Decentralized Control of Petri Nets

Marian V. Iordache and Panos J. Antsaklis[*][†]

Abstract

Supervision based on place invariants (SBPI) is an efficient technique for the supervisory control of Petri nets. In this paper we propose extensions of the SBPI to a decentralized control setting. In our setting, a decentralized supervisor consists of local supervisors, each controlling and observing a part of the Petri net. We consider both versions of decentralized control, with communication, and with no communication. In the case of communication, a local supervisor may receive observations of events that are not locally observable and send enabling decisions concerning events that are not locally controllable. In the first part of the paper we propose efficient algorithms for the design of decentralized supervisors, based on the extension of the SBPI concept of admissibility that we define. Then, in the second part of the paper, we propose the design of decentralized supervisors based on transformations to admissible constraints. The feasibility of this problem is demonstrated with a simple integer programming approach. This approach can incorporate communication between local supervisors as well as communication constraints.

1 Introduction

The decentralized control of discrete event systems (DES) has received considerable attention in the recent years [13]. The current research effort has been focused on the automata setting, and has considered both versions of decentralized control, with communication and with no communication. This paper considers the decentralized control of Petri nets by means of the supervision based on place invariants (SBPI) [5, 11, 20].

Petri nets are compact models of concurrent systems, as they do not represent explicitly the state space of the system. Petri net methods relying on the structure of the net rather than the state space are of special interest, as the size of the state space, when finite, can be exponentially related to the size of the net. Among such methods, the SBPI offers an efficient technique for the design of supervisors enforcing on Petri nets a particular class of state predicates, called *generalized mutual exclusion constraints.* Note that the generalized mutual exclusion constraints can represent any state predicate of a *safe*[1] Petri net [20]. Furthermore, without loss of any of its benefits, the SBPI has been extended in [7] to handle any constraints that can be enforced by control (monitor) places. While SBPI has been considered so far in a centralized setting, this paper proposes extensions of SBPI to a decentralized setting.

Admissibility is a key concept in the SBPI of Petri nets with uncontrollable and unobservable transitions. When dealing with such Petri nets, the SBPI approach classifies the specifications as admissible and inadmissible, where the former can be directly enforced, and the latter are first

[*]Department of Electrical Engineering, University of Notre Dame, IN 46556, USA. E-mail: iordache.1, antsaklis.1@nd.edu.

[†]The partial support of the Lockheed Martin Corporation, of the National Science Foundation (NSF ECS99-12458), and of DARPA/IXO-NEST Program (AF-F30602-01-2-0526) is gratefully acknowledged.

[1]A Petri net is safe if for all reachable markings no place has more than one token.

transformed to an admissible form and then enforced. In the automata setting [12], admissibility corresponds to controllability and observability, and the transformation to an admissible form to the computation of a controllable and observable sublanguage.

The main contributions of this paper are as follows. First, we define *d-admissibility* (decentralized admissibility), as an extension of admissibility to the decentralized setting. Our concept of d-admissibility extends the admissibility concept in the sense that a set of constraints that is d-admissible can be *directly* enforced via SBPI in a decentralized setting. Since d-admissibility identifies constraints for which the supervisors can be easily computed, rather than the class of constraints for which supervisors can be computed, it does not parallel controllability and coobservability in the automata setting [15]. Second, we show how to enforce *d-admissible* constraints and show how to check whether a constraint is d-admissible. Third, to deal with constraints that are not d-admissible, we provide an algorithmic approach to make the constraints d-admissible by enabling communication of events (transition firings). Fourth, to deal with the case in which the constraints are not d-admissible and communication is restricted or unavailable, we propose a simple linear integer programming approach for the design of the decentralized control. The design process generates both the local supervisors and the communication policy. Communication enables the local supervisors to observe events that are not locally observable and to control events that are not locally controllable. The communication policy specifies for each local supervisor the events it remotely observes and the events it remotely controls. This approach allows communication constraints to be incorporated in the design process and can be used to minimize the communication. With regard to our use of integer programming, note that while the development of alternative methods that are less computationally intensive are a direction for future research, in the automata setting it was shown that a decentralized solution cannot be found with polynomial complexity [13]. Note also that the size of the integer program depends on the size of the Petri net structure, and not on the size of its state space (i.e. the size of its equivalent automaton), which may not be finite.

To our knowledge, the decentralized supervisory control of Petri nets has not been yet considered in the literature, except for [3]. In [3], distributed supervisors and a central coordinator are designed for specifications that are given from the beginning in a distributed form. In our approach there is no central coordinator and the specifications are not required to be given in a distributed form. In the automata setting, the work on decentralized control can be found in [13] and the references therein. In particular, we mention [15] for the decentralized control with no communication and [1, 14] for decentralized control with communication. As in our paper, the communication in [14] consists of events rather than states estimates or observation strings [1, 18]. Other related work includes [18], which approaches the problem of finding a decentralized solution with the same performance as a centralized solution when communication is available. The vast majority of the decentralized control papers consider language specifications. In this paper we focus our attention on the particular class of state predicate specifications supported by SBPI. In the automata setting, the existence of a decentralized solution enforcing state predicates is studied in [17]. Literature on SBPI or closely related to it is found in [5, 20, 11, 16, 8, 9] and the references therein. Finally, note that the decentralized control of DES can be used in various applications, including manufacturing [10, 3], failure detection [2], and communication protocols [4].

The paper is organized as follows. Section 2 describes the notation and outlines the SBPI. Section 3 describes the decentralized setting of our approach. Section 4 defines the d-admissibility, shows how d-admissibile constraints can be enforced, and presents the algorithm checking whether a constraint is d-admissible. Then, d-admissibility is applied to the design of local supervisors with communication in section 5. The algorithm presented in section 5 uses communication in order to reduce (when possible) the enforcement of constraints that are not d-admissible to the enforcement of d-admissible constraints. Section 6 describes the supervisory approach for the enforcement of constraints that are not d-admissible in the case in which the communication is restricted or not available. Finally, section 7 illustrates our approach on a manufacturing example from [10].

2 Preliminaries

A Petri net structure is denoted by $\mathcal{N} = (P, T, F, W)$, where P is the set of places, T the set of transitions, F the set of transition arcs, and W the weight function. The incidence matrix of $\mathcal{N}$ is denoted by D (places correspond to rows and transitions to columns). A place (transition) denoted by p_j (t_i) is the place (transition) corresponding to the j'th (i'th) row (column) of the incidence matrix.

The specification of the SBPI [5, 11, 20] consists of the state constraints

$$L\mu \leq b \tag{1}$$

where $L \in \mathbb{Z}^{n_c \times |P|}$, $b \in \mathbb{Z}^{n_c}$, and μ is the marking of $\mathcal{N}$. To distinguish between the case $n_c = 1$ and $n_c > 1$, we say that (1) represents *a constraint* when $n_c = 1$, and that (1) represents *a set of constraints* when $n_c > 1$. Note that $\mathcal{N}$ represents the **plant**. The SBPI provides a supervisor in the form of a Petri net $\mathcal{N}_s = (P_s, T, F_s, W_s)$ with

$$D_s = -LD \tag{2}$$
$$\mu_{0,s} = b - L\mu_0 \tag{3}$$

where D_s is the incidence matrix of the supervisor, $\mu_{0,s}$ the initial marking of the supervisor, and μ_0 is the initial marking of $\mathcal{N}$. The places of the supervisor are called **control places**. The supervised system, that is the **closed-loop** system, is a Petri net of incidence matrix:

$$D_c = \begin{bmatrix} D \\ -LD \end{bmatrix} \tag{4}$$

An example is shown in Figure 7, in which the control places C_1 and C_2 enforce $\mu_2 + \mu_3 \leq 1$ and $\mu_5 + \mu_6 \leq 1$, respectively.

Note that (3) implies that when the plant and the supervisor are in closed-loop, the initial marking of the plant satisfies (1). Let μ_c be the marking of the closed-loop, and let $\mu_c|_{\mathcal{N}}$ denote μ_c restricted to the plant $\mathcal{N}$. Let $t \in T$ be a transition. t is **closed-loop enabled** if μ_c enables t. t is **plant-enabled**, if $\mu_c|_{\mathcal{N}}$ enables t in $\mathcal{N}$. The supervisor **detects** t if t is closed-loop enabled at some reachable marking μ_c and firing t changes the marking of some control place. The supervisor **controls** t if there is a reachable marking μ_c such that t is plant-enabled but not closed-loop enabled. Given μ_c, the supervisor **disables** t if there is a control place C such that $(C, t) \in F_s$ and $\mu_c(C) < W_s(C, t)$.

In Petri nets with uncontrollable and unobservable transitions, admissibility issues arise. Indeed, a supervisor generated as shown above may include control places preventing plant-enabled uncontrollable transitions to fire, and may contain control places with marking varied by firings of closed-loop enabled unobservable transitions. Such a supervisor is clearly not implementable. We say that a supervisor is admissible, if it only controls controllable transitions, and it only detects observable transitions. The constraints $L\mu \leq b$ are **admissible** if the supervisor defined by (2–3) is admissible. When inadmissible, the constraints $L\mu \leq b$ are transformed (if possible) to an admissible form $L_a\mu \leq b_a$ such that $L_a\mu \leq b_a \Rightarrow L\mu \leq b$ [11]. Then, the supervisor enforcing $L_a\mu \leq b_a$ is admissible, and enforces $L\mu \leq b$ as well. Our discussion on admissibility is carried out in more detail in section 4. We will denote $\mathcal{N}$ with sets of uncontrollable and unobservable transitions T_{uc} and T_{uo} by $(\mathcal{N}, T_{uc}, T_{uo})$.

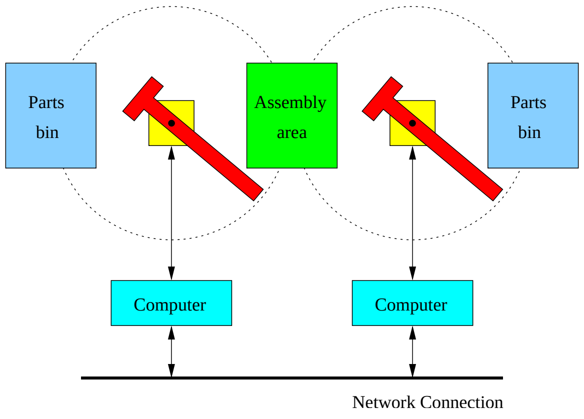

Figure 1: Robotic manufacturing system.

3 The Model

We assume that the system is given as a Petri net structure $\mathcal{N} = (P, T, F, W)$. A decentralized supervisor consists of a set of local supervisors $\mathcal{S}_1$, $\mathcal{S}_2$,... $\mathcal{S}_n$, each acting upon individual parts of the system, called *subsystems*, where the simultaneous operation of the local supervisors achieves a global specification. A local supervisor $\mathcal{S}_i$ observes the system through the set of locally observable transitions $T_{o,i}$, and controls it through the set of locally controllable transitions $T_{c,i}$. So, from the viewpoint of $\mathcal{S}_i$, the sets of uncontrollable and unobservable transitions are $T_{uc,i} = T \setminus T_{c,i}$ and $T_{uo,i} = T \setminus T_{o,i}$. This is the design problem: *Given a global specification and the sets of uncontrollable and unobservable transitions $T_{uc,1}$, $T_{uc,2}$, ... $T_{uc,n}$ and $T_{uo,1}$, $T_{uo,2}$, ... $T_{uo,n}$, find a set of local supervisors $\mathcal{S}_1$, $\mathcal{S}_2$,... $\mathcal{S}_n$ whose simultaneous operation guarantees that the global specification is satisfied, where each $\mathcal{S}_i$ can control $T \setminus T_{uc,i}$ and observe $T \setminus T_{uo,i}$.* A system $\mathcal{N}$ with subsystems of uncontrollable and unobservable transitions $T_{uc,i}$ and $T_{uo,i}$ will be denoted by $(\mathcal{N}, T_{uc,1}, \ldots T_{uc,n}, T_{uo,1}, \ldots T_{uo,n})$.

As an illustration, consider a manufacturing example in which two robots transport parts to a common assembly area (Figure 1). The system is modeled by the Petri net of Figure 2(a), where $\mu_2 = 1$ ($\mu_4 = 1$) when the left (right) robot is in the parts bin, and $\mu_1 = 1$ ($\mu_3 = 1$) when the left (right) robot is in the assembly area. The set of controllable transitions of the left (right) subsystem may be taken as $T_{c,1} = \{t_1, t_2\}$ ($T_{c,2} = \{t_3, t_4\}$). Assume that the subsystem of each robot knows when the other robot enters or leaves the parts bin. Then each subsystem contains the controllable transitions of the other subsystem as observable transitions; a possible graphical representation of the subsystems is shown in Figure 2(b) and (c).

4 Admissibility

To distinguish between admissibility in the centralized case and admissibility in the decentralized case (to be defined in this section), we denote by **c-admissibility** the admissibility property in the centralized case. Therefore, *c-admissibility* is taken with respect to a Petri net $(\mathcal{N}, \mu_0)$ of uncontrollable transitions T_{uc} and unobservable transitions T_{uo}. The significance of c-admissibility is as follows. A c-admissible set of constraints (1) can be implemented with the simple construction of (2–3), as in the fully controllable and observable case.

In the decentralized case, we are interested to define admissibility with respect to a Petri net $(\mathcal{N}, \mu_0)$, and the sets of uncontrollable and unobservable transitions of the subsystems: $T_{uc,1}$ ...

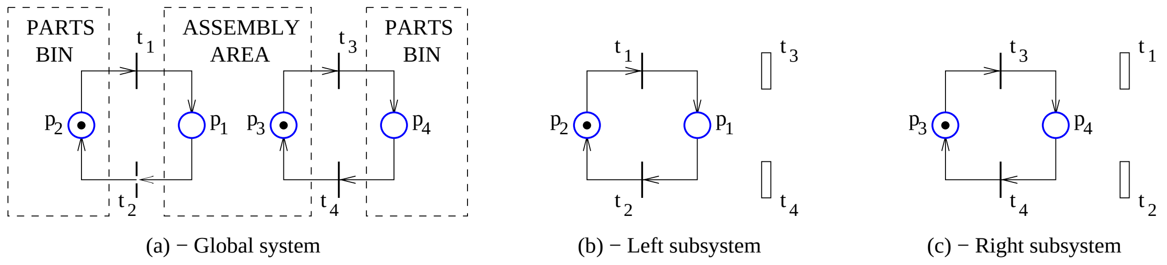

Figure 2: A Petri net model of the robotic manufacturing system.

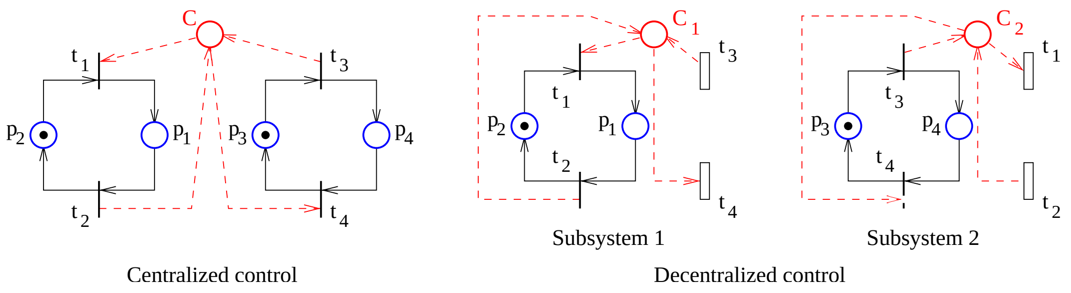

Figure 3: Centralized control versus decentralized control.

$T_{uc,n}$ and $T_{uo,1} \ldots T_{uo,n}$. Admissibility in the decentralized case is called **d-admissibility**. As in the case of c-admissibility, we would like d-admissibility to guarantee that the (decentralized) supervisor can be easily constructed. This is achieved by the following definition.

Definition 4.1 *A constraint is* **d-admissible** *with respect to* $(\mathcal{N}, \mu_0, T_{uc,1} \ldots T_{uc,n}, T_{uo,1} \ldots T_{uo,n})$, *if there is a collection of subsystems* $\mathcal{C} \subseteq \{1, 2, \ldots n\}$, $\mathcal{C} \neq \emptyset$, *such that the constraint is c-admissible with respect to* $(\mathcal{N}, \mu_0, T_{uc}, T_{uo})$, *where* $T_{uc} = \bigcap_{i \in \mathcal{C}} T_{uc,i}$ *and* $T_{uo} = \bigcup_{i \in \mathcal{C}} T_{uo,i}$. *A set of constraints is* **d-admissible** *if each of its constraints is d-admissible.*

To illustrate the definition, assume that we have a constraint that is c-admissible only with respect to the first subsystem. Then, it is d-admissible, as we can select $\mathcal{C} = 1$. Note also that when each subsystem has full observability of the net and every transition is controllable with respect to some subsystem, any constraint is d-admissible.

The construction of a decentralized supervisor, given a d-admissible set of constraints, is illustrated on the Petri net of Figure 2. The mutual exclusion constraint

$$\mu_1 + \mu_3 \leq 1 \tag{5}$$

is to be enforced. The centralized control solution is shown in Figure 3. In the case of decentralized supervision, there are two subsystems: the first one has $T_{uo,1} = \emptyset$ and $T_{uc,1} = \{t_3, t_4\}$, and the other has $T_{uo,2} = \emptyset$ and $T_{uc,2} = \{t_1, t_2\}$. Note that (5) is not c-admissible with respect to any of $(\mathcal{N}, T_{uc,1}, T_{uo,1})$ or $(\mathcal{N}, T_{uc,2}, T_{uo,2})$. However, it is d-admissible for $\mathcal{C} = \{1, 2\}$. Given two variables $x_1, x_2 \in \mathbb{N}$, a decentralized supervisor $\mathcal{S}_1 \wedge \mathcal{S}_2$ enforcing (5) can be defined by the following rules:

The supervisor $\mathcal{S}_1$:

- initialize x_1 to 0.
- disable t_1 if $x_1 = 0$
- increment x_1 if t_2 or t_3 fires.
- decrement x_1 if t_1 or t_4 fires.

The supervisor $\mathcal{S}_2$:

- initialize x_2 to 0.
- disable t_4 if $x_2 = 0$
- increment x_2 if t_2 or t_3 fires.
- decrement x_2 if t_1 or t_4 fires.

A graphical representation of $\mathcal{S}_1$ and $\mathcal{S}_2$ is possible, as shown in Figure 3. Thus, $\mathcal{S}_1$ is represented by C_1 and $\mathcal{S}_2$ by C_2; x_1 is the marking of C_1 and x_2 the marking of C_2. Graphically, C_1 and C_2 are copies of the control place C of the centralized supervisor. Note that (C_1, t_4) and (C_2, t_1) model observation, not control. This is due to the fact that $\mathcal{S}_1$ never disables t_4 and $\mathcal{S}_2$ never disables t_1. As C_1 and C_2 have the same initial marking as C, their markings stay equal at all times. So, whenever t_1 should be disabled, the disablement action is implemented by C_1, and whenever t_4 is to be disabled, the disablement action is implemented by C_2.

In the general case, the construction of a supervisor enforcing a d-admissible constraint $l\mu \leq c$ ($l \in \mathbb{N}^{1\times|P|}$ and $c \in \mathbb{N}$) is as follows: (Note that the notation of Definition 4.1 is used)

Algorithm 4.2 *Supervisor Design for a D-admissible Constraint*

1. Let μ_0 the initial marking of $\mathcal{N}$, C the control place of the centralized SBPI supervisor $\mathcal{N}_s = (P_s, T, F_s, W_s)$ enforcing $l\mu \leq c$, and $\mathcal{C}$ the set of Definition 4.1.
2. For all $i \in \mathcal{C}$, let $x_i \in \mathbb{N}$ be a state variable of $\mathcal{S}_i$.
3. Define $\mathcal{S}_i$, for $i \in \mathcal{C}$, by the following rules:
   - Initialize x_i to $c - l\mu_0$.
   - If $t \in T_{c,i}$, $t \in C\bullet$ and $x_i < W_s(C,t)$, then $\mathcal{S}_i$ disables t.
   - If t fires, $t \in T_{o,i}$ and $t \in \bullet C$, then $x_i = x_i + W_s(t, C)$.
   - If t fires, $t \in T_{o,i}$ and $t \in C\bullet$, then $x_i = x_i - W_s(C, t)$.

To enforce a d-admissible set of constraints $L\mu \leq b$, the construction above is repeated for each constraint $l\mu \leq c$. Note that in the graphical representation the supervisors $\mathcal{S}_i$ correspond to $|\mathcal{C}|$ copies of the control place C of the centralized supervisor, where each copy has the same initial marking as C.

Next we prove that the resulting decentralized supervisor is feasible (physically implementable) and as performant as the centralized supervisor. The decentralized supervisor is **feasible** if for all reachable markings μ_c of the closed-loop and for all transitions t: (i) for all $i = 1 \ldots n$, if t is closed-loop enabled and $t \notin T_{o,i}$, firing t does not change the state (marking) of $\mathcal{S}_i$; (ii) if t is plant-enabled but not closed-loop enabled, there is an $\mathcal{S}_i$ disabling t such that $t \in T_{c,i}$.

Theorem 4.3 *The decentralized supervisor constructed in Algorithm 4.2 is feasible, enforces the desired constraint, and is as permissive as the centralized supervisor of* $(\mathcal{N}, T_{uc}, T_{uo})$.

Proof: Feasibility is an immediate consequence of the construction of Algorithm 4.2. To prove the remaining part of the theorem, we show that a firing sequence σ is enabled by the centralized supervisor at the initial marking if and only if it is enabled by the decentralized supervisor at the initial marking. The proof uses the notation of the Algorithm 4.2 and of Definition 4.1. In addition, let $\mathcal{S}$ be the centralized supervisor implemented by the control place C, and $\mathcal{S}_d$ the decentralized supervisor $\bigwedge_{i \in C} \mathcal{S}_i$. Given a firing sequence $\sigma = t_{i_1} t_{i_2} \ldots t_{i_k}$ enabled from μ_0 in the open-loop $(\mathcal{N}, \mu_0)$, let's denote by μ_j the markings reached while firing σ: $\mu_0 \xrightarrow{t_{i_1}} \mu_1 \xrightarrow{t_{i_2}} \mu_2 \xrightarrow{t_{i_3}} \ldots \mu_k$.

First, note that for all firing sequences $\sigma = t_{i_1} t_{i_2} \ldots t_{i_k}$ enabled by both $\mathcal{S}$ and $\mathcal{S}_d$ from μ_0, we have that at all markings μ_j reached while firing σ

$$x_i = c - l\mu_j \quad \forall i \in \mathcal{C} \tag{6}$$

This is proven by induction. For $i = 0$, (6) is satisfied, due to the way the variables x_i are initialized. Assume (6) satisfied for $j < k$. According to the SBPI, when the plant has the marking μ_j the marking of C is $c - l\mu_j$, the same as x_i $\forall i \in \mathcal{C}$. There are two cases: (a) $l\mu_j = l\mu_{j+1}$ and (b) $l\mu_j \neq l\mu_{j+1}$. In case (a), in view of (2), $W_s(C, t_{i_j}) = W_s(t_{i_j}, C) = 0$. Therefore, neither the marking of C, nor any of the x_i's is changed by firing t_{i_j}. Hence, (6) is satisfied at μ_{j+1}. In case (b), note that by Definition 4.1, the d-admissibility of $l\mu \leq c$ implies that $\mathcal{S}$ is c-admissible with respect to $(\mathcal{N}, \mu_0, T_{uc}, T_{uo})$. Then, since t_{i_j} is not dead and $l\mu_j \neq l\mu_{j+1}$: $t_{i_j} \notin T_{uo}$. However, $t_{i_j} \notin T_{uo} \Rightarrow (\forall i \in \mathcal{C})\ t_{i_j} \notin T_{uo,i}$. Hence, t_{i_j} is observable to all $\mathcal{S}_i$, and so all x_i are changed in the same way. Moreover, according to the SBPI, firing t_{i_j} changes the marking of C the same way as x_i are changed. From the SBPI we know that the new marking of C is $c - l\mu_{j+1}$. It follows that when μ_{j+1} is reached, $x_i = c - l\mu_{j+1}$ $\forall i \in \mathcal{C}$.

Finally, we prove by contradiction that the firing sequences enabled by $\mathcal{S}$ from μ_0 are the firing sequences enabled by $\mathcal{S}_d$ from μ_0. Assume the contrary, that there is σ that is enabled by one supervisor and not enabled by the other. We decompose σ into $\sigma = \sigma_x t_x \sigma_y$, $t_x \in T$, where σ_x is enabled by both supervisors and $\sigma_x t_x$ is not. If $\mu_0 \xrightarrow{\sigma_x} \mu_x$, then (6) is satisfied at $\mu_j = \mu_x$; the marking of C is also $c - l\mu_x$. There are two cases: (a) t_x enabled by C; (b) t_x not enabled by C. As in the previous part of the proof, case (a) leads to the conclusion that $\mathcal{S}_d$ enables also t_x, which contradicts the assumption that not both $\mathcal{S}$ and $\mathcal{S}_d$ enable t_x. In case (b), according to the SBPI, we have that $W_s(C, t_x) < c - l\mu_x$ and $t_x \notin T_{uc}$, by the d-admissibility of $l\mu \leq c$. It follows that there is $i \in \mathcal{C}$ such that $\mathcal{S}_i$ disables t_x, and hence that $\mathcal{S}_d$ does not enable t_x. This contradicts the assumption that one of $\mathcal{S}$ and $\mathcal{S}_d$ enables t_x. □

Let $\mathcal{S}$ be the centralized supervisor that enforces the constraint in the fully controllable and observable version of $\mathcal{N}$. Let T_{uo}^M be the set of transitions that are *not* detected by $\mathcal{S}$ and T_{uc}^M the set of transitions that are *not* controlled by $\mathcal{S}$. The d-admissibility of a constraint can be tested as follows:

Algorithm 4.4 *Checking whether a Constraint is D-admissible*

1. Find T_{uo}^M and T_{uc}^M.

2. Find the largest set of subsystems $\mathcal{C}$ such that $\forall i \in \mathcal{C}$: $T_{uo,i} \subseteq T_{uo}^M$.

3. If $\mathcal{C} = \emptyset$, declare that the constraint is not d-admissible and exit.

4. Define $T_{uc} = \bigcap_{i \in \mathcal{C}} T_{uc,i}$.

5. Does T_{uc} satisfy $T_{uc} \subseteq T_{uc}^M$? If yes, declare the constraint d-admissible. Otherwise, declare that the constraint is not d-admissible.

Note that a d-admissible constraint can be implemented for a minimal set $\mathcal{C}_{min} \subseteq \mathcal{C}$ containing the minimal number of subsystems such that $T_{uc}^M \supseteq \bigcap_{i\in\mathcal{C}_{min}} T_{uc,i}$. Note also that checking whether a set of constraints is d-admissible involves checking each constraint individually.

Proposition 4.5 *The algorithm checking d-admissibility is correct.*

Proof: A constraint is declared d-admissible if $\mathcal{C} \neq \emptyset$ and $T_{uc} \subseteq T_{uc}^M$. The definition of T_{uo}^M and T_{uc}^M implies that the constraint is c-admissible with respect to $(\mathcal{N}, T_{uc}, T_{uo})$ (where $T_{uo} = \bigcup_{i\in\mathcal{C}} T_{uo,i}$). Then, in view of Definition 4.1, the algorithm is right to declare the constraint d-admissible.

Next, assume a d-admissible constraint. Then, there is a set of subsystems $\mathcal{C}' \neq \emptyset$ such that the constraint is c-admissible with respect to $(\mathcal{N}, T'_{uc}, T'_{uo})$ (where $T'_{uc} = \bigcap_{i\in\mathcal{C}'} T_{uc,i}$ and $T'_{uo} = \bigcup_{i\in\mathcal{C}'} T_{uc,i}$). Then $T'_{uo} \subseteq T_{uo}^M$; $T'_{uo} \subseteq T_{uo}^M \Rightarrow \mathcal{C}' \subseteq \mathcal{C} \Rightarrow T_{uc} \subseteq T'_{uc} \Rightarrow T_{uc} \subseteq T_{uc}^M$. Consequently, the algorithm declares the constraint to be d-admissible. □

In general, it may be difficult to compute the sets T_{uc}^M and T_{uo}^M. Then estimates $T_{uc}^e \subseteq T_{uc}^M$ and $T_{uo}^e \subseteq T_{uo}^M$ can be used in the algorithm instead. In this case the algorithm only checks a sufficient condition for d-admissibility, and so it can no longer detect constraints that are not d-admissible. In the case of the SBPI, such estimates can be found from the structural admissibility test of [11], stating that $L\mu \leq b$ is c-admissible if $LD_{uc} \leq 0$ and $LD_{uo} = 0$, where D_{uc} and D_{uo} are the restrictions of D to the columns of T_{uc} and T_{uo}.

Note that when it is possible and convenient to communicate in a reliable fashion with each subsystem of a decentralized system, a centralized solution with $T_{uc} = \bigcap_{i=1\ldots n} T_{uc,i}$ and $T_{uo} = \bigcap_{i=1\ldots n} T_{uo,i}$ is possible. Finally, note that in the implementation of d-admissible constraints, each supervisor $\mathcal{S}_i$ with $i \in \mathcal{C}$ relies on the proper operation of the other supervisors $\mathcal{S}_j$ with $j \in \mathcal{C}$. By itslef, a local supervisor may not be able to implement a d-admissible constraint or its implementation may be overrestrictive. For instance, in the example of Figure 2, the supervisor of the first subsystem can only enforce $\mu_1+\mu_3 \leq 1$ by itself by enforcing $\mu_1 = 0$. However, this solution is overrestrictive. D-admissibility illustrates the fact that more can be achieved when supervisors cooperate to achieve a given task, rather than when a supervisor tries on its own to achieve it (cf. "two heads better than one" in [15]).

5 Supervision with Communication

The design technique introduced in this section uses communication to reduce the set of unobservable transitions $T_{uo,i}$ such that, if possible, the given constraints are c-admissible with respect to $(\mathcal{N}, T_{uc}, T_{uo})$. Note that communication cannot reduce T_{uo} below the attainable lower bound $T_{uo,L} \subseteq T_{uo}$, where $T_{uo,L} = \bigcap_{i=1\ldots n} T_{uo,i}$. T_{uc} can be changed by selecting a different set $\mathcal{C}$. However, it cannot be reduced below $T_{uc,L} = \bigcap_{i=1\ldots n} T_{uc,i}$. Indeed, $T_{uc,L}$ ($T_{uo,L}$) is the set of transitions uncontrollable (unobservable) in all subsystems.

As an illustration, consider the system of Figure 2(a), this time with $T_{c,1} = T_{o,1} = \{t_1, t_2\}$ and $T_{c,2} = T_{o,2} = \{t_3, t_4\}$. In this setting the constraint $\mu_1+\mu_3 \leq 1$ is cleary not d-admissible. However, by communicating the firings of the transitions t_1 and t_2 to the right subsystem and of t_3 and t_4 to the left subsystem, the sets of locally observable transitions become $T'_{o,1} = T'_{o,2} = \{t_1, t_2, t_3, t_4\}$, the constraint becomes d-admissible, and the supervisory solution of Figure 3 can be again used. For the general case, we have the following algorithm:

Algorithm 5.1 *Decentralized Supervisor Design*

1. Is the specification admissible with respect to $(\mathcal{N}, T_{uc,L}, T_{uo,L})$? If not, transform it to be admissible (an approach of [11] could be used).

2. Let $\mathcal{S}$ be the centralized SBPI supervisor enforcing the specification. Let T_c be the set of transitions controlled by $\mathcal{S}$ and T_o the set of transitions detected by $\mathcal{S}$.

3. Find a set $\mathcal{C}$ such that $T_{uc} = \bigcap_{i\in\mathcal{C}} T_{uc,i} \subseteq T \setminus T_c$.[2]

4. Design the decentralized supervisor by applying Algorithm 4.2 to $\mathcal{N}$ and $\mathcal{C}$.

5. The communication can be designed as follows: for all $t \in T_o \cap (\bigcup_{i\in\mathcal{C}} T_{uo,i})$, a subsystem j such that $t \in T_{o,j}$ transmits the firings of t to all supervisors $\mathcal{S}_k$ with $t \in T_{uo,k}$ and $k \in \mathcal{C}$.

Note the following. First, no communication arises when $T_o \cap (\bigcup_{i\in\mathcal{C}} T_{uo,i}) = \emptyset$. Second, the algorithm does not take in account communication limitations, such as bandwidth limitations of the communication channel. Bandwidth limitations can be considered in the approach described in the next section. Third, in this solution communication is used only to make some locally unobservable transitions observable; there is no remote control of locally uncontrollable transitions. Fourth, this solution tends to require less communication than a centralized solution. Indeed, a central supervisor not only needs to send the control decisions to the local subsystems, but also to remotely observe *all* transitions in T_o. Fifth, the main limitation of the algorithm is that in the case of inadmissible specifications, the transformation at the step 1 may result in constraints that are too restrictive. If so, the alternative solution proposed in the next section can be used. Finally, the only way the algorithm can fail is at step 1, when the specification is inadmissible and the transformations to an admissible form fail.

Proposition 5.2 *The decentralized supervisor is feasible and equally permissive to the centralized supervisor $\mathcal{S}$ enforcing the specification on $(\mathcal{N}, T_{uc}, T_{uo,L})$.*

Proof: Since $\mathcal{S}$ is admissible, $T_c \cap T_{uc} = \emptyset$ and $T_o \cap T_{uo,L} = \emptyset$. Communication ensures that the sets of locally unobservable transitions become $T'_{uo,i} = T_{uo,i} \setminus T_o$. It follows that the specification is d-admissible with respect to $(\mathcal{N}, T_{uc,1}, \ldots T_{uc,n}, T'_{uo,1}, \ldots T'_{uo,n})$ and so the conclusion follows by Theorem 4.3. □

6 Constraint Transformations

6.1 Control without communication

In this section we propose a method for the transforamtion of constraints that are not d-admissible to (stronger) constraints that are d-admissible. As an illustration, consider the Petri net of Figure 2(a), this time with the initial marking $\mu_0 = [0, 3, 0, 3]^T$, $T_{c,1} = T_{o,1} = \{t_1, t_2\}$, $T_{c,2} = T_{o,2} = \{t_3, t_4\}$, and the desired constraint $\mu_1 + \mu_3 \le 2$. It can be seen that there is no way to transform $\mu_1 + \mu_3 \le 2$ to a single d-admissible constraint. However, we can transform it to two d-admissble

[2]At least one solution exists, $\mathcal{C} = \{1 \ldots n\}$. This can be seen from the fact that $\mathcal{S}$ admissible w.r.t. $(\mathcal{N}, T_{uc,L}, T_{uo,L})$ implies $T_{uc,L} \cap T_c = \emptyset$, and from $T_{uc,L} = \bigcap_{i=1\ldots n} T_{uc,i}$.

Marian Iordache, Panos J. Antsaklis, “Decentralized Control of Petri Nets,” Proceedings of the Workshop on Discrete Event Systems Control at the 24th International Conference on Application and Theory of Petri Nets (ATPN 2003), pp.143-158, Eindhoven, The Netherlands, 24 June, 2003.

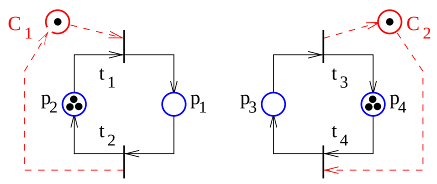

Figure 4: Decentralized control example.

constraints: $\mu_1 \leq 1$ and $\mu_3 \leq 1$, where the first is d-admissible for $\mathcal{C} = \{1\}$ and the second for $\mathcal{C} = \{2\}$. This solution is shown in Figure 3.

In the general case, the problem can be stated as follows: *Given a set of constraints $L\mu \leq b$ that is not d-admissible and the subsystem clusters $\mathcal{C}_1$, $\mathcal{C}_2$, ... $\mathcal{C}_m$, find sets of constraints $L_1\mu \leq b_1$ ... $L_m\mu \leq b_m$ d-admissible with respect to $\mathcal{C}_1$, $\mathcal{C}_2$, ... $\mathcal{C}_m$, respectively, such that*

$$(L_1\mu \leq b_1 \wedge L_2\mu \leq b_2 \wedge \ldots L_m\mu \leq b_m) \Rightarrow L\mu \leq b \tag{7}$$

Note that unlike to the algorithm of section 5, which computes also a set $\mathcal{C}$, here we have a number of sets $\mathcal{C}_1$, $\mathcal{C}_2$, ... $\mathcal{C}_m$ that are already given. For instance, if $T_{o,i} \cap T_{o,j} = \emptyset$ for all $i \neq j$, then we may take $\mathcal{C}_i = \{i\}$, for all i. Furthermore, note that this framework includes the case when not all constraints $L_i\mu \leq b_i$ are necessary to implement $L\mu \leq b$, by allowing $L_i = 0$ and $b_i = 0$.

The problem is more tractable if we replace (7) with the stronger condition below:

$$\left[\sum_{i=1}^{m} \alpha_i L_i \mu \leq \sum_{i=1}^{m} \alpha_i b_i\right] \Rightarrow L\mu \leq b \tag{8}$$

where α_i are nonnegative scalars.[3] Without loss of generality, (8) assumes that L_1 ... L_m have the same number of rows. Again, without loss of generality, (8) can be replaced by

$$\left[\sum_{i=1}^{m} L_i \mu \leq \sum_{i=1}^{m} b_i\right] \Rightarrow L\mu \leq b \tag{9}$$

We further simplify our problem to

$$L_1 + L_2 + \ldots L_m = R_1 + R_2 L \tag{10}$$
$$b_1 + b_2 + \ldots b_m = R_2(b + \mathbf{1}) - \mathbf{1} \tag{11}$$

for R_1 with nonnegative integer elements and R_2 diagonal with positive integers on the diagonal. Note that $[(R_1 + R_2L)\mu \leq R_2(b + \mathbf{1}) - \mathbf{1}] \Rightarrow L\mu \leq b$ has been proved in [11].

It is known that a sufficient condition for the c-admissibility of a set of constraints $L\mu \leq b$ is that $LD_{uc} \leq 0$ and $LD_{uo} = 0$, where D_{uc} and D_{uo} are the restrictions of the incidence matrix D to the sets of uncontrollable and unobservable transitions [11]. The admissibility requirements in our setting can then be written as

$$L_i D_{uc}^{(i)} \leq 0 \tag{12}$$
$$L_i D_{uo}^{(i)} = 0 \tag{13}$$

[3]In the literature, a relaxation of a hard problem that is similar to the relaxation from (7) to (8) is the S-procedure mentioned in [19] at page 62.

where $D_{uc}^{(i)}$ and $D_{uo}^{(i)}$ are the restrictions of D to the sets $T_{uc}^{(i)} = \bigcap_{i \in \mathcal{C}_i} T_{uc,i}$ and $T_{uo}^{(i)} = \bigcup_{i \in \mathcal{C}_i} T_{uo,i}$.

Integer programming can be used to find a feasible solution to (10–13), where the unknowns are R_1, R_2, L_i, and b_i. In general it is difficult to find constraints or a cost function that guarantee that the least restrictive solution is found, when a least restrictive solution exists. However, given a finite set $\mathcal{M}_I$ of markings of interest, it is possible to insure that the feasible space of the solution will include the markings of $\mathcal{M}_I$ by using the constraints:

$$L_i M \leq b_i \mathbf{1}^T \qquad i = 1 \dots m \tag{14}$$

where $\leq$ means that each element of $L_i M$ is less or equal to the element of the same indices in $b_i \mathbf{1}^T$, M is a matrix whose columns are the markings of $\mathcal{M}_I$, and $\mathbf{1}^T$ is a row vector of appropriate dimension in which all elements are 1. The next result is an immediate consequence of our previous considerations.

Proposition 6.1 *Any sets of constraints $L_i \mu \leq b_i$ satisfying (10–14) are d-admissible and $\bigwedge_{i=1 \dots n} [L_i \mu \leq b_i] \Rightarrow L\mu \leq b$.*

6.2 Control with communication

This section extends the procedure of section 6.1 to the case in which communication is possible. Here communication is used to relax the admissibility constraints (12) and (13) by reducing the number of locally uncontrollable or unobservable transitions. However, this reduction may be limited by various communication constraints, such as bandwidth limitations. The framework of this section allows communication constraints to be incorporated in the design process, and can be used to minimize communication by defining a cost function. The analysis of this section, without being comprehensive, serves as an illustration of the fact that such problems can be approached in this framework.

For each set $\mathcal{C}_i$ and transition t_j, let α_{ij} be a binary variable, where $\alpha_{ij} = 1$ if the firing of t_j is made known to the subsystems in $\mathcal{C}_i$. Note that we have the following constraints:

$$\forall t_j \in T_{uo,L} : \alpha_{ij} = 0 \tag{15}$$

where $T_{uo,L} = \bigcap_{i=1 \dots n} T_{uo,i}$ is the set of transitions that cannot be observed anywhere in the system.

Let B_L^i and B_U^i be lower and upper bounds of $L_i D$ and $A = [\alpha_{ij}]$ be the matrix of elements α_{ij}. Given a vector v and a matrix M, let $diag(v)$ denote the diagonal matrix of diagonal v, $M(k, \cdot)$ the k'th row of M, and $M|_{T_{uo}^{(i)}}$ the restriction of M to the transitions of $T_{uo}^{(i)}$ (i.e. $M|_{T_{uo}^{(i)}}$ contains the columns $M(\cdot, j)$ such that $t_j \in T_{uo}^{(i)}$). We require

$$L_i D_{uo}^{(i)} \leq [B_U^i diag(A(i, \cdot))]|_{T_{uo}^{(i)}} \tag{16}$$
$$L_i D_{uo}^{(i)} \geq [B_L^i diag(A(i, \cdot))]|_{T_{uo}^{(i)}} \tag{17}$$

instead of $L_i D_{uo}^{(i)} = 0$. In this way, the admissibility requirement $L_i D_{uo}^{(i)} = 0$ is relaxed by eliminating the constraints corresponding to the transitions of $T_{uo}^{(i)}$ that have their firings communicated to the subsystems of $\mathcal{C}_i$.

Similarly, (12) can also be relaxed by communicating enabling decisions of supervisors. Naturally, for each transition t it controls, each supervisor $\mathcal{S}_i$ has two enabling decisions: *enable* and *disable*. They depend on whether all control places C of $\mathcal{S}_i$ that are connected to t satisfy $\mu_c(C) \geq W_s(C,t)$ or not. A possible implementation is to require a supervisor announce a remote actuator each time its enabling decision changes. Then the actuator can determine its enabling by taking the conjunction of the decisions corresponding to all supervisors controlling it. In our setting, d-admissibility implies that the supervisors within a cluster $\mathcal{C}_i$ have always the same enabling decisions, and so only communication between clusters needs to be considered. Similarly to α_{ij}, we can introduce binary variables ε_{ij} describing the communication of enabling decisions pertaining to t_j. Thus, $\varepsilon_{ij} = 1$ if a supervisor from $\mathcal{C}_i$ communicates its enabling decisions to t_j. As in the case of α_{ij}, we have

$$\forall t_j \in T_{uc,L} : \varepsilon_{ij} = 0 \tag{18}$$

for $T_{uc,L} = \bigcap_{i=1\dots n} T_{uc,i}$. Furthermore, if $E = [\varepsilon_{ij}]$, (12) becomes:

$$L_i D_{uc}^{(i)} \leq [B_U^i diag(E(i,\cdot))]|_{T_{uc}^{(i)}} \tag{19}$$

Communication constraints stating that certain transitions cannot be observed by communication or that certain transitions cannot be remotely controlled by communication, can be incorporated by setting coefficients α_{ij} and ε_{ij} to zero. Constraints limiting the average network traffic can be incorporated as constraints of the form:

$$\sum_i A(i,\cdot)g_i + \sum_i E(i,\cdot)h_i \leq p \tag{20}$$

where g_i and h_i are vectors of appropriate dimensions and p is a scalar. As an example, the elements of g_i could reflect average firing counts of the transitions over the operation of the system. Note that (20) can be written more compactly as

$$Tr(AG + EH) \leq p \tag{21}$$

where G and H are the matrices of columns g_i and h_i, and $Tr(M)$ denotes the trace of a matrix M.

We may also choose to minimize the amount of communication involved in the system. Then we can formulate our problem as

$$\min_{L_i, b_i, A, E, R_1, R_2} Tr(AC + EF) \tag{22}$$

where the weight matrices C and F are given, and the minimization is subject to the constraints (10–11), (14), (15–19), and $\alpha_{ij}, \varepsilon_{ij} \in \{0,1\}^{|T|}$. This problem can be solved using linear integer programming.

6.3 Liveness Constraints

A difficulty of this approach is that the permissivity of the generated constraints can be hard to control. In the worst case, the generated constraints may cause parts of the system to unavoidably deadlock. Such a situation can be prevented by using a special kind of constraints, that we call liveness constraints.

A liveness constraint consists of a vector x such that for all i: $L_i x \leq 0$. A possible way to obtain such constraints is described next. Given a finite firing sequence σ, let x_σ be a vector such that $x_\sigma(i)$ is the number of occurrences of the transition t_i in σ. Given the Petri net of incidence matrix D and the constraints $L\mu \leq b$, let y be a nonnegative integer vector such that $Dy \geq 0$ and $-LDy \geq 0$. A vector y satisfying these inequalities has the following property. If σ is a firing

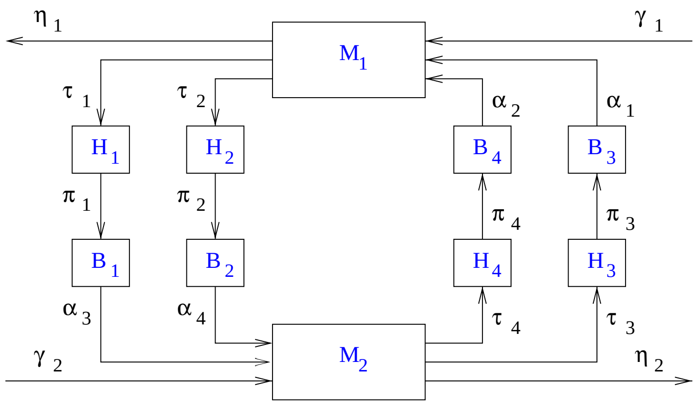

Figure 5: A manufacturing system.

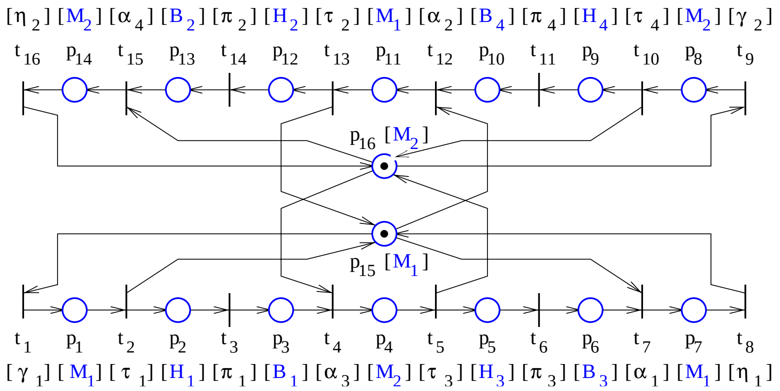
[η 2] [M2] [α 4] [B 2] [π 2] [H2] [τ 2] [M1] [α 2] [B 4] [π 4] [H4] [τ 4] [M2] [γ 2]
t16 p14 t15 p13 t14 p12 t13 p11 t12 p10 t11 p9 t10 p8 t9
p16 [M2]
p15 [M1]
t1 p1 t2 p2 t3 p3 t4 p4 t5 p5 t6 p6 t7 p7 t8
[γ 1] [M1] [τ 1] [H1] [π 1] [B 1] [α 3] [M2] [τ 3] [H3] [π 3] [B 3] [α 1] [M1] [η 1]

Figure 6: Petri net model of the system.

sequence such that (a) σ can be fired without violating $L\mu \leq b$ and (b) $x_\sigma = y$, then σ can be fired infinitely often without violating $L\mu \leq b$. However, if the decentralized control algorithm generates a constraint $L_i\mu \leq b_i$ such that $L_iDy \not\leq 0$, then any firing sequence σ having $x_\sigma = y$ cannot be infinitely often fired in the closed-loop. If such a situation is undesirable, the matrices L_i can be required to satisfy $L_ix \leq 0$ for $x = Dy$. An illustration will be given in section 7.

7 Example

This section illustrates our approach on the manufacturing example from [10], shown in Figure 5. The system consists of two machines (M_1 and M_2), four robots (H_1 ... H_4), and four buffers of finite capacity (B_1 ... B_4). The events associated with the movement of the parts within the system are marked with Greek letters. There are two types of parts. The manufacturing process of the first type of parts is represented by the following sequence of events: $\gamma_1\tau_1\pi_1\alpha_3\tau_3\pi_3\alpha_1\eta_1$. The manufacturing process of the second kind of parts is represented by $\gamma_2\tau_4\pi_4\alpha_2\tau_2\pi_2\alpha_4\eta_2$. These processes can be represented by the Petri net of Figure 6. In the Petri net, the transitions are labeled by the events they represent, and the places by the names of the manufacturing components. For instance, a token in p_{16} indicates that M_2 is idle, and a token in p_8 indicates that M_2 is working on a part of type 2 that has just entered the system. Furthermore, the number of parts in a buffer

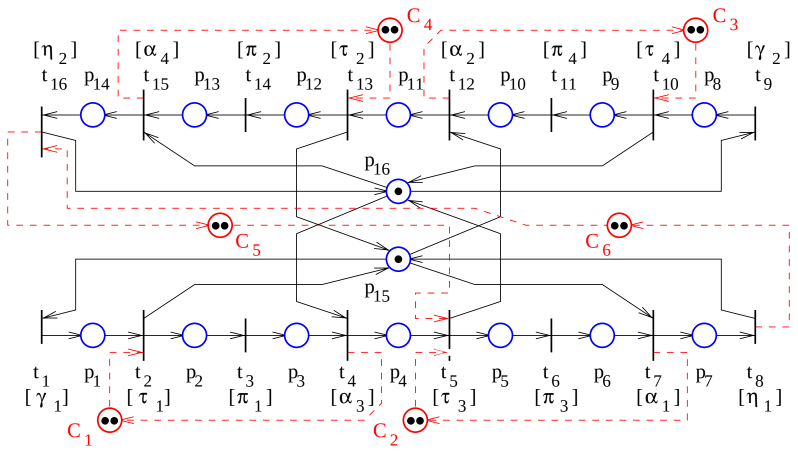

Figure 7: Decentralized supervision.

is the marking of the place modeling the buffer; for instance, μ_{13} represents the number of parts in B_2 at the marking μ. The number of parts the machines M_1 and M_2 can process at the same time is $\mu_1 + \mu_7 + \mu_{11} + \mu_{15} = n_1$ and $\mu_4 + \mu_8 + \mu_{14} + \mu_{16} = n_2$, respectively. In [10], $n_1 = n_2 = 1$.

The first supervisory requirements are that the buffers do not overflow. Assuming that the buffers B_1 and B_2 share common space, the requirement can be written as:

$$\mu_3 + \mu_{13} \leq 2k \tag{23}$$

where $2k$ is the maximum number of parts that can be in B_1 and B_2 at the same time. Similarly, if the buffers B_3 and B_4 share a common space of the same capacity, the constraint is

$$\mu_6 + \mu_{10} \leq 2k \tag{24}$$

Another requirement is that the number of completed parts of type 1 is about the same as the number of completed parts of type 2:

$$v_8 - v_{16} \leq u \tag{25}$$

$$v_{16} - v_8 \leq u \tag{26}$$

where v_8 and v_{16} denote the number of firings of t_8 and t_{16}, respectively. In [10], $u = 2$. Note that constraints involving the vector v can be easily represented as marking constraints in a transformed Petri net [7].

Following [10], the constraints (23–24) are enforced assuming that the system consists of the subsystems: $T_{c,1} = \{t_2\}$ and $T_{o,1} = \{t_2, t_3, t_4\}$, $T_{c,2} = \{t_5\}$ and $T_{o,2} = \{t_5, t_6, t_7, t_8\}$, $T_{c,3} = \{t_{10}\}$ and $T_{o,3} = \{t_{10}, t_{11}, t_{12}\}$, $T_{c,4} = \{t_{13}, t_{16}\}$ and $T_{o,4} = \{t_{13}, t_{14}, t_{15}, t_{16}\}$. We take $\mathcal{C}_i = \{i\}$ for $i = 1 \ldots 4$. Enforcing (23–24) for $k = 2$ results in the control places C_1, C_2, C_3, and C_4 shown in Figure 7. They correspond to the subsystems 1, 2, 3 and 4, respectively, and enforce $\mu_2 + \mu_3 \leq 2$, $\mu_5 + \mu_6 \leq 2$, $\mu_9 + \mu_{10} \leq 2$, and $\mu_{12} + \mu_{13} \leq 2$.

In order to enforce (25–26), we need communication of events. Indeed, without communication there is no acceptable solution. For instance, a solution is to enforce $\mu_5 + \mu_6 + \mu_7 + v_8 \leq u$ in subsytem 2 and $v_{16} \leq u$ in subsystem 4. However, this implies that the manufacturing system is constrained to produce no more than $2u$ parts! To force a computer implementation to generate constraints that do not block the system, we can introduce liveness constraints. In this example, we can add the liveness constraints $L_i x \leq 0$ for $x = Dy$ and $y = [1, 1, \dots 1]^T$. This is to prevent the constraints generated by the algorithm from blocking the firing sequence $t_1 t_2 \dots t_{16}$ to occur infinitely often. However, with this liveness constraint and no communication, the problem is infeasible. Therefore, since communication is necessary, we are interested to minimize it. Assuming broadcast ($\alpha_{ij} = \alpha_j$, $\varepsilon_{ij} = \varepsilon_j$, for all i) and that the cost of remote control and remote observation is nonzero and equal (i.e. $A = E$), the following is an optimal solution:

$$\mu_5 + \mu_6 + \mu_7 + v_8 - v_{16} \leq 2 \tag{27}$$
$$v_{16} - v_8 \leq 2 \tag{28}$$

which involves communicating the occurrences of t_8 and t_{16}. The constraint (27) is implemented in the subsystem 2, and the constraint (28) in the subsystem 4. In Figure 7, the two constraints are enforced by the control places C_5 and C_6.

Finally, note that in general a computer implementation may generate solutions that are too restrictive, such as $\mu_{12} + \mu_{13} + \mu_{14} + v_{16} - v_8 \leq 2$ instead of $v_{16} - v_8 \leq 2$. Then, a second integer program can be used to improve the permissivity, by minimizing the sum of the positive coefficients of the constraints, while requiring the other coefficients to stay less or equal to zero (the integer program is also subject to the constraints of the previous integer program and to a constraint that fixes the communication cost to the minimal value previously computed.)

8 Conclusions

The design of decentralized supervisors is computationally easy for the class of specifications identified as d-admissible. When communication between the local supervisors is allowed, the concept of d-admissibility can also be used for the design of supervisors enforcing specifications that are not d-admissible, by the identification of the events that need to be communicated. In the decentralized settings with no communication or with restricted communication, the enforcement of specifications that are not d-admissible can be done via linear integer programming. The integer programming approach is suboptimal, as it may not produce the least restrictive solution, when it exists. However, it allows to design both the supervisors and their communication policy. Moreover, it can be used to minimize the communication of the local supervisors.

References

[1] G. Barrett and S. Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Transactions on Automatic Control*, 45(9):1620–1638, 2000.

[2] R. Boel and J. van Schuppen. Decentralized failure diagnosis with costly communication between diagnosers. In *Proceedings of the 6th International Workshop on Discrete Event systems*, pages 175–181, 2002.

[3] H. Chen and H. Baosheng. Distributed control of discrete event systems described by a class of controlled Petri nets. In *Preprints of IFAC International Symposium on Distributed Intelligence Systems*, 1991.

[4] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varayia. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[5] A. Giua, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 974–979, 1992.

[6] M. V. Iordache and P. J. Antsaklis. Decentralized control of DES using Petri nets. Technical report isis-2002-005, University of Notre Dame, September 2002.

[7] M. V. Iordache and P. J. Antsaklis. Synthesis of supervisors enforcing general linear vector constraints in Petri nets. In *Proceedings of the 2002 American Control Conference*, pages 154–159, 2002.

[8] Y. Li and W. Wonham. Control of Vector Discrete-Event Systems I - The Base Model. *IEEE Transactions on Automatic Control*, 38(8):1214–1227, 1993.

[9] Y. Li and W. Wonham. Control of Vector Discrete-Event Systems II - Controller Synthesis. *IEEE Transactions on Automatic Control*, 39(3):512–530, 1994.

[10] F. Lin and W. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, 35(12):1330–1337, 1990.

[11] J. O. Moody and P. J. Antsaklis. Petri net supervisors for DES with uncontrollable and unobservable transitions. *IEEE Transactions on Automatic Control*, 45(3):462–476, 2000.

[12] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[13] K. Rudie. The current state of decentralized discrete-event control systems. In *Proceedings of the 10th Mediteranean Conference on Control and Automation*, 2002.

[14] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event system. In *Proceedings of the 1999 American Control Conference*, pages 1965–1970, 1999.

[15] K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[16] G. Stremersch. *Supervision of Petri Nets*. Kluwer Academic Publishers, 2001.

[17] S. Takai and S. Kozama. Decentralized state feedback control of discrete event systems. *Systems & Control Letters*, 22(5):369–375, 1994.

[18] J.H. van Schuppen. Decentralized supervisory control with information structures. In *Proceedings of the International Workshop on Discrete Event Systems (WODES98)*, pages 36–41, 1998.

[19] L. Vandenberge and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[20] E. Yamalidou, J. O. Moody, P. J. Antsaklis, and M. D. Lemmon. Feedback control of Petri nets based on place invariants. *Automatica*, 32(1):15–28, 1996.