Supervisory Controller Design to Enforce Boundedness, Reversibility and Liveness in Systems Modeled by Timed Petri Nets

ALTUĞ İFTAR
Department of Electrical and Electronics Engineering
Eskişehir Technical University
26555 Eskişehir, TURKEY

Abstract: - Supervisory controller design to enforce boundedness, reversibility, and liveness in discrete-event systems modeled by timed Petri nets where both transitions and places are timed is considered. A controller design approach which uses the approach of stretching is proposed. The approach produces a supervisory controller which guarantees boundedness and reversibility simultaneously. The designed controller also guarantees $\mathcal{T}$-liveness for the largest possible subset $\mathcal{T}$ of the set of transitions. Therefore, boundedness, reversibility, and liveness are enforced simultaneously whenever possible. Furthermore, the designed controller is maximally permissive in the sense that no transitions are disabled unnecessarily and the reachability set of the controlled system is the largest possible set in which boundedness and reversibility can be enforced simultaneously. Controller design for an automated manufacturing system example is also presented to demonstrate the proposed approach.

Key-Words: - Discrete Event Systems, Timed Petri Nets, Supervisory Control, Algorithms, Manufacturing Systems, Robotic and Automation Systems

Received: November 17, 2022. Revised: August 23, 2023. Accepted: September 25, 2023. Published: October 16, 2023.

1 Introduction

Discrete-event systems (DES) are very common in manufacturing, robotics, and automation systems. A common formalism to model such systems is Petri nets, [1], [2]. Although Petri nets were first introduced without the notion of time, time may play an important role in many DES. Thus, timed Petri nets (TPNs) were introduced to describe the evolution of these systems throughout time, [3], [4], [5], [6], [7], [8], [9]. A TPN, unlike an untimed Petri net, includes some time-delays, which are usually associated with either transitions or places. A Petri net in which time-delays are associated with transitions is commonly called a *timed-transition Petri net* (TTPN) and a Petri net in which time-delays are associated with places is commonly called a *timed-place Petri net* (TPPN). The transitions and places in a Petri net usually represent *events* and *resources*, respectively. Therefore, when an event takes a certain time, it is natural to associate this time-delay with the transition which represents that event. On the other hand, when the use of a resource takes a certain time, the time-delay must be associated with the place which represents that resource. Therefore, in a system in which both events and use of resources take time, it may be necessary to use a TPN model, where both transitions and places are timed.

To design a controller for any system, representation of the state of the system is, in general, necessary. The state of an untimed Petri net can easily be represented by the *marking vector*. However, the representation of the state of a TPN is much more complicated, [10]. To overcome this difficulty, the method of *stretching* was first introduced for TTPNs in [11], and then for TPPNs in [12]. The case of TPNs where both transitions and places are timed was then considered in [13].

In the present work, as in [11], [12], [13], it is assumed that all the time-delays are commensurate;

i.e., they are an integer multiple of a common divisor. This assumption is justified since in many practical systems a suitable time unit can be found which divides (at least approximately) all the process delays. Therefore, in this work, this unit is taken as the unit of time and all the time-delays are thus represented by an integer. Furthermore, the time variable is also represented by an integer and is denoted by k. Moreover, without loss of generality, the initial time is assumed to be $k = 0$.

Since in any manufacturing and/or automation system, any task requires a minimum time to complete, in this work it is assumed that every transition has a positive time-delay. On the other hand, since some of the resources may become immediately available, some of the places may have zero time-delay.

Supervisory control is, in general, necessary for DES to avoid undesirable behaviour, such as deadlock, [14], or to enforce desirable behaviour, such as boundedness, reversibility, and/or liveness, [15]. Controller design to avoid deadlock in TTPNs and TPPNs were respectively considered in [16], and in [17], and to enforce boundedness, reversibility, and liveness in TTPNs and TPPNs were respectively considered in [18], and in [19]. Deadlock avoidance controller design for TPNs in which both transitions and places are timed was then considered in [13]. In the present work, we consider supervisory controller design to enforce boundedness, reversibility, and liveness in TPNs in which both transitions and places are timed.

Throughout the paper, we use $\mathbf{N}$ and $\mathbf{N}^+$ to denote the sets of natural numbers (with zero) and positive integers, respectively. For two vectors, a and b, of the same dimension, $a \geq b$ ($a \leq b$) means that each element of a is greater (less) than or equal to the corresponding element of b. For any vector a, a^T denotes its transpose. For a set S, $|S|$ denotes the number of elements of S and $[S]_i$ denotes the i^{th} element of S ($i = 1, \ldots, |S|$).

2 Preliminaries

2.1 Timed Petri Nets

A TPN is a tuple $G = (P, T, N, O, M_0, \mathcal{D}_P, \mathcal{D}_T)$, where P is the set of places, T is the set of transitions, $N : P \times T \to \mathbf{N}$ is the *input matrix* that specifies the weights of arcs from places to transitions, $O : P \times T \to \mathbf{N}$ is the *output matrix* that specifies the weights of arcs from transitions to places, $M_0 : P \to \mathbf{N}$ is the *initial marking vector* which specifies the number of tokens at each place at the *initial time* $k = 0$, $\mathcal{D}_P : P \to \mathbf{N}$ is the vector of the *time-delays* of the places, and $\mathcal{D}_T : T \to \mathbf{N}^+$ is the vector of the *time-delays* of the transitions.

Graphically, a TPN can be represented as in Fig. 1, where circles represent the places, bars represent the transitions, and the arrows represent the arcs. An integer next to an arc indicates the weight of that arc, an integer next to a place indicates the time-delay of that place, and an integer next to a transition indicates the time-delay of that transition. However, to keep the graphics simple, when an arc has a unit weight, we do not write its weight. Also, an arc with a zero weight is not shown at all. Similarly, when a place has a zero time-delay, or when a transition has a unit time-delay, we do not write those. The dots inside the circles indicate the tokens present in the corresponding place at the initial time. Therefore, for the example TPN shown in Fig. 1, we have $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2, t_3\}$,

$$N = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \qquad O = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$M_0 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \quad \mathcal{D}_P = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \mathcal{D}_T = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}.$$

In a TPN, a token which enters a place $p \in P$ at time k can be used to enable a downstream transition no earlier than at time $k + \mathcal{D}_P(p)$. A transition $t \in T$ can fire at time k only if it is enabled at time k. The transition $t \in T$ is enabled at time k only if $M(k) \geq N(t)$, where $M(k)$ denotes the number of tokens at each place $p \in P$ at time k, which entered place p at time $k - \mathcal{D}_P(p)$ or earlier, and $N(t)$ denotes the column of N which corresponds to transition t. When a transition $t \in T$ fires at time k, it withdraws $N(p, t)$ tokens from each place $p \in P$ at time k and introduces $O(p, t)$ tokens to each place $p \in P$ at time $k + \mathcal{D}_T(t)$, where $N(p, t)$ and $O(p, t)$ are the elements of, respectively, N and O, which correspond to place p and transition t.

2.2 Stretched Petri Nets

The method of stretching for TPNs in which both transitions and places are timed was introduced in [13], to represent the state of such a TPN efficiently. In this method, given a TPN, another TPN, called the

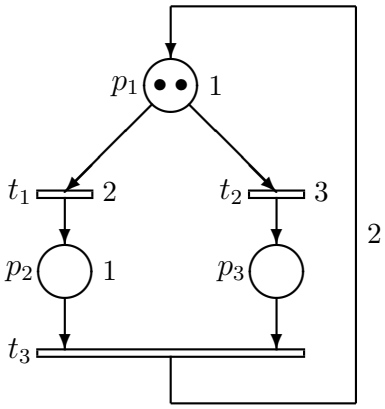

Fig. 1. Example TPN, [13].

stretched Petri net (SPN) is first defined. The SPN has, in general, more places and more transitions than the original TPN. However, each place of the SPN has zero time-delay and each transition has a unit time-delay. To obtain the SPN, $P_1 := \{p \in P \mid \mathcal{D}_P(p) = 0\}$, $P_2 := \{p \in P \mid \mathcal{D}_P(p) \geq 1\}$, $T_1 := \{t \in T \mid \mathcal{D}_T(t) = 1\}$, and $T_2 := \{t \in T \mid \mathcal{D}_T(t) \geq 2\}$ are first defined. Then,

I) for any $p \in P_2$,
   i) $\mathcal{D}_P(p)$ new places, p_1^p, p_2^p, ..., $p_{\mathcal{D}_P(p)}^p$, and $\mathcal{D}_P(p)$ new transitions, t_1^p, t_2^p, ..., $t_{\mathcal{D}(p)}^p$ are defined. Each new place is assigned zero time-delay. The time-delay of p is also redefined as zero. Furthermore, the newly introduced transitions are such that each of them has a unit time-delay and any one of them fires immediately as it becomes enabled.
   ii) all the arcs which leave place p are detached from p; instead, the originating end of these arcs are attached to place $p_{\mathcal{D}_P(p)}^p$. Furthermore, new arcs with unity weights from p to t_1^p, from t_1^p to p_1^p, from p_1^p to t_2^p, ..., and from $t_{\mathcal{D}(p)}^p$ to $p_{\mathcal{D}_P(p)}^p$ are introduced.
   iii) the tokens inside place p (which indicate the *initial marking*) are moved to place $p_{\mathcal{D}_P(p)}^p$.

II) for any $t \in T_2$,
   i) $\mathcal{D}_T(t)-1$ new places, p_1^t, p_2^t, ..., $p_{\mathcal{D}_T(t)-1}^t$, and $\mathcal{D}_T(t)-1$ new transitions, t_1^t, t_2^t, ..., $t_{\mathcal{D}_T(t)-1}^t$ are defined. Each new place is assigned zero time-delay and each new transition is assigned unit time-delay. The time-delay of t is also redefined as unity. Furthermore, the newly introduced transitions are such that any one of them fires immediately as it becomes enabled.
   ii) all the arcs which leave transition t are detached from t; instead, the originating end of these arcs are attached to transition $t_{\mathcal{D}_T(t)-1}^t$. Furthermore, new arcs with unity weights from t to p_1^t, from p_1^t to t_1^t, from t_1^t to p_2^t, ..., and from $p_{\mathcal{D}_T(t)-1}^t$ to $t_{\mathcal{D}_T(t)-1}^t$ are introduced.

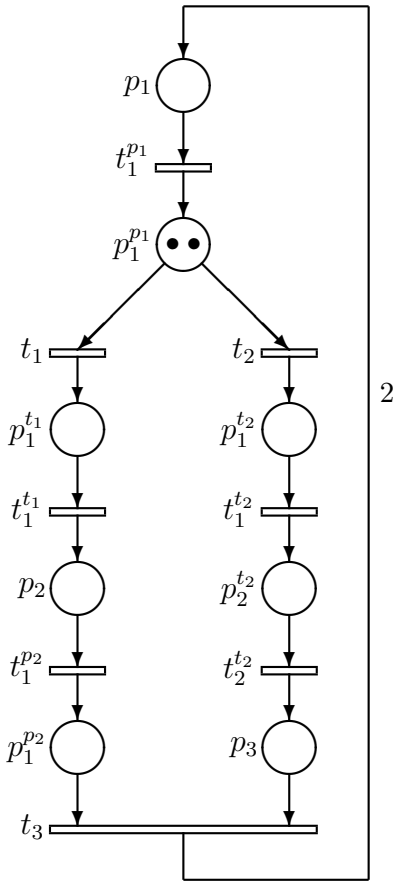

Fig. 2. SPN for the Example TPN, [13].

As a result of this procedure, the SPN shown in Fig. 2 is obtained for the example TPN shown in Fig. 1. The SPN can be defined as a tuple $\hat{G} = (\hat{P}, \hat{T}, \hat{N}, \hat{O}, \hat{M}_0)$. Here, $\hat{P} := P \cup P_n$ is the set of places, where P_n is the set of newly introduced places; $\hat{T} := T \cup T_n$ is the set of transitions, where T_n is the set of newly introduced transitions; $\hat{N} : \hat{P} \times \hat{T} \to \mathbf{N}$ and $\hat{O} : \hat{P} \times \hat{T} \to \mathbf{N}$ are, respectively, the input and the output matrices; and $\hat{M}_0 : \hat{P} \to \mathbf{N}$ is the initial marking vector. Since all the places of the

SPN are delay-free and all the transitions have unit time-delays, there is no need to include any vectors to indicate the time-delays in the SPN definition $\hat{G}$. We refer the reader to [13], for further details.

Although the SPN has, in general, more places and transitions than the original TPN, its state at any time $k \in \mathbf{N}$ is solely and uniquely represented by the marking vector $\hat{M}(k)$ at time k. Furthermore, the state of the SPN can be used to define the state of the original TPN. Moreover, starting with $\hat{M}(0) = \hat{M}_0$, the evolution of the state of the SPN can easily be described by the equation

$$\hat{M}(k+1) = \hat{M}(k) + \sum_{t \in \phi(k)} \left[\hat{O}(t) - \hat{N}(t) \right] \quad (1)$$

where $\phi(k)$ is the set of transitions which fire at time k and $\hat{O}(t)$ and $\hat{N}(t)$ indicate the columns of $\hat{O}$ and $\hat{N}$, respectively, which correspond to the transition t.

2.3 Boundedness, Reversibility and Liveness

The *reachability set*, $R(G)$, of a TPN G is the set of all reachable states from the initial state. Although it is rather complicated to describe the states, and thus the reachability set, of a TPN (e.g., see, [18]), the reachability set, $R(\hat{G})$, of the corresponding SPN $\hat{G}$, can simply be described as the set of all the markings $\hat{M}$, such that there exists a sequence of enabled transitions which lead from the initial marking $\hat{M}_0$ to $\hat{M}$. Furthermore, there is a one-to-one correspondence between the states of $\hat{G}$ and of G, thus, between $R(\hat{G})$ and $R(G)$, [18].

For a *bound vector* $K : P \to \mathbf{N}$, a TPN G is said to be *K-bounded* if $M(p) \leq K(p)$, $\forall p \in P$, for all markings M corresponding to every state in $R(G)$. G is said to be *bounded* if it is K-bounded for some bounded K. G is said to be *reversible* if for every state $S \in R(G)$, there exists a sequence of enabled transitions which lead from S to the initial state. $t \in T$ is said to be *live* if for every state $S \in R(G)$, there exists a sequence of enabled transitions which lead from S to $S' \in R(G)$ such that t is enabled at S'. For $\mathcal{T} \subset T$, G is said to be *$\mathcal{T}$-live* if every $t \in \mathcal{T}$ is live. Finally, G is said to be *live* if it is T-live.

3 Supervisory Controller Design

In this section we consider supervisory controller design to enforce boundedness, reversibility and liveness in a system modeled by a TPN where both transitions and places are timed. Since it is easier to represent the state of the TPN using its SPN, we first design a controller for the SPN and then describe the corresponding controller for the original TPN (OPN). Since there is a one-to-one correspondence between the states of the SPN $\hat{G}$ and of the OPN G, we have the following:

- G is bounded if and only if $\hat{G}$ is bounded.
- G is reversible if and only if $\hat{G}$ is reversible.
- G is live if and only if $\hat{G}$ is live.

Thus, a controller which guarantees boundedness, reversibility and liveness in the SPN also guarantees boundedness, reversibility and liveness in the OPN. Furthermore, such a controller also avoids deadlock, since deadlock can not occur in a live Petri net, [20].

To design a controller for the SPN, first, for a given bound vector $\hat{K} : \hat{P} \to \mathbf{N}$, we determine the *bounded reachability set*, R_B, of $\hat{G}$. The bounded reachability set is the largest subset of $R(\hat{G})$ such that any $\hat{M} \in R_B$ is bounded by $\hat{K}$ (i.e., $\hat{M}(p) \leq \hat{K}(p)$, $\forall p \in \hat{P}$) and can be reached from $\hat{M}_0$ by passing through only states which are bounded by $\hat{K}$. Note that, although $R(\hat{G})$ may include infinitely many elements for an unbounded Petri net, [15], R_B is always a finite set. Algorithm 1, which is obtained by properly modifying Algorithm A in [19], constructs the bounded reachability set, R_B. The function $\tau(m)$, used in Algorithm 1, returns the set of sets of simultaneously enabled transitions at state m and $\rho(m, \phi)$ returns the next state, as given by (1), when ϕ fires at m. Algorithm 1 returns an empty set for R_B if $\hat{M}_0(p) > \hat{K}(p)$, for some $p \in \hat{P}$. In such a case, $\hat{G}$ is not $\hat{K}$-bounded at the initial time and, hence, there exists no controller which can enforce $\hat{K}$-boundedness.

Once the bounded reachability set, $R_B \neq \emptyset$, is constructed as above, we construct the *bounded reversible reachability set*, R_S. This set is the largest subset of R_B such that for any $\hat{M} \in R_S$ either $\hat{M} = \hat{M}_0$ or $\hat{M}$ can be reached from $\hat{M}_0$ and $\hat{M}_0$ can be reached from $\hat{M}$ without passing through any markings outside R_S. Algorithm 2, which is obtained by properly modifying Algorithm B in [19], constructs this set. Besides R_S, this algorithm also returns the set of all live transitions, $\hat{\mathcal{T}}$. Here, functions τ and ρ are as in Algorithm 1 and the function $\epsilon(m)$ returns the set of enabled transitions at state m.

Algorithm 1 : Algorithm to construct R_B

Inputs: $\hat{G} = (\hat{P}, \hat{T}, \hat{N}, \hat{O}, \hat{M}_0)$ and $\hat{K}$
Output: R_B
if $\hat{M}_0 \leq \hat{K}$ **then**
  $R_B = R_1 = \{\hat{M}_0\}$
  loop
    $R_2 = \emptyset$
    for $i = 1$ to $|R_1|$ **do**
      $m = [R_1]_i$
      $\Phi = \tau(m)$
      for $j = 1$ to $|\Phi|$ **do**
        $\phi = [\Phi]_j$, $\hat{m} = \rho(m, \phi)$
        if $\hat{m} \notin (R_2 \cup R_B)$ **and** $\hat{m} \leq \hat{K}$ **then**
          $R_2 \leftarrow R_2 \cup \{\hat{m}\}$
        end if
      end for
    end for
    if $R_2 == \emptyset$ **then**
      exit loop
    end if
    $R_B \leftarrow R_B \cup R_2$
    $R_1 = R_2$
  end loop
else
  $R_B = \emptyset$
end if
return R_B

Once the bounded reversible reachability set, R_S, is determined, a controller which keeps the state of the SPN within R_S guarantees $\hat{K}$-boundedness and reversibility simultaneously for the SPN. Such a controller can be described as

$$c(m, \phi) := \begin{cases} 1\ , & \text{if } \rho(m, \phi) \in R_S \\ 0\ , & \text{otherwise} \end{cases} \quad , \qquad (2)$$

for any $m \in R_S$ and for any $\phi \in \tau(m)$, where $c(m, \phi) = 0$ means that the firing of the set ϕ at state m is disabled by the controller and $c(m, \phi) = 1$ means that the firing of this set at state m is not disabled by the controller. This controller also guarantees liveness, if and only if $\hat{\mathcal{T}} = \hat{T}$, where $\hat{\mathcal{T}}$ is as returned by Algorithm 2. Otherwise, $\hat{\mathcal{T}}$ is the largest subset of $\hat{T}$ such that a controller exists which guarantees $\hat{K}$-boundedness, reversibility, and $\hat{\mathcal{T}}$-liveness simultaneously.

Algorithm 2 : Algorithm to determine R_S and $\hat{\mathcal{T}}$

Inputs: $\hat{G} = (\hat{P}, \hat{T}, \hat{N}, \hat{O}, \hat{M}_0)$ and R_B
Outputs: R_S and $\hat{\mathcal{T}}$
$R_X = R_B$, $\hat{\mathcal{T}} = \emptyset$
Start: $R' = R_X \setminus \{\hat{M}_0\}$, $R_S = \{\hat{M}_0\}$
loop
  $R = R'$, $Flag = 0$
  for $i = 1$ to $|R|$ **do**
    $\bar{m} = [R]_i$, $\Phi = \tau(\bar{m})$
    for $k = 1$ to $|\Phi|$ **do**
      $\phi = [\Phi]_k$, $\hat{m} = \rho(\bar{m}, \phi)$
      if $\hat{m} \in R_S$ **then**
        $R_S \leftarrow R_S \cup \{\bar{m}\}$, $R' \leftarrow R' \setminus \{\bar{m}\}$,
        $Flag = 1$, **go to** Break
      end if
    end for
    Break: **continue**
  end for
  if $Flag == 0$ **then**
    exit loop
  end if
end loop
if $R_S == \{\hat{M}_0\}$ **then**
  go to Finish
end if
$R_R = \emptyset$, $R_a = \{\hat{M}_0\}$
loop
  $R_b = \emptyset$, $R_R \leftarrow R_R \cup R_a$
  for $i = 1$ to $|R_a|$ **do**
    $\bar{m} = [R_a]_i$, $\Phi = \tau(\bar{m})$
    for $j = 1$ to $|\Phi|$ **do**
      $\phi = [\Phi]_j$, $\hat{m} = \rho(\bar{m}, \phi)$
      if $\hat{m} \in R_S$ **and** $\hat{m} \notin R_R \cup R_b$ **then**
        $R_b \leftarrow R_b \cup \{\hat{m}\}$
      end if
    end for
  end for
  if $R_b == \emptyset$ **then**
    exit loop
  end if
  $R_a = R_b$
end loop
if $R_S \neq R_R$ **then**
  $R_X = R_R$, **go to** Start
end if
for $i = 1$ to $|R_S|$ **do**
  $\bar{m} = [R_S]_i$, $\mathcal{T}_1 = \epsilon(\bar{m})$
  for $j = 1$ to $|\mathcal{T}_1|$ **do**
    $t = [\mathcal{T}_1]_j$
    if $\rho(\bar{m}, \{t\}) \in R_S$ **then**
      $\hat{\mathcal{T}} \leftarrow \hat{\mathcal{T}} \cup \{t\}$
    end if
  end for
end for
Finish: **return** R_S , $\hat{\mathcal{T}}$

4 Example

We borrow the automated manufacturing system example presented in [13]. The TPN model of the system is shown in Fig. 1. The corresponding SPN is shown in Fig. 2. We aim to design a controller which enforces $\hat{K}$-boundedness for $\hat{K}(p) = 2$, $\forall p \in \hat{P}$, reversibility, and liveness simultaneously. Algorithm 1 returns the bounded reachability set, R_B, shown in Table 1, where the ordering of the places is as follows: p_1, $p_1^{p_1}$, $p_1^{t_1}$, p_2, $p_1^{p_2}$, $p_1^{t_2}$, $p_2^{t_2}$, p_3. This set is same as the reachability set $R(\hat{G})$, obtained in [13], since the system is already $\hat{K}$-bounded. Algorithm 2 then determines that

$$R_S = \{m_0, m_1, m_2, m_3, m_7, m_8, m_9, m_{13}, \ldots, m_{24}\}$$

and $\hat{\mathcal{T}} = \hat{T}$. Thus, the states m_4, m_5, m_6, m_{10}, m_{11}, and m_{12} are excluded from R_S, since it is not possible to reach to $\hat{M}_0 = m_0$ from any one of these states. The controller described by (2) then disables t_1 at states m_1, m_2, and m_3 and t_2 at states m_7, m_8, and m_9. Thus, a controller which disables firing t_1 at states corresponding to m_1, m_2, and m_3 and t_2 at states corresponding to m_7, m_8, and m_9 in the OPN guarantees boundedness, reversibility and liveness in the OPN. Furthermore, this controller also avoids deadlock, since the controlled system is live.

Table 1. Bounded reachability set, R_B.

$m_0 =$	0	2	0	0	0	0	0	0
$m_1 =$	0	1	1	0	0	0	0	0
$m_2 =$	0	1	0	1	0	0	0	0
$m_3 =$	0	1	0	0	1	0	0	0
$m_4 =$	0	0	1	0	1	0	0	0
$m_5 =$	0	0	0	1	1	0	0	0
$m_6 =$	0	0	0	0	2	0	0	0
$m_7 =$	0	1	0	0	0	1	0	0
$m_8 =$	0	1	0	0	0	0	1	0
$m_9 =$	0	1	0	0	0	0	0	1
$m_{10} =$	0	0	0	0	0	1	0	1
$m_{11} =$	0	0	0	0	0	0	1	1
$m_{12} =$	0	0	0	0	0	0	0	2
$m_{13} =$	0	0	1	0	0	1	0	0
$m_{14} =$	0	0	0	1	0	0	1	0
$m_{15} =$	0	0	0	0	1	0	0	1
$m_{16} =$	0	0	0	1	0	1	0	0
$m_{17} =$	0	0	0	0	1	0	1	0
$m_{18} =$	0	0	0	0	1	1	0	0
$m_{19} =$	0	0	0	0	1	0	1	0
$m_{20} =$	0	0	1	0	0	0	1	0
$m_{21} =$	0	0	0	1	0	0	0	1
$m_{22} =$	0	0	0	0	1	0	0	1
$m_{23} =$	0	0	1	0	0	0	0	1
$m_{24} =$	2	0	0	0	0	0	0	0

5 Conclusions

Supervisory controller design to enforce boundedness, reversibility, and liveness in DES modeled by TPNs where both transitions and places are timed has been considered. A controller design approach which uses the approach of stretching has been proposed. The approach produces a supervisory controller which guarantees boundedness and reversibility simultaneously. The designed controller also guarantees $\mathcal{T}$-liveness for the largest possible subset $\mathcal{T}$ of the set of transitions. Therefore, boundedness, reversibility, and liveness are enforced simultaneously whenever possible. Furthermore, the designed controller is *maximally permissive* in the sense that no transitions are disabled unnecessarily and the reachability set of the controlled system is the largest possible set in which boundedness and reversibility can be enforced simultaneously.

The algorithms proposed for controller design terminates in finite time. The computational complexity of Algorithm 1 is proportional to the size of R_B, which is related to both the size of the Petri net and the bound vector. The computational complexity of Algorithm 2, on the other hand, is at worst proportional to the square of the size of R_B. Therefore, the overall computational complexity of the proposed design approach is bounded by the square of the size of R_B.

The proposed controller design uses the so-called *behavioral approach*, [20], [21], as opposed to the so-called *structural approach*, [22], [23]. Although behavioral approach is more tedious (as it requires the construction of the reachability set), it can guarantee the design of maximally permissive controllers; whereas the structural approach, in general, produces conservative controllers, [22].

As discussed in [24], to model some DES, it may be necessary to use Petri net models in which the arcs are timed. A Petri net in which arcs are timed can be called a *timed-arc Petri net* (TAPN). Therefore, as a future study, the present approach can be extended to TAPNs. Furthermore, the present approach can also be extended to decentralized supervisory controller design along the lines of [25].

References

[1] M. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Norwell, MA: Kluwer Academic Publishers, 1993.

[2] J. Proth and X. Xie, *Petri Nets: A Tool for Design and Management of Manufacturing Systems*. West Sussex: John Wiley & Sons, 1996.

[3] F. D. J. Bowden, "A brief survey and synthesis of the roles of time in Petri nets," *Mathematical and Computer Modelling*, vol. 31, pp. 55–68, 2000.

[4] J. Wang, *Timed Petri Nets: Theory and Application*. Boston, MA: Kluwer Academic, 1998.

[5] W. M. Zuberek, "Timed Petri nets in modeling and analysis of cluster tools," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 562–575, 2001.

[6] L. Popova-Zeugmann, *Time and Petri Nets*. New York: Springer, 2013.

[7] L. Liang, F. Basile, and Z. Li, "A reduced computation of state space to enforce GMECs and deadlock-freeness on TPN systems," in *IFAC PapersOnLine 53-4*, pp. 166–172, 2020.

[8] S. Akshay, L. Helouet, and R. Phawade, "Combining free choice and time in Petri nets," *Journal of Logical and Algebraic Methods in Programming*, vol. 101, 2020.

[9] D. Lefebvre and C. Daoui, "Control design for bounded partially controlled TPNs using timed extended reachability graphs and MDP," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 50, no. 6, pp. 2273–2283, 2020.

[10] C. Lakos and L. Petrucci, "Modular state space exploration for timed Petri nets," *International Journal on Software Tools for Technology Transfer*, vol. 9, pp. 393–411, 2007.

[11] A. Aybar and A. İftar, "Supervisory controller design for timed Petri nets," in *Proceedings of the IEEE International Conference on System of Systems Engineering*, (Los Angeles, CA, U.S.A.), pp. 59–64, Apr. 2006.

[12] A. Aybar and A. İftar, "Representation of the state of timed-place Petri nets using stretching," in *Proceedings of the 4th IFAC Workshop on Discrete-Event System Design*, (Playa de Gandia, Spain), pp. 79–84, Oct. 2009.

[13] A. İftar, "Supervisory control of manufacturing systems modeled by timed Petri nets," in *Proceedings of the 12th IFAC Workshop on Intelligent Manufacturing Systems*, (Austin, TX, U.S.A.), pp. 128–132, Dec. 2016.

[14] Z. W. Li, M. C. Zhou, and N. Q. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics–Part C*, vol. 38, pp. 173–188, 2008.

[15] A. Aybar, A. İftar, and H. Apaydın-Özkan, "Centralized and decentralized supervisory controller design to enforce boundedness, liveness, and reversibility in Petri nets," *International Journal of Control*, vol. 78, pp. 537–553, 2005.

[16] A. Aybar and A. İftar, "Deadlock avoidance controller design for timed Petri nets using stretching," *IEEE Systems Journal*, vol. 2, pp. 178–188, 2008.

[17] A. Aybar and A. İftar, "Supervisory controller design for timed-place Petri nets," *Kybernetika*, vol. 48, pp. 1114–1135, 2012.

[18] A. Aybar and A. İftar, "Supervisory controller design to enforce some basic properties in timed-transition Petri nets using stretching," *Nonlinear Analysis: Hybrid Systems*, vol. 6, pp. 712–729, 2012.

[19] A. Aybar and A. İftar, "Supervisory controller design to enforce basic properties in timed-place Petri nets," in *Preprints of the 6th IFAC Conference on Management and Control of Production and Logistics*, (Fortaleza, Brazil), pp. 486–492, Sept. 2013.

[20] R. S. Sreenivas, "On the existence of supervisory policies that enforce liveness in discrete-event dynamic systems modelled by controlled Petri nets," *IEEE Transactions on Automatic Control*, vol. 42, pp. 928–945, 1997.

[21] H. Boucheneb, A. Alger, and G. Berthelot, "Towards a simplified building of time Petri nets reachability graph," in *Proceedings of the 5th International Workshop on Petri Nets and Performance Models*, (Reims, France), pp. 46–55, May 1993.

[22] M. V. Iordache and P. J. Antsaklis, "Design of $\mathcal{T}$-liveness enforcing supervisors in Petri nets," *IEEE Transactions on Automatic Control*, vol. 48, pp. 1962–1974, 2003.

[23] Z. W. Li and M. C. Zhou, "Elemetary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics–Part A*, vol. 34, pp. 38–51, 2004.

[24] A. Aybar and A. İftar, "Supervisory control of discrete-event systems modeled by timed-arc Petri nets," in *Proceedings of the European Control Conference*, (Saint Petersburg, Russia), pp. 656–661, May 2020.

[25] A. Aybar and A. İftar, "Overlapping decompositions and expansions of Petri nets," *IEEE Transactions on Automatic Control*, vol. 47, pp. 511–515, 2002.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding

This work has been supported by the Scientific Research Projects Commission of Eskişehir Technical University under grant numbers 22ADP301 and 23ADP033.

Conflicts of Interest

The author has no conflicts of interest to declare that are relevant to the content of this article.