Correctness Verification and Performance Analysis of Real-Time Systems Using Stochastic Preemptive Time Petri Nets

Giacomo Bucci, Member, IEEE, Luigi Sassoli, and Enrico Vicario, Member, IEEE

Abstract—Time Petri Nets describe the state of a timed system through a marking and a set of clocks. If clocks take values in a dense domain, state space analysis must rely on equivalence classes. These support verification of logical sequencing and quantitative timing of events, but they are hard to be enriched with a stochastic characterization of nondeterminism necessary for performance and dependability evaluation. Casting clocks into a discrete domain overcomes the limitation, but raises a number of problems deriving from the intertwined effects of concurrency and timing. We present a discrete-time variant of Time Petri Nets, called stochastic preemptive Time Petri Nets, which provides a unified solution for the above problems through the adoption of a maximal step semantics in which the logical location evolves through the concurrent firing of transition sets. We propose an analysis technique, which integrates the enumeration of a succession relation among sets of timed states with the calculus of their probability distribution. This enables a joint approach to the evaluation of performance and dependability indexes as well as to the verification of sequencing and timeliness correctness. Expressive and analysis capabilities of the model are demonstrated with reference to a real-time digital control system.

Index Terms—Real-time reactive systems, preemptive scheduling, correctness verification, performance and dependability evaluation, discrete time, maximal step semantics, confusion, well definedness, stochastic preemptive Time Petri nets.

1 INTRODUCTION

In the construction of real-time reactive systems, development complexity is largely biased by nonfunctional requirements addressing ordered sequencing of events, bounded stimulus-response delay, efficient utilization of resources. Due to the criticality of applications involved, this complexity is often exacerbated by the need to guarantee correctness and quality of service with a high or even complete degree of dependability [40], [52], [39].

In the classical literature of real-time systems, restrictive assumptions on the structure of processes circumvent the problem of logical sequencing and enable efficient analytical techniques for the estimate of the worst case response time [24]. Unfortunately, this kind of analysis does not cover complex (but often realistic) systems with such characteristics as: nondeterministic computation times; sporadic tasks and/or tasks with mutual dependencies in the time of release; intertasks dependencies due to mutual exclusion on shared resources or to dataflow precedence relations; internal sequencing of tasks; multiple processors. Under such conditions, the need for predictability can motivate the complexity of state-space analysis techniques based on models such as Timed Automata [2], [6], [36], Time Petri Nets [9], [56], Process algebras [5], [48], [11]. As a common trait, in all these models, the state is comprised of a logical location and a set of clocks.

If clocks take values in a dense domain, the state space must be covered using state classes, each associating a logical location with a dense variety of clock valuations. Properties pertaining to the reachability of individual states, to the firing of execution sequences, and to the variety of timings that can be associated with a firing sequence can then be derived from the relation of reachability among classes. The theory has been largely developed both for Timed Automata [2], [6], [36] and for Time Petri Nets [10], [9], [17], [56]. However, the use of state classes poses severe limitations for model expressivity. In particular, analysis techniques based on state classes are difficult to be efficiently extended to deal with systems where clocks may be suspended and then resumed, as occurring in preemptive behavior [18], [20], [46], [26], [38]. More importantly for the focus of this paper, enumeration methods based on state classes for dense time have not been sufficiently developed yet to deal with a stochastic characterization of nondeterministic time parameters [22].

The assumption of a discrete time model overcomes the need to rely on state classes, thus smoothing the contrast between expressivity and analyzability. In particular, it largely eases the introduction of a stochastic characterization of time parameters and logical choices, which permits integration of correctness verification with dependability and performance evaluation [5], [31], [12], [15], [41], [19].

In [5], [43], a discrete time process algebra is used to model a set of recurring real-time tasks scheduled on a set of resources under various disciplines of practical interest (e.g., fixed priority and earliest deadline first) and to
support schedulability analysis and verification of safety properties [50]. The approach is extended in [44] to represent a failure rate for resources and to evaluate the probability that a deadline is missed due to this kind of failure. However, stochastic characterization does not cover the variability of temporal parameters such as task release times and computation times.

In [31], [12], [15], discrete phase types are used in a Petri Net formalism to provide a stochastic characterization for temporal parameters constrained within a finite support. This enables Markovian analysis in models where the mutual timing of events may yield implicit constraints on the logical sequencing. The use of a Kronecker algebra avoids the need for explicit enumeration of timed states, but requires that the untimed model have a finite number of markings. This rules out those systems where finiteness of behavior depends on sequencing constraints induced by timing (e.g., systems with bounded arrival rate or with timeout protocols).

The limitation is avoided in [59] by affording an explicit enumeration of individual states of a discrete time stochastic Petri Net so as to support a combined approach to schedulability analysis and performance analysis in complex tasking sets running under Rate Monotonic scheduling. However, the explicit enumeration cannot be applied in the presence of stochastic temporal parameters taking values over an infinite support, even in the case of a memoryless (geometric) distribution.

In [41], an analysis technique is proposed which aims at producing a discrete time approximation of the stochastic behavior of a model with non-Markovian dense time parameters. To this end, a discrete time Markov chain is derived by enumerating states comprised of a marking and a phase within a discrete phase type. In the implementation, this is obtained by first enumerating the markings of the untimed model and then expanding them so as to distinguish the different phases in which each marking can be reached [13].

In this paper, we present a Petri Net modeling formalism and an analysis technique which support both the verification of correctness and the evaluation of performance and dependability in real-time systems. The approach assumes a discrete model of time, which permits to manage a stochastic characterization of nondeterministic timed behavior and to encompass the expressivity requested by the application context of real-time systems, including preemptive behavior and flexible computations [25], [42].

The modeling formalism, that we call stochastic preemptive Time Petri Nets (spTPN), is characterized by the use of a step semantics. According to this, the logical location evolves through the concurrent execution of transition sets rather than through the interleaved firing of individual transitions assumed in most timed extensions of Petri Nets [31], [12], [59]. The use of a step semantics provides a unified solution for a number of complex effects deriving from the combination of logical concurrency and quantitative discrete timing, and relieves the designer from the burden of coping with stochastic underspecification of the model [51], [32], [54].

The analysis technique relies on the concept of stochastic state, which integrates the enumeration of a succession relation among sets of timed states with the calculus of their probability distribution. This permits the construction of a transition system labeled with durations and probabilities of events which opens the way to Markovian evaluation of performance and dependability indexes as well as to the verification of correctness in the sequencing of events and in their durational properties.

The rest of the paper is organized in five sections: The modeling formalism is introduced in Section 2 and the relevant characteristics of its semantics are discussed in Section 3 with reference to other timed variants in the literature of Petri Nets. The analysis technique is presented in Section 4. In Section 5, a case study in the context of real-time control systems is reported to demonstrate the expressive capabilities of the model, the complexity of the resulting analysis process, and the kind of results that this can provide. Conclusions are drawn in Section 6.

2 Stochastic Preemptive Time Petri Nets

Stochastic preemptive Time Petri Nets (spTPN) develop Time Petri Nets [47], [9] and preemptive Time Petri Nets (pTPN) [20], [18] to obtain a discrete time formalism enabling stochastic modeling of real-time systems: transition delays are characterized through a discrete probability mass function; nondeterministic choices can be conditioned through stochastic switches; progress of timed transitions depends on preemptable resources, assigned according to priorities that can change with the marking; the model evolves through the concurrent firing of transition sets (maximal step semantics).

2.1 Syntax

A stochastic preemptive Time Petri Net is a tuple:

\[
spTPN = (P; T; A^-; A^+; A^*; M; Res; Req; Prio; D; \mathcal{C}).
\] (1)

- \(P\) and \(T\) are disjoint sets of places and transitions, respectively. \(A^-\), \(A^+\), and \(A^*\) are sets of preconditions, postcondition and inhibitor arcs connecting places and transitions \((A^- \subseteq (P \times T), A^+ \subseteq (T \times P),\) and \(A^* \subseteq (P \times T))\). A place \(p\) is said to be an input, an output or an inhibiting place for a transition \(t\) if there exists a precondition, a postcondition or an inhibitor arc connecting \(p\) and \(t\), respectively. \(M\) is a marking associating each place with a natural number \((M : P \rightarrow \mathbb{N})\).

- \(Res\) is a set of resources disjoint from \(T\) and \(P, Req\) associates each transition with a set of requested resources \((Req : T \rightarrow 2^{\mathbb{R}es})\) called resource request. \(Prio\) associates each transition with a priority level which may depend on the marking \((Prio : T \times \mathbb{N}^{|P|} \rightarrow \mathbb{N})\). For simplicity, we assume here that two transitions cannot have the same priority level under any marking \(\forall M \in \mathbb{N}^{|P|}, t_1 \neq t_2 \rightarrow Prio(t_1, M) \neq Prio(t_2, M))\).
• $D$ associates each transition $t$ with a static probability mass function $D_t$, possibly defined over a nonfinite support. The extrema of the support of $D_t$ are often referred to as static earliest and latest firing time, denoted as $EFT^*(t)$ and $LFT^*(t)$, respectively.

• $C$ is a competitiveness function associating each transition with a natural number ($C : T \rightarrow \mathbb{N}$).

2.2 Semantics

The state of an spTPN is a pair $s = (M, ttf)$, where $M$ is a marking and $ttf$ is a vector associating each transition $t$ with a discrete time to fire $ttf_t \in \mathbb{N}$. The state evolves according to a state transition rule defined through three clauses of firability, step selection, and progress.

2.2.1 Firability

A transition is enabled if each of its input places contains at least one token and none of its inhibiting places contains any token. An enabled transition $t_o$ is progressing iff every resource in $Req(t_o)$ is not in the resource request $Req(t_1)$ of any other enabled transition $t_1$ with higher priority

$$(\forall r \in Req(t_o), \forall \text{ enabled } t_1, Prio(t_1, M) > Prio(t_o, M) \rightarrow r \not\in Req(t_1)).$$

Transitions that are enabled but not progressing are said to be suspended. A progressing transition $t$ is firable iff its time to fire $ttf_t$ is equal to 0. The set of firable transitions is called attempting set.

2.2.2 Step Selection

If the attempting set is empty, time advances by one tick. If the attempting set is not empty, the set of transitions that will actually come to fire, that we call firing set, is derived through a repetitive stochastic selection: Until the attempting set is not empty, one of its transitions is randomly selected, it is removed from the attempting set, and it is added to the firing set iff none of its input places is an input place for any other transition already added to the firing set (what we call a conflict). At each repetition, the probability to select a transition $t$ is made equal to the competitiveness of $t$, divided by the sum of competitiveness values of all the transitions still contained in the attempting set:

$$\text{Prob}(t_s) = \frac{C(t_s)}{\sum_{t \in \text{Attempting set}} C(t)}.$$

2.2.3 Progress

If time advances, the marking and the time to fire of every suspended transition remain unchanged, while the time to fire of every progressing transition is reduced by one time unit.

If a set $\{t_n\}_{n=0}^{N-1}$ fires, the marking $M$ is replaced by a new marking $M'$ which is obtained by removing a token from each input place of each transition in the firing set, and by adding a token to each output place of each transition in the firing set. In the transformation, an intermediate marking $M_{imp}$ is also derived to capture the condition subsequent to the removal of tokens from input places but prior to the addition of tokens in output places:

$$M_{imp}(p) = M(p) - 1 \quad \forall p, (p, t_n) \in A^-$$
$$M'(p) = M_{imp}(p) + 1 \quad \forall (p, t_n, p) \in A^+.$$ (2)

Transitions that are enabled both by the intermediate marking $M_{imp}$ and by the final marking $M'$ are called persistent, while those that are enabled by $M'$ but not by $M_{imp}$ are said newly enabled. Any transition belonging to the firing set which is still enabled after its own firing is always regarded as newly enabled. The time to fire of persistent transitions remains unchanged, while the time to fire of each newly enabled transition $t$ takes a nondeterministic value sampled according to the static probability mass function $D_t$. The time to fire $ttf_t$ of any transition $t$ that is not enabled by the new marking is not relevant for the state of the net as it will be reset to a new value as soon as $t$ will be enabled again.

The state transition rule defines a direct reachability relation $\xrightarrow{ev,\mu}$ between states, where $\mu$ is a measure of probability and $ev$ is either the advancement of time (tick event) or the firing of a set of transitions (firing event). Given two states $s$ and $s'$, we write $s \xrightarrow{ev,\mu} s'$ when $ev$ can occur in $s$ and lead to $s'$ with probability $\mu$. Note that the measure of probability $\mu$ can be different than 1 only when $ev$ is a firing. In this case, its value depends on the resolution of possible conflicts within the attempting set as well as on the determination of the times of fire assumed by newly enabled transitions.

2.3 Remarks

A few remarks emphasize the salient traits that distinguish spTPNs with respect to other timed extensions of Petri Nets.

2.3.1 Maximal Step Semantics

The state of an spTPN evolves through steps each consisting of the concurrent execution of a set of progressing transitions (the firing set). Steps are maximal as a progressing transition cannot avoid the firing unless its time to fire is not null or it has a conflict with some transition in the firing set. This marks a difference with respect to most timed extensions of Petri Nets where the firing of multiple transitions sharing a common time to fire occurs through the interleaved firing of individual transitions.

The set of transitions that attempt the firing is univocally determined by the marking and the vector of times to fire. If the attempting set does not contain any conflict, all attempting transitions come to the firing in a single step. Only in the presence of conflicts, determination of the firing set is conditioned by competitiveness values of attempting transitions.

The definition of a maximal step semantics for a model with nonMarkovian timed transitions is an original aspect of spTPNs. Previous works [58], [23] have addressed the case of transitions firing either in zero time or after a geometrically distributed delay; this setting drastically simplifies the analysis as the state does not depend on time, but prevents the representation of nondeterministic delays comprised within a finite interval, which are crucial in modeling precedence relations induced by timed
behavior. The case of multiple transitions attempting the firing within the same time slot in a discrete timed model is explicitly addressed in [41]. However, since the concurrency of the model is not explicitly formulated as a step semantics, for each firing set, the feasibility with respect to conflicts, the probability of occurrence, and the resulting state are determined through the analysis of intermediate states reached according to an interleaving semantics [15]. This yields a different behavior with respect to that of spTPNs, as discussed in Section 3.3. Moreover, in [41], any subset of the attempting set is considered firable, thus admitting the case of a transition which does not attempt the firing though its time to fire has a null probability to be higher than 0. This results in nonmaximal steps impairing the property of nonnull defer, later introduced in Section 3.1, that relieves spTPNs from the effects of confusion behaviors.

### 2.3.3 Resources, Priorities, and Preemptive Behavior

resources which can give continuity to the flow of a service.

jobs waiting to be started or for a multiplicity of stand-by

time analysis.

approach which would drastically increase the complexity

which enable the transition instance. We avoid this

time to fire depending on the age of the binding of tokens

same transition should be associated with an individual

nonexponential delays, each concurrent instance of the

newly enabled, as usual in Time Petri Nets [9], [56]. In fact,

motivates the assumption that a transition which is still

sufficient for multiple firings (multiple enabledness). This

provides the property of well definedness discussed in Section 3.2.

This limitation prevents considering the concurrent progress of multiple instances of the same transition (multiple service) even when tokens in its input places are sufficient for multiple firings (multiple enabledness). This motivates the assumption that a transition which is still enabled after its own firing must always be considered as newly enabled, as usual in Time Petri Nets [9], [56]. In fact, in order to combine a multiple server semantics with nonexponential delays, each concurrent instance of the same transition should be associated with an individual time to fire depending on the age of the binding of tokens which enable the transition instance. We avoid this approach which would drastically increase the complexity of timed analysis.

In the lack of a multiple server semantics, multiple tokens in a place can still serve to account for a queue of jobs waiting to be started or for a multiplicity of stand-by resources which can give continuity to the flow of a service.

### 2.3.2 Collective Token Philosophy, No Multiple Enabledness, and Single Service

spTPNs follow a so-called Collective Token Philosophy in which any two tokens in the same place are computationally indistinguishable [16]. In fact, the timing semantics is expressed with reference to clocks (the times to fire) associated with enabled transitions. Tokens determine the conditions of continuity in transition enabling, but they do not carry information about their individual time of permanence in a place.

This limitation prevents considering the concurrent progress of multiple instances of the same transition (multiple service) even when tokens in its input places are sufficient for multiple firings (multiple enabledness). This motivates the assumption that a transition which is still enabled after its own firing must always be considered as newly enabled, as usual in Time Petri Nets [9], [56]. In fact, in order to combine a multiple server semantics with nonexponential delays, each concurrent instance of the same transition should be associated with an individual time to fire depending on the age of the binding of tokens which enable the transition instance. We avoid this approach which would drastically increase the complexity of timed analysis.

In the lack of a multiple server semantics, multiple tokens in a place can still serve to account for a queue of jobs waiting to be started or for a multiplicity of stand-by resources which can give continuity to the flow of a service.

### 2.3.4 Timed and Immediate Transitions

In dense time models, the probability that two timed transitions fire at the same time instant has a null measure unless probability density functions of transition delays are dependent or they can yield a nonnull integral over an integration domain of null measure. In most practical applications, this confines the problem of contemporary firings to immediate transitions that are used to model causal connections (possibly with a deterministic delay) between events. Distinction between immediate and timed transitions in the semantics of the model enables ad hoc algorithms which simplify the analysis and separate the problems of concurrency and timing [45].

In discrete time models [31], [33], [15], [59], the firing of transitions is concentrated at discrete instants, and the problem of contemporary firings is extended to timed transitions. This basically reduces the advantage in distinguishing timed and untimed transitions. SpTPNs abolish this distinction permitting a direct modelling of events for which the probability to fire at time 0 after the enabling is strictly higher than 0 but strictly lower than 1. This is an expressive capability that may be relevant to smooth the limits of a discrete formalism in the modeling of asynchronous systems, especially when the analysis is oriented to verify properties of the logical sequencing.
Under the maximal step semantics of spTPNs, a progressing transition with null time to fire necessarily attempts the firing. In the attempt, it will either fire or be overtaken by some conflicting transition. But, in neither of the two cases, the transition can remain persistent and attempt the firing before a finite lapse of time has passed. This mechanism, that we call nonnull defer, rules out the case of an immediate firing sequence which includes choices between transitions that have been persistent since the beginning of the sequence and transitions that have become progressing with a null time to fire along the sequence itself.

Nonnull defer improves modeling convenience as it structurally overcomes the need for the designer to detect and rule implicit conflicts occurring within confusion behaviors. At the same time, it does not practically reduce expressive power as the model language still accepts behaviors where multiple events are serialized along a variety of interleaving sequences within a null lapse of time: this can be obtained with self-loop patterns which explicitly represent a constraint of serialization, as occurring for instance in the net of Fig. 4.

### 3.2 Well Definedness

Quantitative analysis for performance and dependability evaluation relies on a stochastic characterization of choices in the behavior of a model. In principle, this characterization can be partial, combining probabilistic and nondeterministic choices. This supports abstraction about decisions that cannot be stochastically characterized and still permits the derivation of some relevant measures, as for instance in the verification of probabilistic concurrent systems [55], [35].

However, in order to enable derivation of global properties through Markovian analysis, each choice must be characterized through a measure of probability for each feasible determination [32]. To accomplish this condition of well definedness, a model often needs to be extended, either with priorities which force the behavior and avoid choices, or with random switches which characterize them [30], [45], [32].

Under interleaving semantics, the effort of this extension is largely increased by the need to give a fictitious order to the execution of multiple noninterfering transitions. To confine the problem to those choices which actually impact on some relevant reward process, [30] proposes that the set of transitions that may be fired at the same time be partitioned in a number of noninterfering extended conflict sets (ECS). In so doing, choices that must be ruled are only those occurring within the same ECS. However, identification of ECS through structural analysis provides only a necessary condition, and management of false alarms requires additional specification by the designer. The difficulty is largely exacerbated by the fact that an ECS may include transitions that are enabled in different markings visited along a confusion behavior.

In [32], well definedness of a model is automatically verified during the generation of the underlying stochastic process of the net. In [54], sufficient conditions are derived using a structural analysis implemented within an interactive tool which prompts on potential choices that are
missing a probabilistic characterization and which guides the designer through an incremental extension of the model. However, in so doing, conflicts that correspond to an explicit modeling intention of the designer tend to be mixed with fictitious or unexpected conflicts arising in the interleaved sequencing of events. For the latter kind of choices, the designer is requested to express priorities or random switches which do not carry an apparent physical and intentional meaning (see Fig. 1b).

In the step semantics approach of spTPNs, choices occur only in the selection of a time to fire for a newly enabled transition or in the derivation of a firing set from an attempting set with conflicts. The former kind of choice is fully characterized by the probability function of the newly enabled transition. The latter is ruled by the stochastic selection process with the possible conditioning of competitiveness values. This separates the effects of explicit dependencies of mutual exclusion expressed by conflicts on input places from implicit dependencies induced by timing constraints of transitions.

3.3 Conflict and Concurrency

In analysis techniques based on an interleaving approach, concurrency can be partially recovered by classifying as concurrent a set of events which can be performed in any order, within a null lapse of time, and leading to the same final condition. This approach is applied to a timed model in [15], where a preanalysis of the untimed reachability graph is performed to associate each marking with the sets of concurrent firings (transitions that can fire simultaneously, leading from state $M_i$ to $M_j$ in a single step).

In an interleaving approach, the existence of both the instantaneous interleaving sequences $\{t_1\} \rightarrow \{t_2\}$ and $\{t_2\} \rightarrow \{t_1\}$ is interpreted so as to recover the concurrent behavior in which $t_1$ and $t_2$ are fired simultaneously, leading from state $M_0$ to $M_3$ in a single step.

The different orders of concurrent events depending on possible conflicts and nonmonotonic enabling conditions related to priorities and inhibitor arcs. These effects not only prevent complete reconstruction of all the states that are actually reachable in a step semantics (see Fig. 3), but also may detect concurrency between events that are mutually exclusive (see Fig. 4) [16].

4 State Space Enumeration and Analysis

We regard the evolution of an spTPN as a discrete time stochastic process.

The states of the process, that we call stochastic states to distinguish them from the states in the semantics of Section 2.2, are pairs $S = (M, \hat{P})$, where $M$ is a marking and $\hat{P}$ a vector of probability mass functions for the times to fire of transitions enabled by $M$.

The evolution across stochastic states is described through a succession relation $\Rightarrow$, where $ev$ is an event and $\mu$ a measure of probability. Given two stochastic states $S = (M, \hat{P})$ and $S' = (M', \hat{P})$, we write $S \xrightarrow{ev, \mu} S'$ iff the following property holds: If the marking of the net is $M$ and the vector of times to fire is a random variable distributed according to

Fig. 2. In an interleaving approach, the existence of both the instantaneous interleaving sequences $\{t_1\} \rightarrow \{t_2\}$ and $\{t_2\} \rightarrow \{t_1\}$ is interpreted so as to recover the concurrent behavior in which $t_1$ and $t_2$ are fired simultaneously, leading from state $M_0$ to $M_3$ in a single step.

Fig. 3. Under a step semantics, transitions $t_1$ and $t_2$ fire in a single step leading to marking $M_3$. Whereas, under an interleaving semantics, marking $M_3$ cannot be reached by any interleaved sequence of events as the firing of $t_1$ disables $t_2$ and viceversa.

Fig. 4. Transitions $t_1$ and $t_2$ have a structural conflict, but they can be executed in any order, within a null lapse of time, leading to the same marking. Following an interleaving approach and trying to recover concurrency from nondeterminism, a false concurrent behavior would be added leading directly from $M_0$ to $M_1$. 

The assumption of a step semantics not only avoids enumeration of fictitious states visited along interleaved sequences, but, more importantly, it eliminates a number of subtle side effects deriving from the attempt of concealing
\( \hat{P} \), then \( ev \) is a possible next event for the net, which occurs with probability \( \mu \), and which leads to a new marking \( M' \) and a new vector of times to fire distributed according to \( \hat{P} \).

While the state of an spTPN evolves through firing and tick events, in the enumeration of the succession relation among stochastic states, we also consider a third fictitious event that we call defer. This is an outcoming event for stochastic states where both the firing of a set of transitions and the advancement of time are possible, and accounts for the instantaneous choice that no firable transition will fire before the advancement of time. With the introduction of defer, the tick event is considered possible only in stochastic states where no transition is firable, thus avoiding stochastic states where both a firing and a tick are outcoming events. This partitionment associates each stochastic state with a sojourn time, equal to 0 if defer or firing are outcoming events, and equal to 1 if tick is the (unique) outcoming event.

### 4.1 Enumeration

Given an initial stochastic state \( S_0 \), the succession relation \( \overset{ev}{\Rightarrow} \) identifies a set of reachable stochastic states \( S \) and a timed stochastic step transition system \( STS = (S, S, \overset{ev}{\Rightarrow}) \). Enumeration of the \( STS \) requires algorithms for the detection of the outcoming events from a stochastic state, for the calculus of their probability, and for the derivation of successor stochastic states. These algorithms are different for each kind of event.

#### 4.1.1 Tick Event

Tick is an outcoming event from \( S = (M, \hat{P}) \) iff \( P_t(0) = 0 \) for every transition \( t \) progressing under marking \( M \). If this is the case, tick is the unique outcoming event from \( S \) and its probability \( P_{tick}(S) \) is equal to 1. In the transition from \( S \) to \( S' = (M', \hat{P}') \) through a tick event, the marking is not changed. Besides, the time to fire is maintained for each suspended transition and it is reduced by 1 for each progressing transition:

\[
P_t(n) = \begin{cases} P_t(n) & \text{if } t \text{ is suspended in } S \\ P_t(n + 1) & \text{if } t \text{ is progressing in } S. \end{cases}
\]

#### 4.1.2 Defer Event

Defer is an outcoming event for \( S \) iff \( P_d(0) < 1 \) for every progressing transition and \( P_d(0) > 0 \) for at least one progressing transition \( t_o \) (which means that every progressing transition \( t \) can have a time to fire higher than 0, and at least one progressing transition \( t_o \) can have time to fire equal to 0). If defer is an outcoming event, its probability is equal to:

\[
P_{defer}(S) = \prod_{t \in \text{firable}(S)} (1 - P_t(0)),
\]

where \( \text{firable}(S) \) denotes the set of transitions in \( S \) that are progressing and have a nonnull probability to fire at time 0.

In the stochastic state \( S' \) reached through a defer event, the marking remains unchanged while probability functions in the vector \( \hat{P} \) are updated differently for progressing and for suspended transitions. For each transition that was suspended in \( S \), the time to fire remains unchanged and so is its probability function; whereas, for each transition that was progressing in \( S \), the time to fire is replaced through a new stochastic variable conditioned to be higher than 0:

\[
P'_t(n) = \begin{cases} P_t(n) & \text{if } t \text{ is suspended in } S \\ 0 & \text{if } t \text{ is progressing in } S \text{ and } n = 0 \\ P_t(n) \star \frac{1}{1 - P_t(0)} & \text{if } t \text{ is progressing in } S \text{ and } n \neq 0. \end{cases}
\]

According to (5), in every stochastic state reached through a defer event, tick is the only possible outcoming event.

#### 4.1.3 Firing Event

The firing of a set \( \phi = \{t_n \}_{n=1}^{N-1} \) is an outcoming event for \( S \) iff \( \phi \) contains only progressing and nonconflicting transitions, every transition \( t \in \phi \) can fire before time advances, and every progressing transition \( t_{nc} \notin \phi \) that is not conflicting with \( \phi \) can be delayed, i.e.,

\[
P_{t(0)}(t) > 0 \quad \forall t \in \phi 
\]

\[
P_{t_{nc}(0)}(t) < 1 \quad \forall t_{nc} \notin \phi, t_{nc} \text{ progressing and not conflicting with } \phi.
\]

The probability \( P_\phi(S) \) that the firing set \( \phi \) is the outcoming event from \( S \) jointly depends on the probability that each and every transition in \( \phi \) attempts the firing and on the probability that the attempt is not impaired by conflicts with other transitions. For the derivation, \( P_\phi(S) \) can be decomposed as:

\[
P_\phi(S) = \sum_{\sigma \in \text{Att}(S)} P_{\phi|\sigma} \cdot P_\phi(S),
\]

where \( \text{Att}(S) \) denotes the set of attempting sets that are feasible in \( S \), \( P_\phi(S) \) is the probability that \( \sigma \) is the attempting set for a \( t \) distributed according to \( \hat{P} \), and \( P_{\phi|\sigma} \) is the probability that \( \phi \) is the firing set under the condition that \( \sigma \) is the attempting set.

Since times-to-fire of different transitions are independent variables, \( P_\phi(S) \) can be expressed as the product:

\[
P_\phi(S) = \prod_{t_i \in \sigma} P_{t_i(0)} \prod_{t_j \in \text{firable}(S) \setminus \sigma} (1 - P_{t_j(0)}),
\]

where \( \text{firable}(S) \setminus \sigma \) denotes the set of firable transitions that are not included in the attempting set.

Besides, \( P_{\phi|\sigma} \) can be decomposed so as to distinguish the different orderings in which transitions of the attempting set \( \sigma \) are sampled in the repetitive stochastic selection defined in the clause of firability:

\[
P_{\phi|\sigma} = \sum_{\text{Perm}(\sigma)} P_{\phi|\text{seq}} \cdot P_{\text{seq}|\sigma},
\]

where \( \text{Perm}(\sigma) \) is the set of permutations of the elements of \( \sigma \) and seq denotes any of such permutations. \( P_{\text{seq}|\sigma} \) is the probability that the transitions in the attempting set \( \sigma \) are selected in the order defined by seq (i.e., transition \( t_{seq} \) is selected before transition \( t_{seq+1} \)), which can be expressed as:
$$P_{\phi|\sigma} = \prod_{\text{seq} \in \sigma} \frac{C(t_{\text{seq}})}{\sum_{h=1}^{\text{seq}} C(t_{\text{seq}})}.$$  

(10)

Finally, $P_{\phi|\sigma}$ is equal to 1 iff every transition $t \in \phi$ appears in seq before any other transition which has a conflict with $t$ itself and it is equal to 0 otherwise.

For the sake of an efficient implementation, it is useful to note that $P_{\phi|\sigma}$ appearing in (7) does not depend on the stochastic state and can be computed only once along the enumeration of the entire state space. In fact, $P_{\phi|\sigma}$ is determined by the competitiveness factor $C$, which is not affected by the stochastic state, and $P_{\phi|\sigma}$ is determined by structural properties of the net.

In the stochastic state reached through the firing of a transition set, the marking is changed according to the firing clause of spTPNs. The components of the vector $\mathbf{P}$ are updated differently for transitions that are either newly enabled or persistent after the firing of $\phi$. For each newly enabled transition $t_\phi$, the time to fire is a new stochastic variable distributed according to the static probability function $D(t_\phi)$. For each persistent transition $t_w$ that was suspended in $S$, the time to fire remains unchanged. Finally, for each persistent transition $t_i$ that was progressing in $S$, the time to fire becomes higher than 0. In fact, under a maximal step semantics, the observation of a progressing transition $t_i$ with a random time to fire $t_f t_i$, which does not attempt the firing is equivalent to the assumption that $t_f t_i$ is higher than 0. Basically, this is the consequence of the nonnull defer property mentioned in Section 3.1.

According to this, the vector of times to fire of enabled transitions after the firing of $\phi$ is distributed according to $\mathbf{P}$:

$$P'(t) = \begin{cases} D(t) & \text{if } t \text{ is newly enabled} \\ P(t) & \text{if } t \text{ is persistent in } S' \text{ and suspended in } S \\ 0 & \text{if } t \text{ is persistent in } S' \text{ and progressing in } S \\ \frac{1}{1-P(t)} & \text{if } t \text{ is persistent in } S' \text{ and progressing in } S \text{ and } n > 0. \end{cases}$$

(11)

4.2 Boundedness

In general, enumeration of stochastic states is a semi-algorithm as termination is not guaranteed. However, under quite general conditions on the form of static probability mass functions, unboundedness of the stochastic step transition system $STS$ can only derive from an unbounded number of reachable markings:

**Theorem 1.** Assume that each transition $t$ has a static memory scope $k_t$ after which the static probability mass function $D(t)$ is either null or geometrically distributed:

$$\forall t \in T, \exists k_t \in \mathbb{N}_{>0}. (\forall k \geq k_t, D_t(k) = 0) \lor (\exists \lambda_t \in (0, 1), \forall k \geq k_t, D_t(k) = \lambda_t(1 - \lambda_t)^{k_t}).$$

If the $STS$ is unbounded, then it includes an infinite number of different markings.

**Proof.** Ab absurdo, assume that the $STS$ includes an unbounded number of stochastic states with a finite number of markings. This implies the existence of a transition $t$ which is enabled with different dynamic probability mass functions in an unbounded number of stochastic states. In any stochastic state in which $t$ is newly enabled, $P(t)$ is equal to $D_t$, and thus exists a dynamic memory scope $k_t^* = k_t$ such that after $k_t^*$, $P(t)$ is either null or geometrically distributed.

Starting from any stochastic state where $t$ is newly enabled, the form of $P(t)$ can evolve through the repeated application of the transformations corresponding to the events of firing, tick, and defer. However, in these transformations only a finite number of different forms for $P(t)$ can be generated. In fact: at every firing or defer, $P(t)$ is either reset (if $t$ takes part in the firing or if it is disabled by some fired transition) or it is conditioned so that $P(t) = 0$, in which case the form of $P(t)$ cannot change again until the next tick event; at every tick, the dynamic memory scope $k_t^*$ is decreased towards 0, so that only a finite number of ticks can occur before $P(t)$ becomes entirely concentrated in 0 (i.e., $P(t) = 0$) or it assumes a geometric form.

4.3 Correctness Verification

Properties pertaining to states and behaviors in the semantics of an spTN model can be recovered from the analysis of stochastic states and paths in the $STS$. To this end, a stochastic state $S$ can be regarded as a collection of states sharing a common marking but having different times to fire for enabled transitions. In this perspective, we say that a state $s = (M, \mathbf{P})$ is contained in a stochastic state $S = (M, \mathbf{P})$, and we write $s \in S$, when $m = M$ and $P(t) > 0$ for every enabled transition $t$.

The relation between states and stochastic states is made explicit in a contracted step transition system $STS' = (S, S_{root}, \rightarrow_{\mu})$, which is derived from $STS = (S, S_{root}, \rightarrow_{\mu})$ by neglecting the measure of probability $\mu$ and by concreasing the defer event so as to observe only those stochastic states that are reached through a tick or a firing:

$$S \in \tilde{S} \iff S \in S \text{ and } \exists S_0 \in S \text{ such that } S_0 \rightarrow_{\mu} S_0 \rightarrow_{\mu} S$$

$$S \in \langle S \rightarrow_{\mu} \rangle$$

$$S \rightarrow_{\mu} S \iff S \rightarrow_{\mu} S$$

$$S \rightarrow_{\mu} S \iff S \rightarrow_{\mu} S$$

The following two lemmas establish a bidirectional correspondence between the successor relation $\rightarrow_{\mu}$ between stochastic states in the contracted $STS$ and the reachability relation $\rightarrow_{\mu}$ between states in the semantics of an spTN model:

**Lemma 1.** If $s \in \tilde{S}$ and $s \rightarrow_{\mu} S$, then $S \rightarrow_{\mu} \tilde{S}$, with $s_0 \in \tilde{S}$.

**Proof.** If $s \rightarrow_{\mu} S$, then, for every transition $t_p$ progressing in $s$, $t_f t_i > 0$. Since $s \in \tilde{S}$, this implies $P(t_p) > 0$, which guarantees that $S$ accepts a tick as an occurring event.
If \( s^{\phi \mu} \rightarrow s_{\ast} \), then \( \phi \) includes only progressing and nonconflicting transitions, for every \( t_i \in \phi \), \( t_i\mu t_i = 0 \), and, for every \( t_i \notin \phi \) which is progressing and not in conflict with \( \phi \), \( t_i\mu t_i > 0 \). The assumption \( s \in \tilde{S} \) thus implies \( P^M_S(0) > 0 \) and \( P^M_S(1) < 1 \), which guarantees that \( \tilde{S} \) accepts \( \phi \) as an outgoing event.

By comparing the derivation of the marking and the times to fire of \( s \), from \( s \) against the derivation of the marking and the probability mass function of \( \tilde{S} \), from \( \tilde{S} \), respectively, it can be easily shown that the marking of \( s \) is equal to that of \( \tilde{S} \), and that every transition \( t \) enabled in \( s \) has a time to fire within the support of the probability mass function \( P^M_t(\cdot) \), which concludes the proof.

\[ \square \]

**Lemma 2.** If \( S^{ev} \xrightarrow{ev, \mu} \tilde{S} \), then \( \forall s_s \in \tilde{S}_s, \exists s_s \in \tilde{S}, \mu \in (0,1] \) such that \( s \rightarrow s_s \).

**Proof.** Let \( s_s = (M_s, \tau) \) be a state contained in \( \tilde{S} \).

If \( S^{\text{tick}} \rightarrow \tilde{S} \), consider the state \( s = (M, \bar{\tau}) \), where \( M \) is the marking of \( s \) and \( \bar{\tau} \) is a vector of times to fire for the transitions enabled by \( M \) defined as follows:

\[ \tau_i = \begin{cases} \tau_i^t + 1 & \text{if } t \text{ is progressing in } \tilde{S} \\ \tau_i^s & \text{if } t \text{ is suspended in } \tilde{S}. \end{cases} \quad (13) \]

Since in \( \bar{\tau} \) the time to fire of any progressing transition is higher than 0, tick is the next event from \( s \) and, by construction, it leads to \( s_s \), i.e., \( S^{\text{tick}} \rightarrow s_s \). In addition, since \( s_s \in \tilde{S}_s \), for any \( t \) enabled in \( \tilde{S} \), \( P^S_t(\tau_i^t) > 0 \). The assumption \( \tilde{S} \rightarrow \tilde{S} \rightarrow \tilde{S} \), implies that, for any \( t \) progressing in \( \tilde{S} \), \( P^S_t(\tau_i^t + 1) > 0 \) and, for any \( t \) suspended in \( \tilde{S} \), \( P^S_t(\tau_i^s) > 0 \), which is sufficient to demonstrate that \( s \rightarrow s_s \).

If \( \tilde{S}^{\phi} \rightarrow \tilde{S}_s \), we construct a state \( s = (M, \bar{\tau}) \), where the time to fire of each transition enabled by \( M \) is defined as follows: For each \( t_i \in \phi \), let \( \tau_i^t = 0 \); note that, with this choice, \( P^S_t(\tau_i^t) > 0 \) as a consequence of the assumption that \( \phi \) is an outgoing event from \( S \). For each \( t_p \) persistent at the firing of \( \phi \), let \( \tau_i^t = \tau_i^t_p \). Also in this case \( P^S_t(\tau_i^t_p) > 0 \); in fact, since \( s_s \in \tilde{S} \), and \( t_p \) is enabled in \( \tilde{S} \), \( P^S_t(\tau_i^t_p) > 0 \), which, by the assumption \( \tilde{S} \rightarrow \tilde{S}_s \), implies that \( P^S_t(\tau_i^t_p) > 0 \). For each \( t_{nc} \notin \phi \) which is progressing under \( M \) and is not persistent at the firing of \( \phi \) and is not in conflict with \( \phi \), let \( \tau_i^s \) be any value higher than 0 such that \( P^S_t(\tau_i^s) > 0 \); such value exists as a consequence of the assumption that \( \phi \) is an outgoing event from \( \tilde{S} \). For each other transition \( t_x \), let \( \tau_i^t \) be any value such that \( P^S_t(\tau_i^t) > 0 \).

By construction, \( s_s \in \tilde{S} \). In addition, according to the semantics of spTPNs, \( \phi \) is the next step in \( s \) under a feasible result of the competitive selection, and it leads to \( s_s \), under a feasible sampling of times to fire of transitions that are newly enabled at the firing of \( \phi \). This guarantees that \( \exists \mu > 0 \) such that \( s^{ev, \mu} \rightarrow s_s \).

\[ \square \]

The correspondence between the succession relation \( \xrightarrow{ev, \mu} \) in the \( STS \) and the direct reachability relation \( \xrightarrow{ev, \mu} \) in the semantics of spTPNs can be inductively extended to deal with a step sequence \( \rho \). On the one hand, if a state \( s^\ast \) can be reached from a state \( s \in \tilde{S} \) through the step sequence \( \rho \), then the \( STS \) contains a path which originates in \( \tilde{S} \), follows the same steps of \( \rho \), and leads to a stochastic state \( \tilde{S}' \) such that \( s^\ast \in \tilde{S}' \). On the other hand, if the \( STS \) contains a path \( \rho \) which originates in \( \tilde{S} \) and leads to a stochastic state \( \tilde{S}' \), then, for every state \( s^\ast \in \tilde{S}' \), there exists a state \( s \in \tilde{S} \) from which it is possible to reach \( s^\ast \) through the steps of the sequence \( \rho \).

This supports the verification of correctness requirements pertaining to ordered sequencing and time bounded response through real-time model checking techniques [34], [37], [3] or shortest and longest path algorithms [1]. In so doing, the analysis of spTPNs supports the designer with results that are similar to those that could be obtained through (dense time) formalisms accounting for preemption and finite delays, such as preemptive-TPNs [20], scheduling-TPNs [49], Timed Automata with Tasks [4].

In this perspective, it is worth noting that the \( STS \) directly represents the step semantics of spTPNs without requiring that the concurrency of the model be recovered from interleaving sequences as occurring in analysis techniques for most (dense or discrete) timed extensions of Petri Nets.

### 4.4 Performance and Dependability Evaluation

The step transition system \( STS = (S_n, S, \xrightarrow{ev, \mu}) \) can be regarded as a discrete time Markov chain \( \{S_n, n \in N\} \), where \( S_n \) is the stochastic state reached after \( n \) steps and takes values in \( S \), and transition probabilities are encoded by the succession relation \( \xrightarrow{ev, \mu} \):

\[ \text{Prob}\{S_{n+1} = S^\ast | S_n = S\} = \begin{cases} \mu & \text{if } \exists ev \text{ such that } S^{ev, \mu} \rightarrow S^\ast \\ 0 & \text{otherwise}. \end{cases} \quad (14) \]

As already mentioned in Section 4, the \( STS \) can be partitioned in stochastic states with sojourn time equal to 1 (those accepting tick as the unique outgoing event) and stochastic states with null sojourn time (those accepting firing and defer events).

The two cases closely correspond to the concepts of tangible and vanishing states commonly used in the literature of stochastic Petri Nets [45]. As a difference, while the term vanishing usually denotes a state enabling any immediate transition, in our treatment, we use it to denote a stochastic state where the dynamic time to fire of some progressing transition has reached a nonnull probability to be equal to 0. Vanishing states can be eliminated from the \( STS \) yielding a reduced step transition system \( STS = (S, S_{\text{red}}, \xrightarrow{ev, \mu}) \), where \( \rho \) denotes a sequence of steps connecting two stochastic states with sojourn time equal to 1 (by construction, any such sequence is made up of an initial tick followed by any number of firing events and at most one single defer event in the terminal position) and \( \mu \) is the product of probabilities associated with the steps along \( \rho \).

\( STS \) encodes a discrete time Markov chain \( \{S_m, m \in N\} \), where \( S_m \) is the stochastic state reached after \( m \) time units
Fig. 5. (a) a simple example spTPN: static times to fire of transitions $t_1$ and $t_4$ are assumed to be uniformly distributed; competitiveness factors are equal to 1 for all transitions. (b) The $STS$ and (c) its reduction $STS$ concealing vanishing states.

since the beginning of the execution in the initial stochastic state. This supports derivation of transient and steady state probabilities through techniques that are commonly employed on a variety of stochastic timed extension of Petri Nets based on dense [45], [32] or discrete time [15], [33], which permit the derivation of cumulative and instantaneous indexes of dependability and performance [39].

4.5 An Example

We illustrate semantics and analysis of spTPNs with reference to the net of Fig. 5a.

In the $STS$ reported in Fig. 5b, in the initial stochastic state $S_0$, only $t_5$ is enabled with $P_{S_0}^t(1) = 1$ (i.e., the time to fire is determined and equal to 1): *tick* is thus the unique outing event and leads to $S_1$, where $t_5$ becomes compelled to attempt the firing (i.e., $P_{S_1}^t(0) = 1$). In $S_2$, $t_1$, $t_2$, and $t_4$ are enabled but have a null probability to fire before time advances. Transition $t_3$ attempts the firing with probability $\frac{1}{2}$, or defers with probability $\frac{1}{2}$.

In $S_1$, $t_3$, and $t_4$ can attempt the firing or defer with equal probabilities, while $t_1$ and $t_2$ are compelled to attempt the firing. Four different attempting sets are thus possible: $\{t_1, t_2\}$, $\{t_1, t_2, t_3\}$, $\{t_1, t_2, t_4\}$, and $\{t_1, t_2, t_3, t_4\}$. The probability of each case depends on the probability mass functions of times to fire and is computed according to (8).

Due to conflicts, each attempting set may result in different firing sets, with probabilities expressed according to (9). For instance, in the most complex case in which all four transitions take part in the attempt, three firing sets are possible: $\{t_1, t_4\}$, $\{t_2, t_4\}$, and $\{t_3\}$ with probabilities equal to $\frac{3}{5}$, $\frac{2}{5}$, and $\frac{1}{5}$, respectively.

In particular, the attempting set $\{t_1, t_2, t_3, t_4\}$ yields the firing set $\{t_1, t_4\}$ if and only if $t_1$ is selected from the attempting set before $t_2$ and $t_3$ or $t_4$ is selected before $t_3$ and $t_4$ is selected before $t_2$. The probability of each case is derived according to (10). Note that, according to (9), even when competitiveness factors $C$ are equal, transitions involved in multiple conflicts receive a lower probability to be included in the firing set. In the example, this is the case of transition $t_3$, whose firing requires that $t_3$ prevails over $t_1$, $t_2$, $t_4$; conversely, $t_4$ is advantaged by the same mechanism as its firing will occur either because it prevails over $t_1$ or because $t_3$ has been overtaken by $t_1$ or $t_2$. For this reason, the probabilities of firing sets in $S_4$ are biased to include $t_4$ rather than $t_3$, though $t_1$ and $t_3$ have equal probabilities to attempt the firing and equal competitiveness factors.

Also, note that the same firing set can result from different attempting sets. For instance, in $S_u$, the firing set $\{t_1, t_4\}$ can also result from the attempting set $\{t_1, t_2, t_4\}$ in the case that $t_1$ prevails over $t_2$. This combines the effects of time probabilities (which define the choice among attempting sets) and logical concurrency of direct conflicts and competitiveness factors (which define the choice among firing sets that may result from each attempting set).

5 A Case Study

In this section, we show how spTPNs can be used to deal with both the problems of qualitative verification and quantitative evaluation, with reference to the case of a digital control system.

In digital control systems, stability and quality of control depend on the frequency and the time-invariance of control actions. In a worst-case approach, this is achieved by assuming a sampling period longer than the maximum possible delay between the acquisition of a sample and the application of its consequent control action. However, this becomes nonviable or ineffective when limited resources make computation times and scheduling delays significantly variant. In this condition, a control system which assumes a shorter sampling period and applies adequate scheduling strategies to manage variability and occasional overruns can exhibit a better performance and still maintain a correct (stable) behavior [57], [28]. This raises a design trade-off between the nominal sampling period and the probability of a jitter in the timing of control actions. While the maximum jitter is a matter of correctness determining stability, the average jitter and the nominal sampling period are a matter of performance affecting the quality of control.

Fig. 6a presents the basic spTPN model of a digital control thread. Output variables of the system are sampled at a constant rate (transition $t_{05}$: sample). For each sample, the controller calculates the control action to be applied to the system (transition $t_{06}$: compute output) and updates the state of the controller (transition $t_{07}$: update state). Both compute
output and update state are allocated to a resource (cpu) which is usually shared with other similar threads. In addition, the acquisition of the sample (transition $t_{rd}$, read) and the application of the control action (transition $t_{wr}$, write) are protected within critical sections preventing interference with other threads in the access to a card for AD and DA conversions (place $p_{mutex}$). With reference to this model, control jitter is measured as the variation in the time elapsed between subsequent firings of transition $t_{wr}$ (i.e., between the application of subsequent control actions).

In [27] and [28], two complementary scheduling strategies for the reduction of the control jitter are devised and assessed through experimental evaluation of two similar testbenches. According to [27], in each control thread the variant part of the computation represented by $t_{CO}$ can be moved into the code segment represented by $t_{US}$. The resulting timing of $t_{US}$ is well represented as a nominal value plus an additional occasional delay which is uniformly distributed [28]. To account for this special distribution, the spTPN model of each control thread is slightly refined as reported in Fig. 6b: Transitions $t_{nom}$ and $t_{del}$ account for the choice between the case of a nominal or a delayed duration of the state update; their relative probability is set through the respective competitiveness factors $c(t_{nom})$ and $c(t_{del})$; transition $t_{USn}$ is the nominal case, while the sequence of $t_{USd1}$ plus $t_{USd2}$ accounts for the delayed case.

In the testbench of [27], three inverted pendula are controlled by three concurrent threads with equal computations but different sampling periods, running on the same cpu and sharing an exclusive AD/DA converter through a semaphore. This results in a tasking set that can be modeled through three instances of the pattern of Fig. 6b composed so as to share place $p_{mutex}$. Temporal parameters and priorities of the tasking set are reported in Fig. 7a. Following the Theory of Rate Monotonic, higher priorities are assigned to threads with shorter period, and computations in the critical sections are boosted according to a Highest Priority Locking protocol. State space enumeration produces 7,251 stochastic states with 73 different markings. Fig. 7b reports the resulting minimum/maximum bounds and the distribution of probability for the jitter of control actions for each thread. Note that, due to the possible latency in the acquisition of the exclusive resource, the high priority thread also suffers of a timing variability. For both the mid and low-priority threads, variability also depends on update state computations of higher priority threads.

In order to reduce the jitter, [27] proposes that calculate output computations are given precedence with respect to update state computations. This changes the parameters of the model of the tasking set as described in Fig. 8a. In this case, the analysis produces 7,063 stochastic states with 103 different markings, which yield the results of Fig. 8b. Timing variability is largely reduced for all the threads, though the gap between the minimum and the maximum time between subsequent control actions increases for the mid priority thread.

According to [28], the quality of control can be further improved by reducing the sampling period at the expense of occasional overruns, provided that the maximum jitter is maintained within the limits required for the stability of control [29]. To this end, in [28], various overrun management policies are devised: In the queue strategy, when a sample is late, the derivation of its successor is delayed; in the abort strategy, a late sample is aborted and the derivation of its successor is started on time; in the skip strategy, late samples are completed but their successors are skipped. We
focus here on the skip strategy, which is represented by refining the model of each thread as in Fig. 6c so that the overall tasking set is modeled by the spTPN in Fig. 9 with the parameters of Fig. 10a. The priority of $t_{\text{skip}}$ depends on the marking: When $t_{\text{skip}}$ is newly enabled by the arrival of a new sample (firing of $t_S$) and if the processing of the previous sample is still pending (i.e., any further token is contained in any place along the thread), the priority of $t_{\text{skip}}$ becomes higher than that of $t_{\text{acc}}$, so that the token is consumed to represent the skip of the sample. The analysis produces 15,567 stochastic states with 127 different markings, with the results shown in Fig. 10b. While the high priority thread is not affected by the policy, for both the mid and low-priority threads, the distribution of times between the application of two subsequent control actions exhibits a significant reduction. Nevertheless, for both of them, the maximum value increases. The value of the maximum has a limited impact on quantitative performance evaluation but definitely affects the stability of control and, thus, the qualitative correctness of design.
All the analysis techniques described in the paper are implemented in the discrete time suite of the ORIS tool [21]. Running the analysis on a Pentium 4 with 512 mbytes and 1.5 gigahertz clock, enumeration is carried out in few seconds for all the cases, while stochastic analysis requires a maximum of a 10 minutes in the most complex case of skip implementation.

6 CONCLUSIONS
Stochastic preemptive Time Petri Nets (spTPN) support the representation of real-time systems running under preemptive scheduling on multiple processors, including periodic, sporadic, and asynchronous task release, with nondeterministic possibly bounded computation times, with synchronization and precedence constraints, with flexibility policies adapting behavior to dynamic conditions. They also encompass further aspects not reported in this paper including, in particular, marking-dependent temporal parameters which can represent performance polymorphism [42]. While most of this expressivity was already attained in the dense time formulation of preemptive Time Petri Nets (pTPN) [20], spTPNs assume a discrete time and they add a stochastic characterization of all nondeterministic logical and temporal choices.

The salient trait of novelty in the semantics of spTPNs consists in the use of a maximal step semantics of concurrency which enables a convenient separation of the effects of logical concurrency from those of non-Markovian transition delays. This provides a structural solution for the problems of confusion and non-well-definedness arising in interleaved models and exacerbated by the assumption of a discrete time, and relieves the designer from the burden of identifying and characterizing conflicts which do not correspond to an explicit modeling intention.

The timed state space of the model is covered using the concept of stochastic state, a state class which integrates the information about sequencing resulting from explicit synchronization constraints and implicit precedences induced by non-Markovian timed behavior is combined with a measure of probability characterizing both logical and temporal choices. This provides a novel framework unifying the qualitative verification of correctness in the logical sequencing and timing of events with the quantitative evaluation of dependability and performance indexes.

ACKNOWLEDGMENTS
This research was partially supported by the Italian Ministry of Instruction University and Research as a part of the FIRB project Performance Evaluation of Complex Systems (PERF).

REFERENCES
Giaccomo Bucci (M’80) is a graduate of the University of Bologna, Italy. From 1970 to 1982, he was with the University of Bologna. During 1975, he was a visiting researcher at IBM T.J. Watson Research Center, Yorktown Heights, New York. Since 1986, he has been a full professor at the University of Florence, Faculty of Engineering, where he teaches a course in computer architectures and a course in software engineering. Currently, he is the director of the Department of Systems and Informatics and the director of the Software Technologies Lab of University of Florence. His current research interests include software development methodologies, specification and validation techniques for time-dependent systems, and computer performance evaluation. He is a member of the IEEE and the IEEE Computer Society.

Luigi Sassoli received the doctoral degree in informatics engineering in 2002 from the University of Florence where he is now completing the PhD degree in informatics, multimedia, and telecommunications engineering. His research interests are mainly focused on state space analysis methods for performance evaluation and correctness verification of real-time reactive systems, with specific interest on the integration of stochastic characterization into discrete and dense time models. He is a member of the Software Technologies Lab of the University of Florence.

Enrico Vicario (M’95) received a doctoral degree in electronics engineering and the PhD degree in informatics and telecommunications engineering received from the University of Florence in 1990, and 1994, respectively. Since 2002 he has been a full professor in the faculty of engineering of the University of Florence. His present research activity is mainly focused on correctness verification and quantitative evaluation of dependability and performance in reactive and real-time systems. He also worked on content modeling and retrieval for image and video, visual formalisms, and multimedia applications. He is a member of the steering committee of the Center for Communication and Media Integration of the University of Florence, and a member of the Software Technologies Lab of the University of Florence. His technology transfer activity mainly focuses on software engineering with specific application in software architecture and design. He is a member of the IEEE and the IEEE Computer Society.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.