + M_{11} = 5 & (2) \\ M_2 + M_5 + M_6 + M_7 + M_{10} = 5 & (3) \\ M_2 + M_3 + M_4 + M_7 + M_{10} = 5 & (4) \\ M_2 + M_8 + M_9 = 5 & (5) \\ M_1 + M_4 + M_5 + M_{12} = 5 & (6) \\ M_1 + M_3 + M_6 + M_{11} = 4 & (7) \\ M_1 + M_5 + M_6 = 4 & (8) \\ M_1 + M_3 + M_4 = 4 & (9) \\ M_{11} + M_{12} = 1 & (10) \end{cases} \tag{5}$$

We assume that the above set of P-invariants coincides with the macromarking, thus $B_{w_0} = \vec{b} = [6\ 5\ 5\ 5\ 5\ 5\ 5\ 4\ 4\ 4\ 1]^T$.

Moreover, we assume that the controller must enforce two specifications:

$$\begin{cases} M_3 + M_5 \leq 3 & (a) \\ M_9 \leq 3. & (b) \end{cases} \tag{6}$$

The first specification requires that at most 3 raw parts may be simultaneously waiting to be processed by either machine $\mathcal{M}_1$ or $\mathcal{M}_2$. The second specification requires that at most 3 raw parts may be waiting to be processed by machine $\mathcal{M}_5$.

Finally, we assume that the delay times associated to transitions are those reported in Figure 2.

Note that the untimed version of this Petri net system has already been considered by the same authors in [BAS, 01].

If the marking of the net is measurable, then the controlled net is live, as it can be verified by reachability analysis. On the contrary, if the marking of the plant is not measurable, an observer must be used in the control loop and, as shown in [BAS, 01], this may even lead the closed loop net to a deadlock.

In this paper we examine in detail the closed loop behaviour of the timed Petri net system in Figure 2 assuming that the TTO procedure is applied. The resulting evolution is represented in the reachability graph in Figure 3 where each node is labeled $(M/\mu/B)$. A thick arrows denotes a time-out and is labeled by the corresponding set T_{to}. A thin arrow denotes a transition firing.

At step 1 we define the initial estimate and bound. At step 2 we compute for all transitions $t \in T$ the control pattern $f(t, \mathcal{C})$ and set $T_n = \{t_6, t_7\}$, $T_{to} = \emptyset$ and $T_e = T \setminus T_n$. In fact, the firing of t_6 may potentially violate specification (b), while the firing of t_7 may potentially violate specification (a). Then, we set up the clock value of each transition in T_e to its time delay. Given the actual delays, the time-out to wait before either applying the observer update procedure or the deadlock recovery procedure, is $\delta = 1$.

(3 4 1 0 1 0 0 1 0 0 0 1 / 0 0 0 0 0 0 0 0 0 0 0 0 / 6 5 5 5 5 5 4 4 4 1)

⇩ { t_4 } now = 1

(3 4 1 0 1 0 0 1 0 0 0 1 / 0 0 0 0 0 0 0 0 0 0 0 0 / 6 5 5 5 5 5 4 4 4 1)

↓ t_1 now = 2

(3 4 0 1 1 0 0 1 0 0 1 0 / 0 0 0 1 0 0 0 0 0 0 1 0 / 5 4 5 4 5 4 3 4 3 0)

⇩ { t_5 } now = 2

(3 4 0 1 1 0 0 1 0 0 1 0 / 0 0 0 1 1 0 0 1 0 0 1 0 / 4 4 4 4 4 3 3 3 3 0)

⇩ { t_3, t_4, t_5 } now = 3

(3 4 0 1 1 0 0 1 0 0 1 0 / 0 0 0 1 1 0 0 1 0 0 1 0 / 4 4 4 4 4 3 3 3 3 0)

⇩ { t_1, t_3, t_4, t_5 } now = 4

(3 4 0 1 1 0 0 1 0 0 1 0 / 0 0 0 1 1 0 0 1 0 0 1 0 / 4 4 4 4 4 3 3 3 3 0)

↓ t_2 now = 7

(3 4 0 1 0 1 0 1 0 0 0 1 / 0 0 0 1 0 1 0 1 0 0 0 1 / 4 4 4 4 4 3 3 3 3 0)

↓ t_4 now = 8

(4 4 0 0 0 0 0 1 1 0 0 0 1 / 1 0 0 0 0 0 1 1 0 0 0 1 / 4 4 4 4 4 3 3 3 3 0)

⇩ { t_1, t_4, t_5 } now = 9

(4 4 0 0 0 0 0 1 1 0 0 0 1 / 4 0 0 0 0 0 1 1 0 0 0 1 / 4 4 4 4 4 0 0 0 0 0)

↓ t_3 now = 11

(4 5 0 0 0 0 0 0 0 0 0 1 / 4 1 0 0 0 0 0 0 0 0 0 1 / 4 4 4 4 4 0 0 0 0 0)

⇩ { t_1, t_2, t_4, t_5, t_7 } now = 12

(4 5 0 0 0 0 0 0 0 0 0 1 / 4 1 0 0 0 0 0 0 0 0 0 1 / 4 4 4 4 4 0 0 0 0 0)

⇩ { t_1, t_2, t_3, t_4, t_5, t_7 } now = 14

(4 5 0 0 0 0 0 0 0 0 0 1 / 4 5 0 0 0 0 0 0 0 0 0 1 / 0 0 0 0 0 0 0 0 0 0)

Fig. 3. The evolution of the net in Figure 2 under control when the TTO procedure is applied.

At time $now = 1$, no transition fires and t_4 times out. Thus the time-out procedure is activated (step 8). This first implies the updating of $T_{to} = \emptyset$ to $T_{to} = \{t_4\}$. Then, we define the net N_{to} obtained from N removing all transitions not in T_{to}. For all $t \in T$ we compute the new control pattern $f(t, \mathcal{C})$ according to step 8.c and we update the transition partitioning. In particular, we find out that both t_6 and t_7 are still disabled by the controller, thus $T_n = \{t_6, t_7\}$, while $T_e = T \setminus (T_n \cup T_{to})$. Now, by solving $m = 12$ IPP we compute the new marking estimate and bound and go back to step 4 of the algorithm. In such a case, we find out that the updated marking estimate and bound are coincident with the previous ones. We compute the new value δ and, as in the previous step, it holds that $\delta = 1$.

At time $now = 2$, when one more time unit has elapsed, both conditions 6.a and 6.b are simultaneously satisfied because t_1 fires and t_5 times out. Condition 6.a takes priority and transition t_1 fires. The observer update procedure is applied. We update the estimate and bound as shown in Figure 3, while the control pattern keeps the same for all transitions $t \in T$. Note that the firing of t_1 increases the token content of place p_4 that is an imput place for t_4: thus

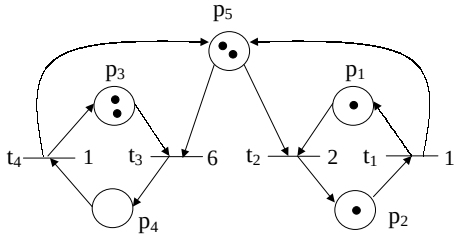

Fig. 4. An example showing how the knowledge of the timing structure may be used to solve partial deadlocks.

t_4 is removed from the set T_{to} at step 7.d. We compute the new value δ and it holds that $\delta = 0$ because t_5 is ready to time out.

Then, always at time $now = 2$, the time-out procedure is activated for t_5. This enables us to improve the marking estimate as shown in Figure 3 and also to make transition t_6 control enabled. More precisely, at time $now = 2$, after the TTO procedure has been applied, it holds that $T_n = \{t_7\}$, $T_{to} = \{t_5\}$ and $T_e = T \setminus (T_n \cup T_{to})$. Once again, at step 4, we find out that $\delta = 1$.

At time $now = 2$, after one more time unit has elapsed, no transition fires. Therefore, the time-out procedure is invoked with $T_{to} = \{t_3, t_4, t_5\}$, and so on.

As it can be seen in Figure 3, at the end of this evolution path, at time $now = 14$, the marking is completely reconstructed and no further deadlock may occur.

Let us finally observe that, if we apply the procedure presented in [BAS, 01], we are only able to completely reconstruct the actual marking of the net after a time interval that is greater than 21 time units.

VI. Partial deadlocks

In this section we want to highlight an important feature of the procedure based on transition time-outs (TTO), namely the fact that it may allow the net to recover from partial deadlocks.

This is a significant improvement with respect to the procedure previously proposed by the same authors in [BAS, 02] that is based on net time-outs (NTO): in fact the latter procedure does not use timing information and can only be invoked when a total deadlock has occurred.

Example 8: Let us consider the net system in Figure 4 with $m = 5$ places and $n = 4$ transitions. There exist 3 circuits, each one corresponding to a P-invariant. If the initial marking is that shown in Figure 4 we have:

$$\mathcal{V}(V, \vec{b}) = \begin{cases} M_1 + M_2 = 2 \\ M_3 + M_4 = 2 \\ M_2 + M_4 + M_5 = 3. \end{cases}$$

Moreover, we assume that the controller must enforce one specification:

$$M_1 \leq 1.$$

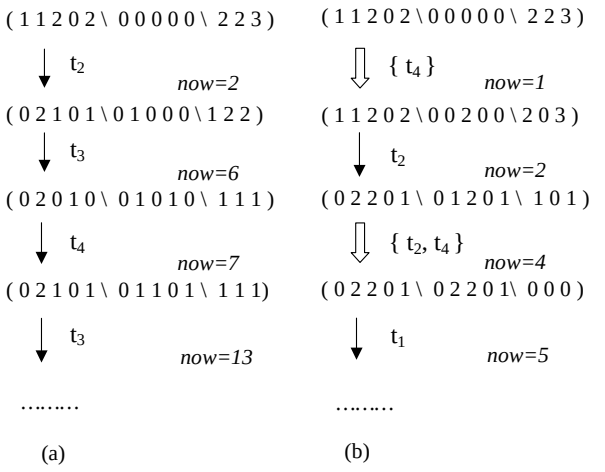

Fig. 5. The behavior of the controlled net in Figure 4: (a) without TTO procedure, (b) when the TTO procedure is used.

Let us first consider the case in which no information on the timing structure is available. In such a case the net never times out and the behavior is that shown in Figure 5.a where we can observe that a partial deadlock occurs. In fact, transition t_2 may initially fire, but in the sequel only t_3 and t_4 may alternately fire. On the contrary, t_1 is always disabled by the controller because the marking of p_1 has not been reconstructed and its firing may potentially violate the specification.

Now, let us assume that the timing structure is known and the TTO procedure is applied. In such a case the partial deadlock can be solved and the net evolution is that shown in Figure 5.b.

At first transition t_1 is disabled by the controller, i.e., $T_n = \{t_1\}$, and $\delta = \delta(t_4) = 1$.

At time $now = 1$ since no transition fires, the TTO procedure is invoked with $T_{to} = \{t_4\}$. The set T_n keeps the same but the marking estimate is improved. In particular, we reconstruct the marking of places p_3 and p_4. It now holds $\delta = \delta(t_2) - now = 1$.

At time $now = 2$, after one more unit of time has elapsed, transition t_2 fires: once again we improve the marking estimate but we still have that $T_n = \{t_1\}$. It now holds $\delta = \delta(t_2) = 2$.

At time $now = 4$ the TTO procedure is applied again with $T_{to} = \{t_2, t_4\}$ and all transitions become control enabled. Note that at this step, when we update the marking estimate, we completely reconstruct the actual marking of the net. ■

VII. Conclusions

In this paper we have dealt with the problem of enforcing a set of GMEC on a timed Petri net by a state feedback control under the assumption that the system state is not measurable but can only be estimated. The use of an estimate instead of the actual marking, may lead to a deadlock even if the controlled system is live. In the case that the net system is structurally bounded, one can use an algorithm that accelerates the state estimation and helps to recover from observer induced deadlocks.

A software architecture implementing the observer-controller and the recovery procedure has been presented and an application example has been discussed. Finally we have also shown that this procedure may allow the net to recover from partial deadlocks.

References

[BAR, 95] K. Barkaoui, I. ben Abdallah, "A deadlock prevention method for a class of FMS," *1995 IEEE Int. Conf. on Systems, Man and Cybernetics*, pp. 4119-4124, (Vancouver, BC, Canada), Oct. 1995.

[BAR, 97] K. Barkaoui, A. Chaoui, B. Zouari, "Supervisory control of discrete event systems using structure theory of Petri nets ," *1997 IEEE Int. Conf. on Systems, Man and Cybernetics* (Orlando, FL, USA), pp. 3750-3755, Oct. 1997.

[BAS, 01] F. Basile, P. Chiacchio, A. Giua, C. Seatzu, "Deadlock recovery of controlled Petri net models using observers," *8th IEEE International Conference on Emerging Technologies and Factory Automation* (Antibes, France), pp. 441–449, Oct. 2001.

[BAS, 02] F. Basile, A. Giua, C. Seatzu, "Petri net control using event observers and timing information," *IEEE 2002 Conference on Decision and Control* (Las Vegas, NV, USA), pp. 787–792, Dec. 2002.

[XIE, 97] F. Chu, X. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming," *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 6, pp. 793–804, 1997.

[EXP, 95] J. Ezpeleta, J.M. Colom, J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems", *IEEE Trans. On Robotics and Automation*, Vol. 11, No. 2, pp. 173–184, 1995.

[GIU, 92] A. Giua, F. DiCesare. M. Silva, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," *Proc. 1992 IEEE Int. Conf. on Systems, Man, and Cybernetics* (Chicago, IL, USA), pp. 974–979, Oct. 1992.

[GIU, 02] A. Giua, C. Seatzu, "Observability of place/transition nets," *IEEE Trans. on Automatic Control*, Vol. 47, No. 9, pp. 1424–1437, Sep. 2002.

[LI, 94] Y. Li, W.M. Wonham, "Control of vector discrete-event systems — part II: controller synthesis," *IEEE Trans. on Automatic Control*, Vol. 39, No. 3, pp. 512–531, 1994.

[MED, 98] M.E. Meda, A. Ramírez, A. Malo, "Identification in discrete event systems," *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics* (San Diego, CA, USA), pp. 740–5, Oct. 1998.

[MUR, 89]] T. Murata, "Petri nets: properties, analysis and applications," *Proc. IEEE*, Vol. Proc. 77, N. 4, pp. 541–580, Apr. 1989.

[PAR, 01] J. Park, S.A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Trans. on Automatic Control*, Vol. 46, No. 10, pp. 1572–1583, 2001.

[PRO, 93] Di Cesare, F., G. Harhalakis, J.M. Proth, M. Silva and F.B. Vernadat, *Practice of Petri nets in manufacturing*, Chapman and Hall, 1993.

[RAM, 00] A. Ramírez–Treviño, I. Rivera–Rangel, E. López–Mellado, "Observer design for discrete event systems modeled by interpreted Petri nets," *2000 IEEE Int. Conf. on Robotics and Automation* (San Francisco, CA, USA) pp. 2871–2876, Apr. 2000.

[YAM, 96] K. Yamalidou, J.O. Moody, M.D. Lemmon, P.J. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, Vol. 32, No. 1, 1996.

[ZHA, 95] L. Zhang, L.E. Holloway, "Forbidden state avoidance in controlled Petri nets under partial observation," *Proc. 33rd Allerton Conf.* (Monticello, IL, USA), pp. 146–155, Oct. 1995.

[DIC, 93] M.C. Zhou, F. DiCesare, *Petri net synthesis for discrete event control of manufacturing systems*, Kluwer Ac., 1993.