49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

An Iterative Control Method for Distributed Continuous Petri nets

Hanife Apaydin-Özkan, Cristian Mahulea, Jorge Júlvez, Manuel Silva

Abstract— **In this paper we study a reachability control problem for distributed systems modeled by timed continuous Petri nets. In particular, we generalize our previous works in this framework where we solved this problem only for distributed systems composed by two subsystems. We will show how the existing algorithm can be adapted in order to be able to deal with distributed systems composed by a finite number of subsystems.**

I. Introduction

The complexity of nowadays systems entails the consideration of more and more autonomous modes of operation for complex distributed plants. Each distributed plant is equipped with *local controllers* and *local observers* resulting in a networked embedded system that consists of computer nodes (or subsystems) and of a communication network connecting the various subsystems.

In this paper we approach the problem of building local controllers for large scale distributed discrete event systems. The problem has been studied before using automata [1], [2], [3] and discrete Petri nets in [4], [5], [6], [7]. Recently, a new formalism has been introduced in order to tackle the state-explosion problem frequently appearing in discrete event systems with high populations. It is called *continuous Petri nets* and represents approximations of the discrete Petri nets. In this relaxed framework many problems have a reduced computational complexity since many *integer linear programming* problems become *linear programming* problems. Even if the control problem on the continuous approximation may look easier than the one on the original discrete net, in many cases the system is distributed and the controller cannot have access to all subsystems. In this case, a distributed controller should be considered.

In this work we consider distributed timed continuous Petri nets with infinite server semantics (DcontPN) [8]. Such a system is composed by several continuous Petri net subsystems that model different parts of a plant, interconnected by buffers modelled by places. These buffers contain the products produced by a given subsystem and needed by another subsystem. As a simple example let us consider a car manufacturing factory composed by two plants A and D in two different cities. The Petri net model is given in Fig. 1. The plant A produces the car body (place p_1) and then sends it to the plant D (place p_2). The plant A can produce concurrently a limited number of car bodies (the initial marking of p_3). In plant D, the engine is constructed (p_4) and then it is put in an intermediate buffer p_5. The same plant paints the body received from plant A in p_6 and puts it in p_7 to be assembled together with the engine. The firing of t_8 means the production of a new car. We assume that D can produce concurrently a limited number of engines (initial marking of p_8) and can paint a limited car bodies in parallel (initial marking of p_9). Place p_b is the buffer containing the car bodies produced by plant A while p_a is the buffer containing the finished products. Since we do not want to produce more than we sell, the plant A starts to produce a new body (firing of t_1) only when a car is sold.

We will first consider the problem of reaching a target marking in a DcontPN. The underlying idea of the strategy is to design a local controller for each subsystem. The controllers should be computed offline, and it is assumed that the structure and state of a given subsystem is unknown to the controllers located at other subsystems. In order not to overload the communication network, it is desirable to interchange as few information as possible between controllers. Each controller computes a control law to reach the target state of its subsystem, and asks the other controllers to produce enough resources in the buffers to execute its control actions. This paper mainly focuses on reaching the target markings of all subsystems in parallel after a finite amount of time. Once the control law for each subsystem is obtained, several control strategies can be used to implement it, see for instance [9].

II. Distributed Continuous Petri nets

The reader is assumed to be familiar with basic Petri net concepts (see [10] for a gentle introduction).

***Definition** 2.1:* A continuous Petri net system is a pair $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ where $\mathcal{N} = \langle P, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$ is a net structure where: (i) P and T are the sets of places and transitions respectively; (ii) $\boldsymbol{Pre}, \boldsymbol{Post} \in \mathbb{R}_{\geq 0}^{|P| \times |T|}$ are the pre and post

This work has been partially supported by the European Community's Seventh Framework Programme under project DISC (Grant Agreement n. INFSO-ICT-224498) and by CICYT - FEDER projects DPI2006-15390, DPI2010-20413 and TIN2007-66523.

The authors are with the the Aragón Institute of Engineering Research (I3A), University of Zaragoza, Maria de Luna 1, 50018 Zaragoza, Spain {hapaydin, cmahulea, julvez, silva@unizar.es}.

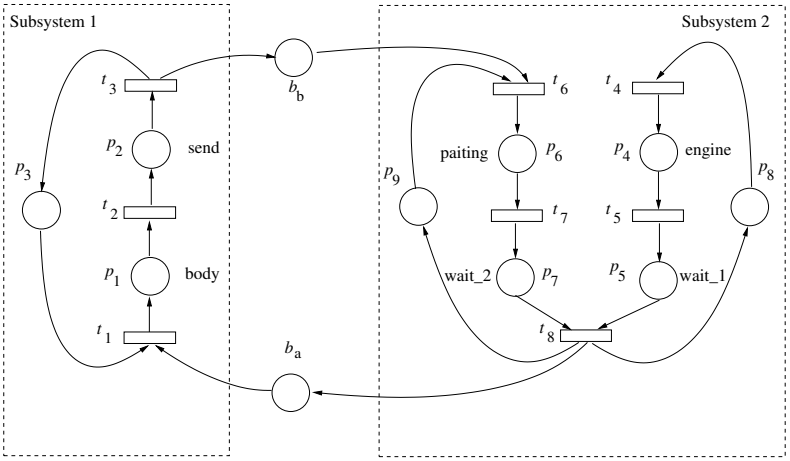

Fig. 1. A DcontPN marked graph modeling a car manufacturing plant where p_a and p_b are buffer places.

matrices; (iii) $\boldsymbol{m}_0 \in \mathbb{R}_{\geq 0}$ is the initial marking (state).

For $v \in P \cup T$, the sets of its input and output nodes are denoted as ${}^\bullet v$ and $v^\bullet$, respectively. Let p_i, $i = 1, \ldots, |P|$ and $t_j, j = 1, \ldots, |T|$ denote the places and transitions. Each place can contain a non-negative real number of tokens, this number represents the marking of the place. The distribution of tokens in places is denoted by $\boldsymbol{m}$. A transition $t_j \in T$ is enabled at $\boldsymbol{m}$ iff $\forall p_i \in {}^\bullet t_j$, $m(p_i) > 0$ and its enabling degree is given by

$$enab(t_j, \boldsymbol{m}) = \min_{p_i \in {}^\bullet t_j} \left\{ \frac{m(p_i)}{Pre(p_i, t_j)} \right\}$$

which represents the maximum amount in which t_j can fire. An enabled transition t_j can fire in any real amount α, with $0 < \alpha \leq enab(t_j, \boldsymbol{m})$ leading to a new state $\boldsymbol{m'} = \boldsymbol{m} + \alpha \cdot \boldsymbol{C}(\cdot, t_j)$ where $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the *token flow matrix*.

If $\boldsymbol{m}$ is reachable from $\boldsymbol{m}_0$ through a finite sequence σ, the state (or fundamental) equation is satisfied: $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}$, where $\boldsymbol{\sigma} \in \mathbb{R}^{|T|}_{\geq 0}$ is the *firing count vector*, i.e., σ_j is the cumulative amount of firings of t_j in the sequence σ.

Left and right natural annullers of the token flow matrix $\boldsymbol{C}$ are called *P-semiflows* (denoted by $\boldsymbol{y}$) and *T-semiflows* (denoted by $\boldsymbol{x}$), respectively. If $\exists\ \boldsymbol{y} > 0,\ \boldsymbol{y} \cdot \boldsymbol{C} = 0$, then the net is said to be *conservative*. If $\exists\ \boldsymbol{x} > 0,\ \boldsymbol{C} \cdot \boldsymbol{x} = 0$ it is said to be *consistent*. The support of a vector $\boldsymbol{v}$ is the set of nonzero components and denoted by $\|\boldsymbol{v}\|$. A semiflow $\boldsymbol{v}$ is said to be *minimal* when its support, $\|\boldsymbol{v}\|$, is not a proper superset of any other and the greatest common of its elements is one.

***Definition** 2.2 (MTS):* A continuous Petri net system is mono T-semiflow (MTS) net if it is conservative, consistent and has only one minimal T-semiflow.

In the time framework, the marking of place p_i at time τ is denoted by $m(p_i, \tau) \in \mathbb{R}_{\geq 0}$. The vector of all token loads is called *state* or *marking* and, is denoted by $\boldsymbol{m}(\tau) \in \mathbb{R}^{|P|}_{\geq 0}$. In contPNs the state equation has an explicit dependence on time: $\boldsymbol{m}(\tau) = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma}(\tau)$ which through time differentiation becomes $\dot{\boldsymbol{m}}(\tau) = \boldsymbol{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$. The derivative of the firing sequence $\boldsymbol{f}(\tau) = \dot{\boldsymbol{\sigma}}(\tau)$ is called the *firing flow*. Depending on how the flow is defined, many firing semantics appear, being the most used ones *infinite* and *finite* server semantics [10], [11]. For a broad class of Petri nets it is shown that infinite server semantics offers better approximation than finite server semantics [12]. This paper deals with *infinite server semantics* for which the flow of a transition t_j at time τ is the product of the firing rate λ_j associated to each transition t_j, and the enabling degree of the transition at $\boldsymbol{m}(\tau)$

$$f(t_j, \tau) = \lambda_j \cdot enab(t_j, \boldsymbol{m}(\tau)) = \lambda_j \cdot \min_{p_i \in {}^\bullet t_j} \left\{ \frac{m(p_i, \tau)}{Pre(p_i, t_j)} \right\} \quad (1)$$

***Definition** 2.3 (DcontPN):* A distributed continuous Petri net system is a set of contPN systems (called subsystems) interconnected through buffers/channels modeled as places (called buffer or channel places).

Let *K* denote the set of subsystems of a given DcontPN. The set of places, transitions and token flow matrix of subsystem $k \in K$ is denoted by P^k, T^k and $\boldsymbol{C}^k \in \mathbb{R}^{|P^k| \times |T^k|}$, respectively. We assume, $P^k \cap P^l = \emptyset$ and $T^k \cap T^l = \emptyset$, $\forall k, l \in K,\ k \neq l$. The directional connection between subsystems is provided by a set of places called *channels* or *buffers*. In particular, the directional connection from subsystem k to l is provided by a set of places denoted $B^{(k,l)}$, whose input transitions are contained only in subsystem k and output transitions are contained only in subsystem l, i.e., $B^{(k,l)} = \{b \in P | {}^\bullet b \in T^k,\ b^\bullet \in T^l,\ b \notin P^q \quad \forall q \in K\}$ for every $k, l \in K$, $k \neq l$, and $B^{(l,l)} = \emptyset$ for every $l \in K$.

Note that $b \in B^{(k,l)}$ is an *input buffer* of subsystem l and an *output buffer* of subsystem k. The set of all output buffers of subsystem k is denoted by $B^{(k,*)}$, i.e.,

$B^{(k,*)} = \bigcup_{l \in K} B^{(k,l)}$, and the set of all input channels of subsystem k is denoted by $B^{(*,k)}$, i.e., $B^{(*,k)} = \bigcup_{l \in K} B^{(l,k)}$.

The marking vector of subsystem k is denoted by $\boldsymbol{m}(P^k) \in \mathbb{R}_{\geq 0}^{|P^k|}$. When designing a controller, it must be taken into account that the controller of a given subsystem only knows its marking and the marking of its input buffers, i.e., the marking of the other subsystems and their input buffers are not observable.

***Example** 2.4:* Let us consider the DcontPN given in Fig. 1. It is composed of two subsystems: $P^1 = \{p_1, p_2, p_3\}$, $T^1 = \{t_1, t_2, t_3\}$; $P^2 = \{p_4, p_5, p_6, p_7, p_8, p_9\}$ and $T^2 = \{t_4, t_5, t_6, t_7, t_8\}$. These two subsystems communicate through two channels: b_b for the communication to subsystem 2 and b_a for the communication to subsystem 1. Hence, $B^{(1,2)} = \{b_b\}$ and $B^{(2,1)} = \{b_a\}$, implying $B^{(*,1)} = B^{(2,*)} = \{b_a\}$ and $B^{(*,2)} = B^{(1,*)} = \{b_b\}$.

III. Control of DcontPN

This section shows how control actions can be introduced and establishes the control problem that is considered. The autonomous (or uncontrolled) behavior of a DcontPN described in the previous section can be modified by introducing control actions. In continuous Petri nets the control actions are applied on the transitions and they can only *slow-down* (never speed-up) the firing flow of the transitions to which they are applied [11].

***Definition** 3.1:* The controlled flow, $\boldsymbol{w}$, of a timed DcontPN is defined as $\boldsymbol{w}(\tau) = \boldsymbol{f}(\tau) - \boldsymbol{u}(\tau)$, with $0 \leq \boldsymbol{u}(\tau) \leq \boldsymbol{f}(\tau)$, where $\boldsymbol{f}$ is the flow of the uncontrolled system, i.e., defined as in (1), and $\boldsymbol{u}$ is the control action.

Therefore, the control input $\boldsymbol{u}$ is dynamically upper bounded by the flow $\boldsymbol{f}$ of the corresponding unforced system. Under these conditions, the overall behavior of the system in which all transitions are controllable is ruled by:

$$\begin{array}{l} \dot{\boldsymbol{m}} = \boldsymbol{C} \cdot [\boldsymbol{f} \ - \ \boldsymbol{u}] = \boldsymbol{C} \cdot \boldsymbol{w} \\ 0 \leq \boldsymbol{u} \leq \boldsymbol{f} \end{array} \tag{2}$$

The integral of the controlled flow of a transition t_j over an interval of time $(\tau_a, \ \tau_b)$ is denoted by $s(t_j) = \int_{\tau_a}^{\tau_b} w(t_j) d\tau$. Note that, the flow integral vector of a contPN is the firing count vector of the underlying untimed net. For the sake of clarity, τ will be omitted: $f(t_j)$, $\boldsymbol{m}$ and $m(p_i)$ will be used instead of $f(t_j, \tau)$, $\boldsymbol{m}(\tau)$ and $m(p_i, \tau)$.

Among the different existing control problems, we will deal with a control problem which aims at reaching a particular target marking $\boldsymbol{m}_f$ at each subsystem. That is, after a finite period of time each subsystem is at its target marking. In contrast to a centralized control, each subsystem is equipped with its own controller that computes the control actions that drive the subsystem to the target marking.

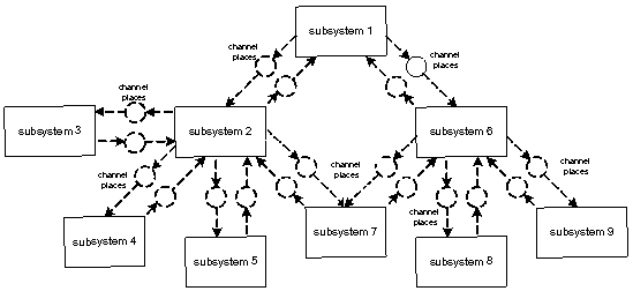

Fig. 2. A tree structure DcontPN system

Given that the subsystems are interconnected, they may require resources to be available in the communication buffers to reach the target marking.

***Example** 3.2:* Consider the DcontPN in Fig. 1 with $\boldsymbol{m}_0(P^1) = [0\ 0\ 3]^T$, $\boldsymbol{m}_0(P^2) = [0\ 0\ 0\ 0\ 2\ 2]^T$, $m_0(b_a) = 1$, $m_0(b_b) = 0$ and let $\boldsymbol{m}_f(P^1) = [0\ \ 0\ \ 3]^T$, $\boldsymbol{m}_f(P^2) = [0\ 0\ 1\ 0\ 2\ 1]^T$ be the target markings of each subsystem. Let the flow integrals of subsystem 1 and 2 be denoted as $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$ respectively.

A controller for the second subsystem could compute $s^2(t_6) = 1$, $s^2(t_4) = s^2(t_5) = s^2(t_7) = s^2(t_8) = 0$ so that the subsystem reaches the target marking. Given that the initial and target markings of subsystem 1 are the same, a controller for that subsystem could yield: $s^1(t_1) = s^1(t_2) = s^1(t_3) = 0$. Since $m_0(b_b) = 0$, transition t_6 cannot fire unless t_3 fires. Unfortunately, according to the computed controls, t_3 will not fire ($s^1(t_3) = 0$). Hence, these controls are not valid to reach the desired target marking of subsystem 2. In order to solve this situation, subsystem 2 may ask subsystem 1 to put enough tokens in b_b. This can be achieved easily by firing t_3. However, this will imply that subsystem 1 moves away from its desired target marking.

It could happen that the target markings cannot be reached due to the system structure or the initial marking of the buffers.

***Example** 3.3:* Consider again the DcontPN in Fig. 1. For subsystem 1, let the target marking be $\boldsymbol{m}_f(P^1) = [0\ 0\ 3]^T$ which is locally reachable from $\boldsymbol{m}_0(P^1) = [0\ 0\ 3]^T$. For subsystem 2, let the target marking be $\boldsymbol{m}_f(P^2) = [0\ 0\ 1\ 0\ 2\ 1]^T$ which is locally reachable from $\boldsymbol{m}_0(P^2) = [0\ 0\ 0\ 0\ 2\ 2]^T$ by firing t_6, i.e., if it is considered isolated from the rest of the system. But when both subsystems are connected through the buffers b_a and b_b with $m_0(b_a) = m_0(b_b) = 0$, the target markings are locally reachable but not globally reachable.

IV. An iterative algorithm to control DcontPNs

This section is devoted to the design of a distributed controller for distributed Petri net systems. We will extend

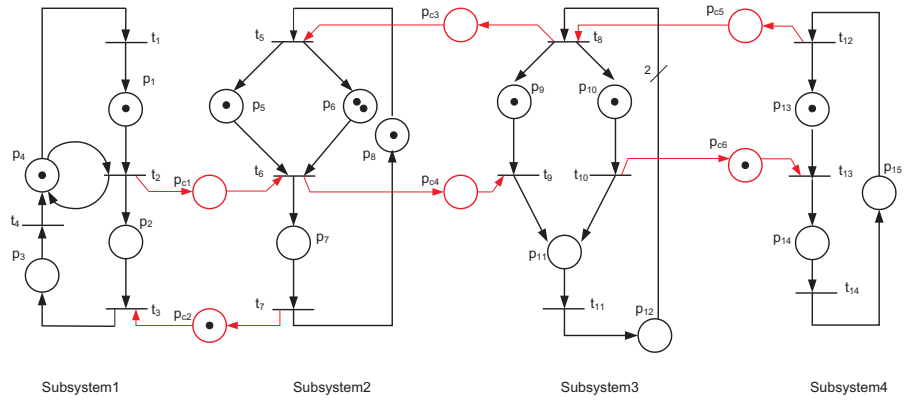

Fig. 3. A DcontPN used in Example 4.3.

our previous results in [8] where the problem has been studied for DcontPN composed only by two subsystems. In fact, we will show that iterating the algorithm in [8] all subsystems will compute their controllers. We will first present the algorithm associated to the local controller of each subsystem and then prove its correctness. We consider the following assumptions:

(A1) The target marking $\boldsymbol{m}_f$ is strictly positive and reachable at the overall system.

(A2) The DcontPN is composed of MTS subsystems. The minimal T-semiflows of the subsystem i is denoted $\boldsymbol{x}^i$.

(A3) The overall system is a MTS net system.

The first assumption is simply a necessary condition for reachability of the target markings. The second assumption reduces the class of DcontPN to those systems composed by MTS subsystems while the third one states that the overall system is MTS. In order to drive the subsystems from their initial states to the target states, Alg. 4.1 is developed. It represents the local controller that will be executed in each subsystem separately.

During the initialization, the algorithm requires the initial and target markings, the token flow matrix, the input and output buffers and the initial marking of the input buffers of the corresponding subsystem. In step 1, each subsystem computes the flow integral $\boldsymbol{z}$ required to reach its target marking without taking into account the marking of the buffers, this step is carried out by means of the Linear Programming Problem (LPP) (3). Step 2 sets the number of iterations to $|K|-1$, where $|K|$ is the number of subsystems, these iterations will be used in the next section where DcontPN with general structure are considered. Step 3 computes the amounts of tokens q_p^{req} to be produced in each input buffer p in order to be able to fire $\boldsymbol{z}$. The connected subsystems are informed about the amounts of required tokens q_p^{req} in step 4. In step 5, each subsystem receives the amount of tokens

---

Algorithm 4.1: [Distributed controller of subsystem k]

1) Solve

$$\begin{aligned} \min \quad & \mathbf{1}^T \cdot \boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{m}_f(P^k) - \boldsymbol{m}_0(P^k) = \boldsymbol{C}^k \cdot \boldsymbol{z}, \\ & \boldsymbol{z} \geq 0 \end{aligned} \tag{3}$$

2) **For** Iteration=1 **to** $|K|-1$ **do**

3) For every $p \in B^{(*,k)}$ compute

$$q_p^{req} = \left(\sum_{t \in p^\bullet} Pre(p,t) \cdot z(t) \right) - m_0(p)$$

4) For all $p \in B^{(*,k)}$ *send* q_p^{req} to the connected subsys.

5) For all $p \in B^{(k,*)}$ *receive* r_p^{req} from the conn. subsys.

6) For all $p \in B^{(k,*)}$ compute

$$h_p = \left(\sum_{t \in {}^\bullet p} Post(p,t) \cdot z(t) \right) - r_p^{req}$$

7) **If** $\min\limits_{p \in B^{(k,*)}} \{h_p\} < 0$ **then** solve

$$\begin{aligned} \min \quad & \mathbf{1}^T \cdot \boldsymbol{s} \\ \text{s.t.} \quad & \boldsymbol{m}_f(P^k) - \boldsymbol{m}_0(P^k) = \boldsymbol{C}^k \cdot \boldsymbol{s}, \\ & \left(\sum_{t \in {}^\bullet p} Post(p,t) \cdot s(t) \right) \geq r_p^{req}, \forall p \in B^{(k,*)} \\ & \boldsymbol{s} \geq 0 \end{aligned} \tag{4}$$

Else

$$\boldsymbol{s} = \boldsymbol{z}$$

End_If

8) $\boldsymbol{z} = \boldsymbol{s}$

9) **End_For**

10) **return** $\boldsymbol{s}$

---

it has to produce (if any) in its output buffers.

In step 6, it is computed how many tokens would remain in each output buffer if the present control was applied. If this value is negative, more tokens must be produced in the output buffers, and therefore the control law must be recomputed. This re-computation is achieved in step 7 using LPP (4). Observe that comparing with LPP (3) of step 1 only one extra constraint is added in order to ensure that enough tokens are produced in the output buffers. Since $|K| = 2$, the algorithm only executes one iteration.

The following Theorem shows that Alg. 4.1 computes a control law for all subsystems that ensures the reachability of their target markings. Moreover, there exists a (finite) time instant at which all subsystems are in their target markings.

***Theorem** 4.2:* Let $\mathcal{N}$ be a DcontPN satisfying assumptions $(A1)$, $(A2)$ and $(A3)$ by each pair of connected subsystems. Let $\boldsymbol{s}^k$ be the flow integral vectors computed by Alg. 4.1 for subsystem k for a given initial and target marking. The application of $\boldsymbol{s}^k$ drives the subsystems to their target markings.

Proof: Let us assume that Alg. 4.1 is applied to all subsystems. Let us consider the first iteration of the algorithm. In step 1 each controller k computes the required firing count vector $\boldsymbol{z}^k$ to reach its final marking by solving LPP (3). In step 5, subsystem k, receives from all its neighbors their requirements r_p^{req}. If it is necessary, the updating of $\boldsymbol{z}^k$ to satisfy such requirements is achieved by LPP (4) in step 7.

Observe that after each iteration, the requirements of each subsystem are satisfied by the updating of the firing count vector of the neighbors subsystems what makes the marking of at least one buffer equal to zero (this is result of minimizing the firing count vector in LPP (4)).

Assume that the Alg. 4.1 has been executed for $|K| - 1$ iterations and that after the last iteration, step 3 to 6 are further executed in order to obtain values for h_p. If all h_p are non-negative then the computed firing count vectors ensure the reachability of the target marking in each subsystem.

On the other hand, let us assume that at least one h_p is negative. We will prove that this leads to the global non-reachibility, violating assumption $(A1)$. An negative h_p implies that: a) at each iteration of the algorithm at least one subsystem updated its firing count vector; b) after $|K| - 1$ iterations, at least one subsystem would still need to update its firing count vector. These both facts imply that there exists a subsystem Q that would need to perform two updates, what in turn involves that there is a cycle of subsystems containing Q whose marking in the connecting buffers is not positive. Notice that there exists a P-semiflow containing such connecting buffers and places of the subsystems for which the target marking is fixed. For this reason, the target marking under consideration violates at least one existing conservation law of the conservative global system, and therefore such a marking is not globally reachable. ■

***Example** 4.3:* Let us consider DcontPN in Fig. 3 which is composed by 4 subsystems. Hence, $|K| = 4$ and Alg. 4.1 will be iterated at most 4 times. These subsystems communicate through 6 channel places. The set of buffers are: $B^{(1,2)} = \{p_{c1}\}$, $B^{(2,1)} = \{p_{c2}\}$, $B^{(3,2)} = \{p_{c3}\}$, $B^{(2,3)} = \{p_{c4}\}$, $B^{(4,3)} = \{p_{c5}\}$ and $B^{(3,4)} = \{p_{c6}\}$.

Let us assume the following initial markings: $\boldsymbol{m}_0(P^1) = [1\ \ 0\ \ 0\ \ 1]^T$, $\boldsymbol{m}_0(P^2) = [1\ \ 2\ \ 0\ \ 1]^T$, $\boldsymbol{m}_0(P^3) = [1\ 1\ 0\ 0]^T$, $\boldsymbol{m}_0(P^4) = [1\ 0\ 0]^T$, $m_0(p_{c2}) = m_0(p_{c6}) = 1$, $m_0(p_{c1}) = m_0(p_{c3}) = m_0(p_{c4}) = m_0(p_{c5}) = m_0(p_{c5}) = 0$ and final markings $\boldsymbol{m}_f(P^1) = [1\ 1\ 0\ 0]^T$, $\boldsymbol{m}_f(P^2) = [0\ 1\ 2\ 0]^T$, $\boldsymbol{m}_f(P^3) = [0\ 1\ 1\ 0]^T$, $\boldsymbol{m}_f(P^4) = [0\ 0\ 1]^T$.

The execution of Alg. 4.1 is given in Table II. According to final flow integrals, the final markings of the buffers are $m_f(p_{c1}) = m_f(p_{c2}) = m_f(p_{c3}) = m_f(p_{c4}) = m_f(p_{c5}) = m_f(p_{c6}) = 0$. In the first iteration of the algorithm it holds that $h_{pc_1} < 0$ and $h_{pc_3} < 0$, thus the flow integral of subsystem 1 and subsystem 3 are recomputed. In the second iteration, $h_{pc_5} < 0$ is obtained, hence the flow integral of subsystem 4 is updated. Observe that after two iterations the control laws are computed. Therefore, the next two iterations that are not given in Table II are redundant. Unfortunatelly they cannot be avoided since at the side of each subsystem the whole system structure is unknown.

Once the flow integral vectors $\boldsymbol{s}$ of the evolution from the initial marking to the target marking have been computed by Alg. 4.1, the value of the control actions $\boldsymbol{u}$ can be derived in several ways (for example applying the procedure in [9]) as long as $\boldsymbol{s} = \int_{\tau_a}^{\tau_b} (\boldsymbol{f} - \boldsymbol{u}) d\tau$ is satisfied where τ_a and τ_b are the initial and final time instants respectively. Remark that $\boldsymbol{s}$ can be seen as a firing count vector in the untimed system.

V. Conclusions

Distributed systems are composed of several subsystems that exchange parts in order to obtain a given goal. This paper focus on distributed systems modeled by continuous Petri nets and a reachability control problem has been considered. The approach developed here is based on the design of a local controller for each subsystem. The main difficulties that must be taken into account when dealing with the mentioned control problem are related to the coordination among local controllers and the possibility of reaching the target marking in every subsystem. It is proved that, under certain assumptions on the system, the proposed algorithm always yields an appropriate control law.

Step	Subsystem 1	Subsystem 2	Subsystem 3	Subsystem 4
Step 1	$\boldsymbol{z} = [1\ 1\ 0\ 0]^T$	$\boldsymbol{z} = [1\ 2\ 0]^T$	$\boldsymbol{z} = [0\ 1\ 0\ 0]^T$	$\boldsymbol{z} = [0\ 1\ 1]^T$
Step 2	$Iteration = 1$	$Iteration = 1$	$Iteration = 1$	$Iteration = 1$
Step 3 & 4	Send $q^{req}_{p_{c2}} = -1$	Send $q^{req}_{p_{c1}} = 2$ Send $q^{req}_{p_{c3}} = 1$	Send $q^{req}_{p_{c4}} = 1$ Send $q^{req}_{p_{c5}} = 0$	Send $q^{req}_{p_{c6}} = 0$
Step 5	Receive $r^{req}_{p_{c1}} = 2$	Receive $r^{req}_{p_{c4}} = 1$ Receive $r^{req}_{p_{c2}} = -1$	Receive $r^{req}_{p_{c6}} = 0$ Receive $r^{req}_{p_{c3}} = 1$	Receive $r^{req}_{p_{c5}} = 0$
Step 6	$h_{p_{c1}} = -1$	$h_{p_{c2}} = h_{p_{c4}} = 1$	$h_{p_{c3}} = -1, h_{p_{c6}} = 0$	$h_{p_{c5}} = 0$
Step 7	$\boldsymbol{s} = [2\ 2\ 1\ 1]^T$	$\boldsymbol{s} = [1\ 2\ 0]^T$	$\boldsymbol{s} = [1\ 2\ 1\ 2]^T$	$\boldsymbol{s} = [0\ 1\ 1]^T$
Step 8	$\boldsymbol{z} = \boldsymbol{s}$	$\boldsymbol{z} = \boldsymbol{s}$	$\boldsymbol{z} = \boldsymbol{s}$	$\boldsymbol{z} = \boldsymbol{s}$
Step 2	$Iteration = 2$	$Iteration = 2$	$Iteration = 2$	$Iteration = 2$
Step 3 & 4	Send $q^{req}_{p_{c2}} = 0$	Send $q^{req}_{p_{c1}} = 2$ Send $q^{req}_{p_{c3}} = 1$	Send $q^{req}_{p_{c4}} = 1$ Send $q^{req}_{p_{c5}} = 1$	Send $q^{req}_{p_{c6}} = 1$
Step 5	Receive $r^{req}_{p_{c1}} = 2$	Receive $r^{req}_{p_{c4}} = 2$ Receive $r^{req}_{p_{c2}} = 0$	Receive $r^{req}_{p_{c6}} = 1$ Receive $r^{req}_{p_{c3}} = 1$	Receive $r^{req}_{p_{c5}} = 1$
Step 6	$h_{p_{c1}} = 0$	$h_{p_{c2}} = h_{p_{c4}} = 0$	$h_{p_{c3}} = 0, h_{p_{c6}} = 1$	$h_{p_{c5}} = -1$
Step 7	$\boldsymbol{s} = [2\ 2\ 1\ 1]^T$	$\boldsymbol{s} = [1\ 2\ 0]^T$	$\boldsymbol{s} = [1\ 2\ 1\ 2]^T$	$\boldsymbol{s} = [1\ 2\ 2]^T$

TABLE I

EXECUTION OF ALG. 4.1 ON SUBSYSTEMS OF DCONTPN IN FIG. 3

REFERENCES

[1] J. Komenda and J. H. van Schuppen, "Control of discrete-event systems with modular or distributed structure," *Theoretical Computer Science*, vol. 388, no. 1–3, pp. 199–226, 2007.

[2] W. M. Wonham and P. J. Ramadge, "Modular supervisory control of discrete-event systems," *Mathematics of Control, Signals, and Systems*, vol. 1, no. 1, 1988.

[3] P. Krishnan, "Distributed timed automata," in *WDS'99, Workshop on Distributed Systems (A satellite workshop to FCT'99)*, ser. Electronic Notes in Theoretical Computer Science, vol. 28, 2000, pp. 5 – 21.

[4] P. Darondeau, "Distributed Implementation of Ramadge-Wonham Supervisory Control with Petri nets," in *CDC05: IEEE Conference on Decision and Control*, December 2005, pp. 2107–2112.

[5] R. P. Moreno, D. Tardioli, and J. Salcedo, "Distributed Implementation of Discrete Event Control Systems based on Petri nets," in *IEEE Int. Symposium on Ind. Electronics*, June 2008, pp. 1738–1745.

[6] B. Bordbar, L. Giacomani, and D. Holding, "Design of Distributed Manufacturing Systems Using UML and Petri nets," in *Proc. of 6th IFAC Workshop on Algorithms and Architectures for Real-Time Control*, Mallorca, Spain, May 2000, pp. 91–96.

[7] E. Fabre and L. Jezequel, "Distributed Optimal Planning: An Approach by Weighted Automata Calculus," in 48^{th} *Conference on Decision and Control (CDC)*, Shanghai, P.R. China, December 2009, pp. 211 – 216.

[8] H. Apaydin-Ozkan, J. Julvez, C. Mahulea, and M. Silva, "A Control Method for Timed Distributed Continuous Petri nets," in *2010 American Control Conference*, Baltimore, USA, June 2010, to appear.

[9] ——, "An Efficient Heuristics for Minimum Time Control of Continuous Petri nets," in 3^{rd} *IFAC Conf. on Analysis and Design of Hybrid Systems*, Zaragoza, Spain, September 2009, pp. 44–49.

[10] R. David and H.Alla, *Discrete, Continuous, and Hybrid Petri Nets,* 2^{nd} *edition*. Springer Berlin Heidelberg, 2010.

[11] M. Silva and L. Recalde, "On fluidification of Petri net models: from discrete to hybrid and continuous models," *Annual Reviews in Control*, vol. 28, no. 2, pp. 253–266, 2004.

[12] C. Mahulea, L. Recalde, and M. Silva, "Basic Server Semantics and Performance Monotonicity of Continuous Petri Nets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 19, no. 2, pp. 189 – 212, June 2009.

[13] C. Mahulea, A. Ramirez, L. Recalde, and M.Silva, "Steady state control reference and token conservation laws in continuous petri net systems," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 2, pp. 307–320, April 2008.