An Efficient Algorithm for the Transient Analysis of a Class of Deterministic Stochastic Petri Nets

M. Gribaudo and M. Sereno
Dipartimento di Informatica, Università di Torino, Torino, Italia

Abstract

In this paper a new algorithm for the transient solution of a sub-class of Deterministic Stochastic Petri Nets (DSPN) is proposed. The technique can be applied to DSPNs comprising only deterministic and immediate transitions and such that in each tangible marking only one deterministic transition is enabled. The algorithm does not require any additional restriction on the deterministic transition delays that can have any positive real value. Most of the optimized algorithms presented in the literature are based on an efficient solution of the equations governing the stochastic process associated with the DSPN; the new algorithm we propose is based on an efficient combinatorial analysis of the paths within the state space underlying the DSPN, instead.

1. Introduction

Stochastic Petri Nets (SPNs) represent a powerful formalism for modeling and evaluating systems exhibiting concurrency, synchronization, and conflict. The ability to model probabilistic behavior is essential in the field of performance and reliability evaluation. This need leads to various different variants of the SPN formalism. *Deterministic and Stochastic Petri Nets (DSPNs)* introduced in [1] are a stochastic formalism which include both exponentially and deterministic delays. Under the restriction that in any marking of a DSPN at most one deterministic transition is enabled, highly efficient numerical methods for steady state analysis have been provided (see [8] and [5]). This restriction has been removed in [10]. This paper proposes an efficient numerical method for steady state analysis of DSPNs with concurrent deterministic transitions.
Previous work on transient analysis of DSPNs was always based on the restriction that deterministic transitions are not concurrently enabled. In [2] it has been show that, with this restriction, the stochastic process underlying a DSPN is a Markov regenerative stochastic process. Based on this result, a numerical method for the transient analysis of such DSPN is proposed. The technique is based on a numerical inversion of Laplace-Stiltjes transforms.
Numerical methods based on the method of supplementary variables have been presented in [6]. Supplementary variable approach methods require the numerical solution of systems of partial differential equations.
The paper [9] introduces an efficient method for transient analysis od DSPNs without restrictions on the enabling of deterministic transitions, i.e., the proposed technique allows concurrent deterministic transitions. This paper introduces a new algorithm for the transient solution of a sub-class of DSPNs. The technique can be applied to DSPNs comprising only deterministic and immediate transitions and such that in each tangible marking only one deterministic transition is enabled (in the following we denote this sub-class of DSPNs a D-DSPNs). Most of the transient algorithms presented in the literature are based on an efficient solution of the equations governing the stochastic process associated with the D-DSPN; instead, the new algorithm we propose is based on an efficient combinatorial analysis of the paths within the state space underlying the D-DSPN. The algorithm we present in this paper does not require any additional restriction on the deterministic transition delays that can have any positive real value.

For the transient solution of D-DSPNs we have different possibilities. In particular, we can use the methods proposed in [6] and implemented in TimeNet package [11]. With some additional restriction on the the deterministic transition delays (i.e., all the transition delays have to be equal) we can also use the results presented in [9] and implemented in new version of the DSPNexpress package [8]. Solution algorithms for the class of D-DSPNs have also been proposed in other works, for instance the paper [3] proposed a technique that is able to derive the embedded DTMC by determining a basic step of the transition delays. By using a fine step, arbitrary delays can be approximated, but this increases the state space of the DTMC.

The algorithm that we propose in this paper does not require any additional restriction on the deterministic transition delays that thus can have any positive real value. Another interesting point of our transient solution algorithm is that it can be easily extended to be used in case of D-DSPN

Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN 2004)
0-7695-2052-9/04 $20.00 © 2004 IEEE

with infinite state space.

The balance of this paper is outlined as follows. Section 2 describes the transient solution algorithm. Section 3 presents two examples of application of this algorithm for the evaluation of interesting models. Finally, Section 4 provides some concluding remarks.

2. The Transient Solution Algorithm

The D-DSPNs are DSPNs comprising only deterministic and immediate transitions. We can handle all the features allowed by the class of DSPNs (see [8] and [5] for details). We also impose the classical restriction that is used for many the solution algorithms for DSPNs, that is, in each tangible marking only one deterministic transition is enabled.

In this section we provide a description of the new transient solution algorithm. In particular,

- in Section 2.1 we present some basic definitions that will be used to describe the technique;
- in Section 2.2 we provide a detailed description of the proposed technique;
- in Section 2.3 we illustrate the new transient solution algorithm with a help of a simple D-DSPN;
- in Section 2.4 we present a pseudo-code description of our new technique and then we discuss some implementation issues.

2.1. Embedded Markov Chain

In order to describe the solution algorithm, we first observe that the stochastic process underlying the D-DSPN model is similar to an embedded Markov process, with a deterministic, state dependent sojourn time, and time dependent state jump probability. Since only deterministic timed transition are involved, the sojourn time in a state can be easily determined. In the following we denote by $T^{(i)}$ the deterministic transition enabled in marking $\boldsymbol{m}_i$. Note that for some marking $\boldsymbol{m}_i \neq \boldsymbol{m}_j$, we may have that $T^{(i)} = T^{(j)}$, since it can happen that both markings $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ enable the same deterministic transition.

Let us denote by $\mathcal{S}$ the tangible state space of the D-DSPN. When timed transition $T^{(i)}$ fires, the next marking can either be tangible or vanishing. Due to the path of immediate transitions that may follow the firing of a deterministic one, different tangible marking can be reached from a single tangible marking. This stochastic process can be described by a matrix $\boldsymbol{C}$ and a vector $\boldsymbol{\sigma}$. Matrix $\boldsymbol{C}$ is the state transition probability matrix of the stochastic process. The size of this matrix is equal to the size of the tangible state space of the D-DSPN. Each element $c_{i\,j}$ represents the state transition probability from state $\boldsymbol{m}_i$ to state $\boldsymbol{m}_j$ when transition $T^{(i)}$ fires (obviously $\sum_j c_{i\,j} = 1$).

The vector $\boldsymbol{\sigma}$ accounts the sojourn time of the stochastic process, i.e., σ_i is the firing time of transition $T^{(i)}$.

The key point of the technique is that, since only deterministic transitions are involved, it is possible to determine the exact time at which the enabled transition will fire. In particular, we denote by τ_k the sequence of time instants where at least a transition will fire. We assume that:

$$\tau_0 = 0, \text{ and } \tau_k < \tau_{k+1}, \forall k \geq 0. \tag{1}$$

In Section 2.2 we will see how to compute the sequence $\tau_k, \;\; k = 0, 1, \ldots$. In each state that has a probability greater than zero, a timed transition is enabled (since we consider only tangible states). Thus it may happen that more than a single deterministic transition fire exactly at the same time instant τ_k. We will see how to consider this possibility.

Let $\boldsymbol{\pi}(\tau)$ be the probability vector at time τ, i.e., the component $\pi_i(\tau)$ represents the probability of being in marking $\boldsymbol{m}_i$ at time τ. Since only deterministic and immediate transitions are involved, the probability distribution does not change between two consecutive firing instants τ_k and τ_{k+1}. That is:

$$\boldsymbol{\pi}(\tau) = \boldsymbol{\pi}(\tau'), \;\; \forall \tau, \tau' \in [\tau_k, \tau_{k+1}),$$

The probability distribution at time τ_{k+1}, will be computed by summing to $\boldsymbol{\pi}(\tau_k)$ two terms, $\boldsymbol{\Delta\pi}^{+}(k)$ and $\boldsymbol{\Delta\pi}^{-}(k)$, that is:

$$\boldsymbol{\pi}(\tau_{k+1}) = \boldsymbol{\pi}(\tau_k) + \boldsymbol{\Delta\pi}^{+}(k) - \boldsymbol{\Delta\pi}^{-}(k). \tag{2}$$

In Section 2.2 we provide a method for computing these probability increments $\boldsymbol{\Delta\pi}^{+}(k)$ and $\boldsymbol{\Delta\pi}^{-}(k)$. We must separate the additive part from the subtractive part because only the incoming probability into a state enables its associated transition.

2.2. The solution algorithm

Let $\boldsymbol{\pi}(\tau_0)$ (with $\tau_0 = 0$) be the initial state probability distribution (which can be easily derived from the initial marking of the model).

To compute the transient distribution it is sufficient to compute the values τ_k, $\boldsymbol{\Delta\pi}^{-}(k)$ $\boldsymbol{\Delta\pi}^{+}(k)$ for every $k > 0$.

Let us denote by

$$\mathcal{S}^{(0)} = \{\boldsymbol{m}_j : \pi_j(\tau_0) > 0\} \tag{3}$$

the set of markings that have a non-zero probability in the initial state, and by

$$\boldsymbol{\Theta}^{(0)} = \left\{\sigma_j : \boldsymbol{m}_j \in \mathcal{S}^{(0)}\right\}, \tag{4}$$

the set of **different** delays of the deterministic transition that can be enabled in the possible initial markings. Since there can be deterministic transitions having the same delay, it may happen that $|\mathcal{S}^{(0)}| > |\boldsymbol{\Theta}^{(0)}|$.

In the following first compute τ_k, $\mathbf{\Delta\pi}^-(k)$, and $\mathbf{\Delta\pi}^+(k)$ for every $k \leq |\mathbf{\Theta}^{(0)}|$ and then for $k > |\mathbf{\Theta}^{(0)}|$. In other words, we first describe the computation of the τ_k, $\mathbf{\Delta\pi}^-(k)$, and $\mathbf{\Delta\pi}^+(k)$ starting from the initial marking, and later on we extend the procedure for the computation of these values for any marking that can be reached from the initial marking.

Computation of τ_k, $\mathbf{\Delta\pi}^+(k)$ and $\mathbf{\Delta\pi}^-(k)$ for $k \leq |\mathbf{\Theta}^{(0)}|$.
We order the set $\mathbf{\Theta}^{(0)}$, in this manner for any $k = 1, \ldots, |\mathbf{\Theta}^{(0)}|$ we have that $\theta_k^{(0)} < \theta_{k+1}^{(0)}$ (with $\theta_k^{(0)}, \theta_{k+1}^{(0)} \in \mathbf{\Theta}^{(0)}$). We can derive that

$$\tau_k = \theta_k^{(0)}, \quad \forall k = 1, \ldots, |\mathbf{\Theta}^{(0)}|. \tag{5}$$

This simply means that all the deterministic transitions that are enabled in the initial markings will fire after their firing time.

Let us address the computation of $\mathbf{\Delta\pi}^+(k)$ and $\mathbf{\Delta\pi}^-(k)$ for $k \leq |\mathbf{\Theta}^{(0)}|$. We can set $\mathbf{\Delta\pi}^+(k) = \mathbf{\Delta\pi}^-(k) = \mathbf{0}$, since at time $\tau_0 = 0$ we know the initial distribution and $\mathbf{\Delta\pi}^+(k)$ and $\mathbf{\Delta\pi}^-(k)$ have no meanings.
Let denote by

$$\mathcal{B}_k = \{\boldsymbol{m}_j : \boldsymbol{m}_j \in \mathcal{S}^{(0)}, \text{ and } \sigma_j = \tau_k\}, \quad k = 1, \ldots, |\mathbf{\Theta}^{(0)}|, \tag{6}$$

the set of all the possible initial markings whose associated deterministic transitions have the same firing time (this is required because as stated before, there may be more than one deterministic transition with the same firing time). We define a diagonal matrix of size equal to $|\mathcal{S}|$, $\boldsymbol{I}^{(k)}$ such that $I_{j\,j}^{(k)} = 1$ if $\boldsymbol{m}_j \in \mathcal{B}_k$, $I_{j\,j}^{(k)} = 0$ otherwise. We can derive that

$$\mathbf{\Delta\pi}^-(k) = \boldsymbol{\pi}(\tau_0)\boldsymbol{I}^{(k)} \tag{7}$$
$$\mathbf{\Delta\pi}^+(k) = \mathbf{\Delta\pi}^-(k)\boldsymbol{C} = \boldsymbol{\pi}(\tau_0)\boldsymbol{I}^{(k)}\boldsymbol{C}. \tag{8}$$

$\mathbf{\Delta\pi}^-(k)$ represents the probability that flows out of the states due to the transition firings at time τ_k (that is the probability leaving the states $\boldsymbol{m}_j \in \mathcal{B}_k$), and $\mathbf{\Delta\pi}^+(k)$ represents the probability that enters the new states reached after the firing of the various transitions that fires at time τ_k. This quantity is simply what leaves the states $\mathbf{\Delta\pi}^-(k)$ distributed according to matrix $\boldsymbol{C}$. This expression takes into account both the cases when there is more than one marking whose associated transitions fires at time τ_k, (in this case the number of non-zero elements of $\mathbf{\Delta\pi}^-(k)$ is greater than one) and when there are some conflict in the vanishing marking reached after the firing of the transition (in this case, the number of non-zero elements of the row of $\boldsymbol{C}$ corresponding to the state that is left are greater than one).

Determining τ_k, $\mathbf{\Delta\pi}^+(k)$ and $\mathbf{\Delta\pi}^-(k)$ for $k > |\mathbf{\Theta}^{(0)}|$.
Now let us consider what happens at firing time τ_1. The process jumps from one of the initial states (in particular from one of the states $\boldsymbol{m}_i$ such that $\sigma_i = \tau_1$) to some new tangible state $\boldsymbol{m}_j$. The set of markings that are reached at time τ_1 can be determined as:

$$\mathcal{S}^{(1)} = \{\boldsymbol{m}_j : \Delta\pi_j^+(1) > 0\}, \tag{9}$$

This set is important because it accounts for the deterministic transitions $T^{(j)} : \boldsymbol{m}_j \in \mathcal{S}^{(1)}$ that becomes enabled at time τ_1. This mean that transition $T^{(j)}$ will fire at time $\tau_1 + \sigma_j$ (σ_j is the delay of the only deterministic transition enabled in marking $\boldsymbol{m}_j$). With these consideration we can determine

$$\mathbf{\Theta}^{(1)} = \{\sigma_j : \boldsymbol{m}_j \in \mathcal{S}^{(1)}\}, \tag{10}$$

that is $\mathbf{\Theta}^{(1)}$ is the the ordered set of delays of the transitions that may be enabled at time τ_1. Let be $m = |\mathbf{\Theta}^{(0)}|$, we first assume that $\tau_m < \tau_1 + \theta_1^{(1)}$, that is the transition with the earliest firing time, will fire after the latest firing of the transition enabled in the initial marking. Later on we will show how to manage the case where $\tau_m \geq \tau_1 + \theta_1^{(1)}$.
From these considerations we can the compute

$$\tau_{m+k} = \tau_1 + \theta_k^{(1)}, \quad \forall k = 1, \ldots, |\mathbf{\Theta}^{(1)}|. \tag{11}$$

After the computation of the firing instants τ_{m+k} we can compute the new probability increments $\mathbf{\Delta\pi}^+(m+k)$ and $\mathbf{\Delta\pi}^-(m+k)$. We first define

$$\mathcal{B}_{m+k} = \left\{\boldsymbol{m}_j : \boldsymbol{m}_j \in \mathcal{S}^{(1)}, \text{ and } \sigma_j = \theta_k^{(1)}\right\}, \quad \text{for } k = 1, \ldots, |\mathbf{\Theta}^{(1)}|, \tag{12}$$

that is the set of states whose associated timed transition fires at time τ_{m+k}, and a diagonal matrix $\boldsymbol{I}^{(m+k)}$ such that $I_{j\,j}^{(m+k)} = 1$ if $\boldsymbol{m}_j \in \mathcal{B}_{m+k}$ and $I_{j\,j}^{(m+k)} = 0$ otherwise. We can derive that

$$\mathbf{\Delta\pi}^-(m+k) = \mathbf{\Delta\pi}^+(1)\boldsymbol{I}^{(m+k)}$$
$$\mathbf{\Delta\pi}^+(m+k) = \mathbf{\Delta\pi}^+(m+k)\boldsymbol{C} = \mathbf{\Delta\pi}(1)\boldsymbol{I}^{(n+k)}\boldsymbol{C}. \tag{13}$$

In the general case, only the probability that enters the state at time τ_1 will *move out of the state* at time τ_{m+k} due to the firing of the deterministic transition. This happens because a deterministic transitions $T^{(i)}$ becomes enabled as soon as some probability enters marking $\boldsymbol{m}_i$, and fires exactly after σ_i. This means that at every time τ, the probability $\pi_i(\tau)$ of a marking $\boldsymbol{m}_i$ considers together many transitions with different clocks. Instead $\Delta\boldsymbol{m}_i$ considers only the one that where enabled at time τ_k and have the same clock. This results in removing all the probability that enabled it, and distributing it among its possible destination. $\mathbf{\Delta\pi}^+(1)$ takes into account the probability that entered a state at a given time instant.

	Tangible		Vanishing
$\boldsymbol{m}_0$	$\{p_4\}$	$\boldsymbol{m}_5$	$\{p_0\}$
$\boldsymbol{m}_1$	$\{p_1\}$	$\boldsymbol{m}_6$	$\{p_5\}$
$\boldsymbol{m}_2$	$\{p_2\}$	$\boldsymbol{m}_7$	$\{p_6\}$
$\boldsymbol{m}_3$	$\{p_3\}$	$\boldsymbol{m}_8$	$\{p_6, p_7\}$
$\boldsymbol{m}_4$	$\{p_3, p_7\}$		

Table 1. List of all the reachable marking for the D-DSPN of Figure 1

If $\tau_m \geq \tau_1 + \theta_1^{(1)}$ (that is, some of the newly enabled transition fires earlier than the one enabled in the initial marking), we simply order the terms τ_l, $\boldsymbol{\Delta\pi}^+(l)$ and $\boldsymbol{\Delta\pi}^-(l)$ such that $\tau_l \leq \tau_{l+1}, \forall l \geq 0$. It may happen that $\tau_l = \tau_{l+1}$ for some index l, which violates the constraints of Equation (1), i.e, all the time instants τ_l are different. In this case we will simply merge the jumps; that is, if $\tau_l = \tau_{l+1}$, we set:

$$\begin{aligned} \boldsymbol{\Delta\pi}^+(l) &= \boldsymbol{\Delta\pi}^+(l) + \boldsymbol{\Delta\pi}^+(l+1) \\ \boldsymbol{\Delta\pi}^-(l) &= \boldsymbol{\Delta\pi}^-(l) + \boldsymbol{\Delta\pi}^-(l+1), \end{aligned} \quad (14)$$

and we then drop the terms τ_{l+1}, $\boldsymbol{\Delta\pi}^+(l+1)$ and $\boldsymbol{\Delta\pi}^-(l+1)$.

2.3. A Simple Example

Consider the D-DSPN depicted in Figure 1. This net represents a system that can perform three different activities. Activity one (deterministic transition T_1) cannot fail. Activity two (deterministic transition T_2) can either succeed (immediate transition t_4), or fail (immediate transition t_5), in the latter case it must be repeated until it succeeds. Activity three (deterministic transition T_3) can also either succeed (immediate transition t_7) or fail (immediate transition t_6), but in contrast with activity two, it can be repeated only once. This is ensured by place p_7 that becomes marked and hence transitions t_6 and t_7 are not enabled (inhibitor arcs from place p_7 to transition t_6 and transition t_7).

The D-DSPN has 9 possible markings: 4 vanishing and 5 tangible. In Table 1 we provide all the reachable markings, while Figure 2 shows the reachability graph (Figure 2(a) presents the complete reachability graph while Figure 2(b) depicts only the tangible reachability graph).
We have that:

$$\begin{aligned} \mathcal{S} &= \{\boldsymbol{m}_0, \boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, \boldsymbol{m}_4\} \\ \boldsymbol{C} &= \begin{bmatrix} 0 & 0.3 & 0.2 & 0.5 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0.75 & 0 & 0.25 & 0 & 0 \\ 0.6 & 0 & 0 & 0 & 0.4 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \boldsymbol{\sigma} &= [2.7, 2, 1, 2, 2], \end{aligned}$$

and $T^{(0)} = T_4$, $T^{(1)} = T_1$, $T^{(2)} = T_2$, $T^{(3)} = T_3$, and $T^{(4)} = T_3$. Note that $T^{(3)} = T^{(4)}$.
Since the initial marking is a vanishing marking (since three immediate transitions are enabled), we have that

$$\boldsymbol{\pi}(\tau_0) = [0, 0.3, 0.2, 0.5, 0],$$

and by Equation (3) we can write

$$\mathcal{S}^{(0)} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3\},$$

that is, we have three possible different tangible markings that have a non-zero probability in the initial state. The set $\boldsymbol{\Theta}^{(0)}$ can be defined by using Equation (4), and in particular we have that

$$\boldsymbol{\Theta}^{(0)} = \{1, 2\},$$

since $\sigma_1 = \sigma_3$.
The first firing time instants that can be defined by using Equation (5) are:

$$\tau_1 = 1, \quad \tau_2 = 2.$$

By using Equation (6) we can derive the two sets of markings whose associated deterministic transitions have the firing time equal to τ_1 and to τ_2. In particular we have that $\mathcal{B}_1 = \{\boldsymbol{m}_2\}$ and $\mathcal{B}_2 = \{\boldsymbol{m}_1, \boldsymbol{m}_3\}$.
By using Equations (7) we can derive the probability increments/decrements at times τ_1 and τ_2:

$$\begin{aligned} \boldsymbol{\Delta\pi}^+(1) &= [0.15, 0, 0.05, 0, 0] \\ \boldsymbol{\Delta\pi}^-(1) &= [0, 0, 0.2, 0, 0] \end{aligned}$$

$$\begin{aligned} \boldsymbol{\Delta\pi}^+(2) &= [0.6, 0, 0, 0, 0.2] \\ \boldsymbol{\Delta\pi}^-(2) &= [0, 0.3, 0, 0.5, 0]. \end{aligned}$$

At time τ_1, by using Equation (9) we can derive the set

$$\mathcal{S}^{(1)} = \{\boldsymbol{m}_0, \boldsymbol{m}_2\},$$

i.e., we have two states that can be reached at time τ_1 and by Equation (10) we derive that

$$\boldsymbol{\Theta}^{(1)} = \{1, 2.7\}.$$

Equation (11) allows us to compute the firing time instants after τ_1, in particular, since $|\boldsymbol{\Theta}^{(0)}| = 2$, we have that

$$\begin{aligned} \tau_{2+1} &= \tau_3 = \tau_1 + \theta_1^{(1)} = 1 + 1 = 2 \quad \text{and} \\ \tau_4 &= \tau_1 + \theta_2^{(1)} = 1 + 2.7 = 3.7. \end{aligned}$$

The sets of markings $\mathcal{B}_{2+k}$ (for $k = 1, 2$) can be defined by using Equation (12): $\mathcal{B}_3 = \{\boldsymbol{m}_2\}$ and $\mathcal{B}_4 = \{\boldsymbol{m}_0\}$, and their probability increments/decrements:

$$\begin{aligned} \boldsymbol{\Delta\pi}^+(3) &= [0.0375, 0, 0.0125, 0, 0] \\ \boldsymbol{\Delta\pi}^-(3) &= [0, 0, 0.05, 0, 0] \end{aligned}$$

$$\begin{aligned} \boldsymbol{\Delta\pi}^+(4) &= [0, 0.045, 0.03, 0.075, 0] \\ \boldsymbol{\Delta\pi}^-(4) &= [0.15, 0, 0, 0, 0]. \end{aligned}$$

Since $\tau_3 = \tau_2$ we can merge the corresponding probability increments/decrements (see Equations (14)) and then we obtain the update values for $\boldsymbol{\Delta\pi}^+(2)$ and $\boldsymbol{\Delta\pi}^-(2)$:

$$\begin{aligned} \boldsymbol{\Delta\pi}^+(2) &= [0.6375, 0, 0.0125, 0, 0.2] \\ \boldsymbol{\Delta\pi}^-(2) &= [0, 0.3, 0.05, 0.5, 0]. \end{aligned}$$

We can repeat the same reasoning for $\tau_2 = 2$, and in particular we obtain:

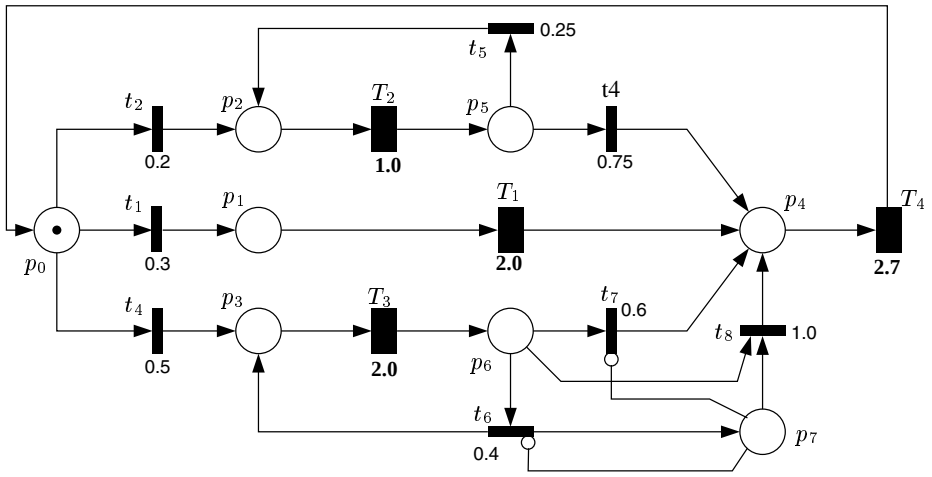

Figure 1. A D-DSPN model to illustrate the transient solution algorithm

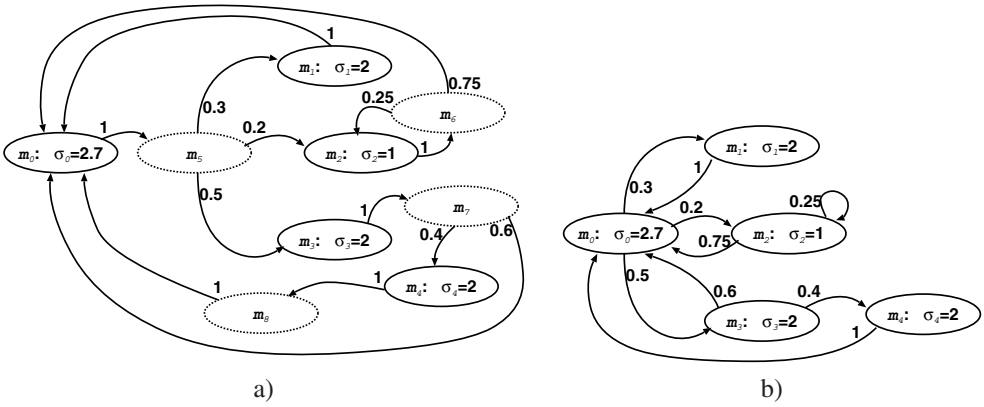

Figure 2. Reachability graph (a) and tangible reachability graph (b) of the D-DSPN of Figure 1

$$\mathcal{S}^{(2)} = \{\boldsymbol{m}_0, \boldsymbol{m}_2, \boldsymbol{m}_4\},$$

and

$$\boldsymbol{\Theta}^{(2)} = \{1, 2, 2.7\}.$$

We can determine the firing time instants τ_4, τ_5, and τ_6 (note that in the previous step we merge two firing time instants). In particular, if we apply Equation (11) (with some re-ordering of the terms $\tau_{(\cdot)}$) we obtain that

$$\begin{aligned}
\tau_3 &= 3 \quad \text{(this is due to a re-order of the terms)} \\
\tau_4 &= 3.7 \quad \text{(this is a time instant computed at the previous step)} \\
\tau_5 &= 4 \\
\tau_6 &= 4.7.
\end{aligned}$$

We can derive the sets $\mathcal{B}_3, \mathcal{B}_5$, and $\mathcal{B}_6$. Note that we do not derive the set $\mathcal{B}_4$ because it has been derived at the previous step (at the previous label, before the re-labeling, this set has been denoted as $\mathcal{B}_3$). In particular we have that: $\mathcal{B}_3 = \{\boldsymbol{m}_2\}$, $\mathcal{B}_5 = \{\boldsymbol{m}_4\}$, and $\mathcal{B}_6 = \{\boldsymbol{m}_0\}$. The corresponding probability increments/decrements are:

$$\begin{aligned}
\boldsymbol{\Delta\pi}^{+}(3) &= [0.009375, 0, 0.003125, 0, 0] \\
\boldsymbol{\Delta\pi}^{-}(3) &= [0, 0, 0.0125, 0, 0] \\
\\
\boldsymbol{\Delta\pi}^{+}(5) &= [0.2, 0, 0, 0, 0] \\
\boldsymbol{\Delta\pi}^{-}(5) &= [0, 0, 0, 0, 0.2] \\
\\
\boldsymbol{\Delta\pi}^{+}(6) &= [0, 0.19125, 0.1395, 0.31875, 0] \\
\boldsymbol{\Delta\pi}^{-}(6) &= [0.6375, 0, 0, 0, 0].
\end{aligned}$$

We can summarize the solution up to time $\tau \leq \tau_3 = 3$:

$$\boldsymbol{\pi}(\tau) = \left\{ \begin{array}{c|c|c|c} 0 \leq \tau < 1 & 1 \leq \tau < 2 & 2 \leq \tau < 3 & 3 \\ \hline 0 & 0.15 & 0.7875 & 0.796875 \\ 0.3 & 0.3 & 0 & 0 \\ 0.2 & 0.05 & 0.0125 & 0.003125 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0.2 \end{array} \right.$$

Figure 3 presents a graphical representation of the solution process. The arrows in this figure show how the probability flow among the states. The circle at time $\tau_2 = 2$ shows how the probabilities coming from different states are merged together. In the example, the value 0.6375 is obtained by adding the terms $0.3 \cdot 1 + 0.5 \cdot 0.6$ in the matrix multiplication, and 0.0375 when merging τ_2 with τ_3.

2.4. The algorithm

The technique proposed in Section 2.2 can be efficiently implemented by using a structure similar to a standard *discrete event simulation* program. Each firing time instant τ_k can be considered as a discrete event of a simulation. Since the analysis of each τ_k produces new τ_j with $j > k$, that may require the rearrangement of the previous τ_u with $\tau_u > \tau_j$, this can be considered as the scheduling of new events. However, with respect to a standard discrete event simulator, in the implementation of our technique, events can only be reordered and not removed.

The state of the system is composed by the three vectors: the probability distribution $\boldsymbol{\pi}(\tau_k)$, and the probability increments/decrements $\boldsymbol{\Delta\pi}^+(\tau_k)$ and $\boldsymbol{\Delta\pi}^-(\tau_k)$. Another difference with respect to a standard simulator is that this scheduler must join events if the new τ_j is equal to some $\tau_j = \tau_u$ for some $u > k$, and the old event must be update to consider also the new contribution. The state of the algorithm is a probability distribution vector. In this manner the scheduler does not consider a single path of the state space at a time as in the case of classical discrete event simulator, but it is exploring all the possible paths of in parallel.

Assume that we need to compute the transient solution up to time τ_{max}. All the events that occur after this time threshold can be discarded and we do not insert them into the event list. Table 2 presents a description of the basic steps of our transient solution algorithm.

We used an object-oriented like syntax to define the procedure. The constructor *orderedDoubleLinkedList* creates a double linked list, ordered according the time parameter τ. The elements of the double linked list are composed by two parameters: $\boldsymbol{\Delta\pi}^+$ and $\boldsymbol{\Delta\pi}^-$. Method *get* finds the element at time τ if present, or return *null* otherwise. Method *newNode* creates a new node for the list, and method *insert*, insert it into the list, by preserving the order with respect to τ. The procedure *insertIntoEventList* first checks if the event that it is going to schedule happens before the end of the transient computation, and than it verifies if there is other event occurring at the same time instance. If there is no other event with the same time instance the procedure insertIntoEventList creates a new event and schedule it at that time. On the other hand if there is another event with the same time instance the procedure does not insert the new event but it simply updates the event already scheduled.

In the procedure *generateNewEvents* that generates the new events due to the transitions enabled in the states that have a probability greater than zero in $\boldsymbol{\pi}$, $\boldsymbol{u}^{(i)}$ is a square matrix with the element $u_{ii} = 1$ and all the other elements equal to 0. This matrix is used to derive a vector that as its i-th component equal to π_i, and all other components equal to zero. The procedure *generateNewEvents* implicitly generates the sets $\mathcal{S}^{(k)}$, $\boldsymbol{\Theta}^{(k)}$, and $\mathcal{B}_k$.

The complexity of the proposed technique is under study. We only have some initial considerations on this issue. In particular we can say that the computational complexity is mainly dominated by three parameters: the first paramenter is $\tau_{\max}$. The impact of $\tau_{\max}$ on the complexity of solution is quite trivial: it forces a limit on the number of time instances that must be considered. Another parameter that influences the computational complexity is the length of the event list. This length depends on $\tau_{\max}$ and on the relations among the deterministic transition delays. The last parameter that influences the computational complexity is the number of new events generated when a given event is scheduled. The procedure *generateNewEvents* generates a new event for each non-zero entry on a specific row of matrix $\boldsymbol{C}$. In principle we can generate a number of new events equal to the state space size, in practice most of the row of matrix $\boldsymbol{C}$ have very few non-zero entries.

3. Numerical Experiments

In this section we present some numerical experiments to compare the performance of the proposed transient solution algorithm with other methods that allow to solve the class of D-DSPN models.

Example 1 We consider a pharmaceutical manufacturing system. In this field, common policies in manufacturing lines are generally determined by specific rules (for instance rules determined by the *Food and Drug Administration* [7]). For example, there can be some faults in the equipment that compromise the sterilization process and in these cases the product contained in a buffer is no longer "safe" and all the content of the buffer should be discarded. A simple D-DSPN model of a pharmaceutical production line is depicted in Figure 4 where we consider a machine that produces nUnits of product (firing of deterministic transition Start). All these product units have to pass a quality test that allows to recognize the corrupted units (immediate transitions nerr and err). There can be two different types of faults: a soft fault (immediate transition softerr) and a more serious error in the sterilization process (immediate transition hderr). The former corresponds to a recoverable fault and in this case the corrupted unit of product can

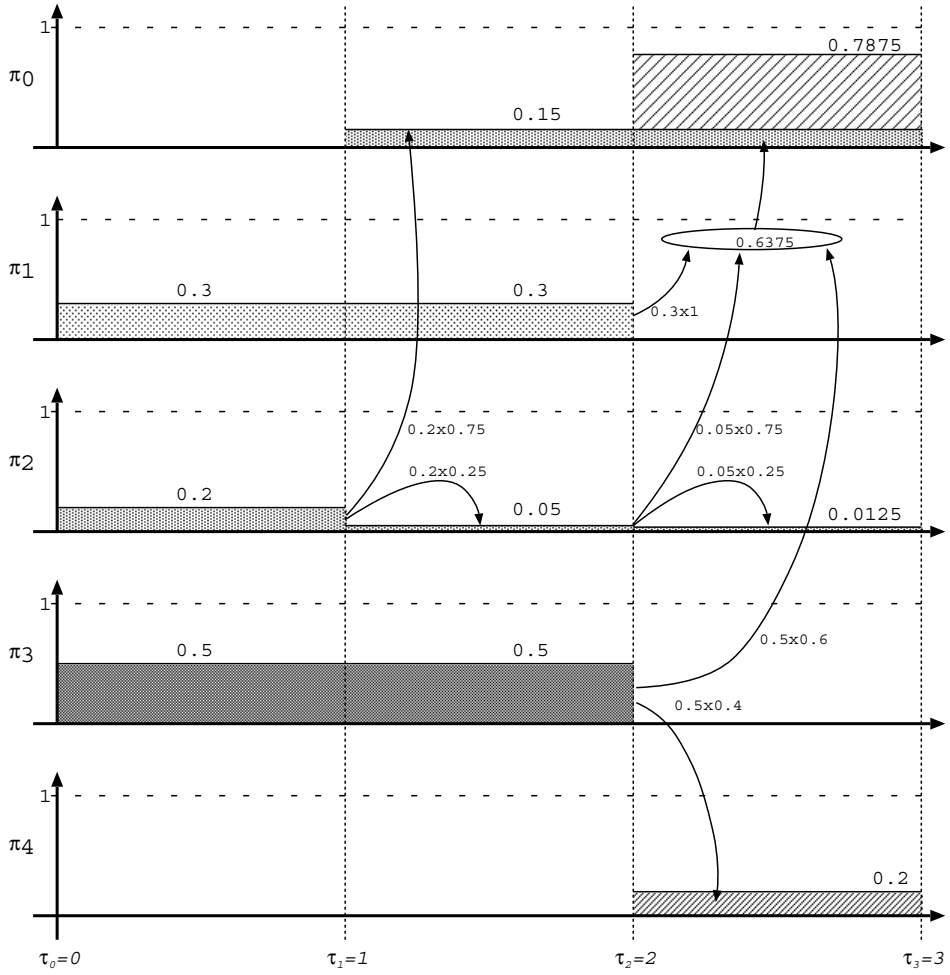

Figure 3. Graphical representation of the solution process (up to the time instant τ_3)

be "repaired" by means of a recovery phase (timed transition SEmng). In case of error in the sterilization process all the nUnits of product produced during the cycle are no longer safe, all the content of the buffer (places P2 and P3) should be discarded, and the machine requires a "restoration" phase (deterministic transition HEmng). The duration of this restoration phase depends on the number of consecutive errors in the sterilization process (number of tokens in place MxHE). The machine ends its cyclic activities when it accumulates at least NCyc units of product in the place Acc. When this event occurs transition tEnd may fire.

To evaluate the performance of the proposed transient algorithm we compute the completion time distribution of a machine-phase, this is represented by the probability that place End is marked. Table 3 contains the the delays of the deterministic transitions, and weights and priorities for the immediate transitions.

For our experiments we set nParts equal to 10, and ME equal to 3, that is, the maximum number of consecutive ster-

Transition label	Weight/Delay	Priority
nerr	49	2
err	1	2
softerr	9	3
hderr	1	3
endC	1	1
tEnd	1	5
limHE	1	10
Start	1	
SEmng	10	
HEmng	$1000 \times 2^{\#\mathsf{MxHE}-1}$	

Table 3. Delays of the deterministic transitions, and weights and priorities for the immediate transitions

```
procedure solveDDSPN(C, σ, π_0, τmax)
    initEventList()
    τ = 0
    π(τ) = π_0
    generateNewEvents(τ, π_0, C, σ)
    while (eventList.notEmpty() AND τ < τmax )do
        E = eventList.getFirstEvent()
        τ = E.τ
        π(τ) = π(τ)+ E.Δπ+ − E.Δπ−
        generateNewEvents(τ, E.Δπ+, C, σ)
        eventList.removeEvent(E)
    end while
end procedure solveDDSPN
```

```
procedure initEventList()
    eventList = new orderedDoubleLinkedList(τ, Δπ+, Δπ− )
end procedure initEventList
```

```
procedure generateNewEvents(τ, π, C, σ)
    for each i such that π_i > 0 do
        π− = π u^(i)
        π+ = π C
        insertIntoEventList(τ + σ_i, π+, π− )
    end for
end procedure generateNewEvents
```

```
procedure insertIntoEventList(τ, Δπ+, Δπ− )
    if (τ ≤ τ_max ) then
        if (E = eventList.get(τ)) ≠ null then
            E = eventList.newNode()
            E.τ = τ
            E.Δπ+ = Δπ+
            E.Δπ− = Δπ−
            eventList.insert(E)
        else
            E.Δπ+ = E.Δπ+ + Δπ+
            E.Δπ− = E.Δπ− + Δπ−
        end if
    end if
end procedure insertIntoEventList
```

Table 2. Basic steps of the transient solution algorithm

ilization errors that increases the duration of "restoration phase" is equal to 3 after this value the time required by this phase does not increase any longer. For NCyc we use the following values: 100, 300, 500, 700, 1000, and 10000. All the experiments were performed on a Pentium IV (2.4 Ghz) and a 1.5 Gbytes of memory.

Table 4 summarizes the comparison between the transient algorithm proposed in this paper and the one used by the package *TimeNET*. In all the experiments the measure (probability that place End is marked) is computed up to a time $t = 2000$.

As can be observed by the results presented in Table 4 the transient solution algorithm presented in this paper is much faster that the one implemented in TimeNET. We have to point out that this kind of comparison is not too fair. The transient solution method implemented in TimeNET is much more general that the one that we present in this paper because it can be used for DSPNs with exponential, deterministic (or generally distributed) transitions. Nevertheless, to the best of our knowledge, we compare our proposal with the only package that implements a transient solution method that is able to manage this sub-class of DSPNs.

NCyc	State Space Size	New algorithm (sec)	TimeNET (sec)
100	471	0.10	343.8
300	1411	0.29	432.9
500	2351	0.51	574.3
700	3291	0.66	754.3
1000	4701	0.95	1032.7
10000	47001	9.56	16752.6

Table 4. Comparison time between the new transient algorithm and that one used the package *TimeNET*

Example 2 We also perform another set of experiments by using the D-DSPN model of Figure 5. This is a D-DSPN presented in [4]. The model allows to compute the completion time distribution of finite TCP connections. In the present paper we do provide a detailed explanation of the model of Figure 5, interested readers can found all the details in [4].

The DSPN model presented in [4] belongs to the same

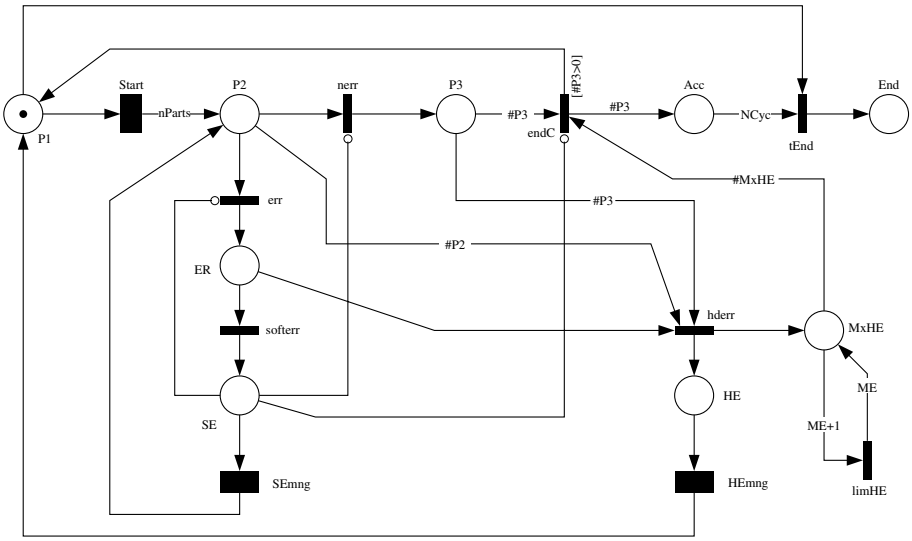

Figure 4. A D-DSPN that models a pharmaceutical manufacturing system

class of model that can be managed by our new transient algorithm, i.e., it has only deterministic and immediate transitions. The deterministic transitions represent the different timings that a TCP source has to manage: the Round Trip Time (RTT), the initial the Retransmission Time Out (I_RTO), and the estimated Retransmission Time Out (E_RTO). The TCP model requires the following input parameters: RTT, I_RTO, number of packets that the TCP source has to transmit (NPackets), and packet loss probability (Ploss). All the details concerning this TCP model can be found in the paper [4], in the present paper we use this model (with different values for NPackets) to compare the performance of the our new transient algorithm with the transient solution method implemented in TimeNET.

All the experiments have been performed by using the following set of parameters: $RTT = 0.129137$ (delay of transitions RTT and Td_Time), the delay of transition To_Time is defined by the following equation by using a marking dependent firing time for transition *TO_TIME* defined in the following manner:

$$\text{f.time } TO\_TIME = \begin{cases} 2^{(\#\text{TIMEOUT\_CNT}-1)}\, T_{out} & \text{if } \#\text{PCK\_SENT} = 0 \\ 2^{(\#\text{TIMEOUT\_CNT}-1)}\, T_{out\_RTT} & \text{if } \#\text{PCK\_SENT} > 0, \end{cases}$$

where $T_{out\_RTT} = RTT$ while $T_{out} = 6.0$.

Table 5 summarizes the comparisons between the transient algorithm implemented by TimeNET package and the one we propose in this paper.

4. Conclusions and Further Developments

In this paper we presented a new algorithm for the transient solution of a sub-class of DSPNs comprising only deterministic and immediate transitions and such that in each tangible marking only one deterministic transition is enabled. Although a formal derivation of the computational complexity of such algorithm is under investigation, the proposed method is in general, three order of magnitude faster that the transient solution algorithms that can compute the transient solution for D-DSPNs.

NPackets	State Space Size	New algorithm (sec)	TimeNET (sec)
8	158	0.01	2.53
10	229	0.01	2.66
30	1799	0.08	5.84
50	5052	0.25	29.66
80	14375	0.83	239.87
100	23372	1.28	812.14
120	34479	1.91	2460.40
150	53125	2.93	8388.66

Table 5. TCP model presented in [4]: Comparison time between the new transient algorithm and that one used the package *TimeNET*

References

[1] M. Ajmone Marsan and G. Chiola. On Petri Nets with Deterministic and Exponential Distributed Firing Times. In G. Rozenberg, editor, *Advances in Petri Nets*, LNCS, N. 266, pages 132–145. Springer Verlag, 1987.

[2] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Transient Analysis of Deterministic and Stochastic Petri Nets. In *Proc. of*

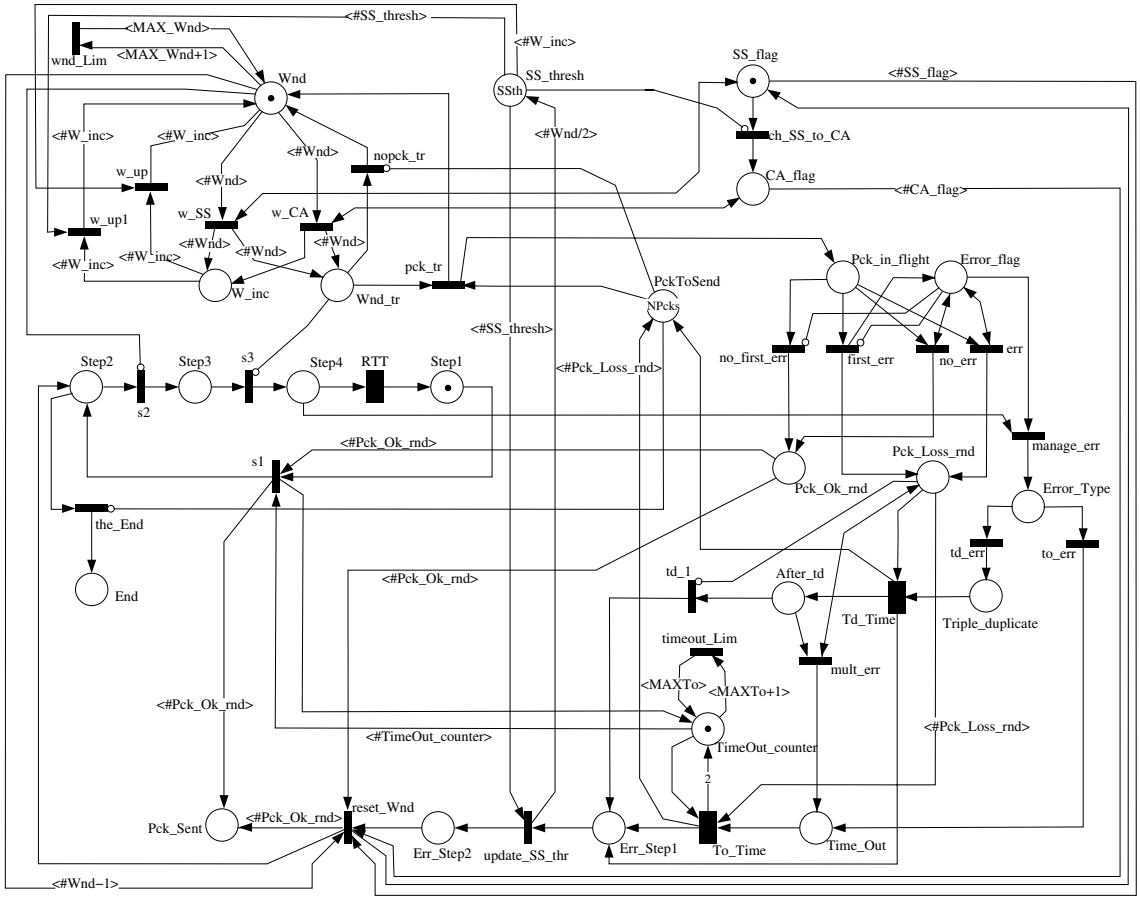

Figure 5. D-DSPN model of a TCP source

the 14th Int. Conf. on Application and Theory of Petri Nets, Springer Verlag.

[3] G. Ciardo. Discrete-time Markovian Stochastic Petri Nets. In *In Proc. of the 2-nd International Workshop on Numerical Solution of Markov Chains*, pages 339–358, 1995.

[4] R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno. On the Use of Petri Nets for the Computation of Completion Time Distributon for Short TCP Transfers. In *Proc. of 24th Application and Theory of Petri Nets Intern. Conference*, Springer Verlag.

[5] R. German. *Performance Analysis of Communication Systems: Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

[6] A. Heindl and R. German. A Fourth-Order Algorithm with Automatic Stepsize Control for the Transient Analysis of DSPNs. *IEEE Trans. on Soft. Eng.*, 25(2), March/April 1999.

[7] R. Howard and P. E. Leary. Strategies in Pharmaceutical Packaging Line Automation. *Pharmaceutical Engineering*, 6(5):8–18, 1995.

[8] C. Lindemann. *Performance Modelling with Deterministic and Stochastic Petri Nets*. John Wiley and Sons, 1998.

[9] C. Lindemann and A. Thummler. Transient Analysis of Deterministic and Stochastic Petri Nets with Concurrent Deterministic Transitions. *Performance Evaluation*, 36 & 37, 1999. Special Issue Proc. of PERFORMANCE '99.

[10] L. Lindemann and G. S. Shedler. Numerical Analysis of Deterministic and Stochasitc Petri Nets with Concurrent Deterministic Transitions. *Performance Evaluation*, 27 & 28:565–582, October 1996.

[11] A. Zimmermann, R. German, J. Freiheit, and G. Hommel. TimeNET 3.0 Tool Description. In 8^{th} *Intern. Workshop on Petri Nets and Performance Models*, Zaragoza, Spain, Sep 1999. IEEE-CS Press.