

Regenerative Simulation of Stochastic Petri Nets with Discrete and Continuous Timing

Rob Jones

ASRC Aerospace Corporation
P.O. Box 183, Williamsburg, VA 23187
robert.jones@akspace.com

Gianfranco Ciardo

College of William & Mary, Dept. of Computer Science
P.O. Box 8795, Williamsburg, VA 23187
ciardo@cs.wm.edu

Abstract

We present a regenerative simulation method applicable to a special class of non-Markovian stochastic Petri nets (SPN) that may be useful in practice for constructing and solving performability models. Introduced in [1], this special SPN is called a phased delay Petri net (PDPN) and was conceived to allow efficient numerical analysis while increasing the modeling fidelity. The PDPN permits phase-type firing delays in both discrete and continuous time simultaneously in the same model. However, exact numerical analysis requires that the discrete-time transitions be synchronized when active. We propose here, instead, a new regenerative simulation technique that offers efficient analysis of PDPNs where numerical solution by exact means or other (known) types of regenerative simulation is too difficult.

1. Introduction

As far as we know, simulation of phase-type models has not been specifically addressed before in the literature—not surprisingly, since phase-type distributions are typically introduced to allow an exact numerical solution. One might ask: Why simulate models with phase-type probability distributions when completely general distributions could be sampled instead? Well, regenerative simulation of PDPNs is an interesting idea for at least two reasons. First, simulating a PDPN is easy and efficient, as only exponential random variables and uniform random variables between zero and one are needed; the former are used to determine the time until the next phase-change event in continuous time (this time is a constant τ for discrete-time delays), the latter are used to select the next phase from the stochastic transition matrix associated with each phase-type distribution. The second reason is that, unlike general SPNs, PDPNs possess properties that aid the identification of hidden regenerative structures and easier determination of regeneration times.

2. Background

We adopt the standard SPN definition: a bipartite graph with two finite sets of nodes, places, \mathcal{P} , and transitions, \mathcal{T} , which are connected by directed arcs. Places can contain tokens and the marking $M \in \mathbb{N}^{|\mathcal{P}|}$ denotes the number of tokens in each place. Marking changes occur when an enabled transition fires, possibly after delaying a (finite, random) amount of time, thereby removing tokens from input places and depositing tokens in output places according to the arc multiplicities. We assume that the transition selected to fire is the one with the minimum remaining firing time (RFT) over all enabled transitions.

The marking process, $\{M(\theta) : \theta \geq 0\}$, observes the marking evolution over time. This gives rise to an underlying stochastic process amenable to mathematical analysis. Among the most convenient stochastic processes are the discrete-time Markov chain (DTMC), which results when firing delays are geometrically distributed, and the continuous-time Markov chain (CTMC), which results when firing delays are exponentially distributed. When the applicability of these Markovian models to more complex and realistic systems is questionable, other extensions attempt to incorporate non-Markovian behavior. However, the complexity of exact solutions to non-Markovian processes oftentimes limits the practical usefulness to problems with small dimensions unless simulation is employed.

A phase-type random variable is defined as the time to absorption of a given Markov chain with at least one absorbing state. We let `ph int` and `ph real` refer to phase-type random variables (or distributions) in discrete and continuous time, respectively. An absorbing CTMC (DTMC) is used to define a `ph real` (`ph int`) distribution, which includes initial state probabilities to be used when sampling the distribution. We let $\mathcal{T}_{\text{real}}$ denote the set of `ph real` transitions and \mathcal{T}_{int} , the set of `ph int` transitions. The set $\mathcal{T}_{\text{zero}}$ contains the immediate transitions with zero firing delays, and $\mathcal{T} = \mathcal{T}_{\text{real}} \cup \mathcal{T}_{\text{int}} \cup \mathcal{T}_{\text{zero}}$.

In general, the state of an SPN model is the current

marking augmented with clocks that record the RFT before the next change of state can occur. The underlying stochastic process is a general state space Markov chain (GSSMC), $\{X_n \equiv [M(\theta_n), C(\theta_n)] : n \geq 0\}$, constructed by observing, at successive state transition times, the marking, $M(\theta_n)$, and all necessary clock readings, $C(\theta_n)$. A GSSMC evolves in discrete time but takes values from a continuous state space due to the clocks.

A stochastic process is *regenerative* if it probabilistically restarts independently and identically at certain random times. Regenerations occur when a state transition leaves the process in a fixed probability distribution that depends only on the transition itself, not the history of the process prior to the transition. For Markov chains on a discrete state space, transitions into or out of a fixed state are trivial regeneration events. However, this property does not hold in general when the state space is continuous. Instead, regeneration times tend to rise from certain *sequences* of state transitions, as opposed to a *single* state transition, that ultimately leaves the process in a fixed distribution with only positive probability, as opposed to probability one. This notion of regeneration is exhibited by a Harris recurrent GSSMC.

Definition 2.1 [2] *A GSSMC, $\{X_n : n \geq 0\}$, with transition kernel P on the measurable space (\mathcal{S}, Σ) is Harris recurrent if there exists a function $\alpha : \mathcal{S} \rightarrow [0, 1]$, a number $e > 0$, a probability measure ν , and an integer $0 < m < \infty$ such that, $\forall X_n \in \mathcal{S}, \forall \mathcal{X} \in \Sigma$:*

- (1) $P^m(X_n, \mathcal{X}) \geq \alpha(X_n) \nu(\mathcal{X})$,
- (2) $\Pr\{\alpha(X_n) > e \text{ i.o.}\} = 1.$ (i.o. \equiv infinitely often)

The stochastic process underlying a PDPN is a GSSMC that is more amenable to analysis than most because of the way RFTs are included in the state. As it is possible to (partially) encode the distribution of the phase-type delays within the state space, we need only remember the time until the next phase change for each transition when examining the future evolution.

Execution rules governing the behavior of each transition consider two types of *model events*: one, a transition simply advances its phase when enabled, which moves it closer to its firing time and, two, a transition fires after completing its phased delay, which moves the net to a new marking. For PDPNs, the GSSMC is given by $\{X_n \equiv [M_n, D_n, C_n] : n \geq 0\}$, constructed by observing, respectively, the marking, phases, and the time remaining until the next phase change immediately after each model event.

We say that a PDPN is *synchronous* if, whenever the firing time of a `ph int` transition is resampled because of the firing of a `ph real` transition fires, any other `ph int` transition enabled in the new marking must also have its firing time resampled. If this is not the case then the PDPN

is said to be *asynchronous*. PDPNs make identifying regeneration times relatively easy, although the procedure to do so is more involved for asynchronous PDPNs than synchronous ones. For synchronous PDPNs, we need only observe visits to a fixed marking and phase, both discrete as the clock readings are synchronized with probability mass concentrated at discrete and measurable values. For asynchronous PDPNs, the increased difficulty arises because, when `ph int` and `ph real` transitions interact through conflicts and preemptions, the clock *skews* among active `ph int` transitions can be arbitrary random variables.

3. Approach

Our approach towards identifying regeneration times essentially fixes a distinguished state, s , of some discrete marking and phase and known to be visited i.o., and observes visits to a clock-enriched state, $[s, C_1, C_2, \dots]$, by the GSSMC such that each clock reading has a distinguished probability distribution that is fixed and independent of the past. We base our approach, in part, on these observations: First, only the clock readings of `ph int` transitions active in state s matter in determining whether a regeneration has taken place each time s is visited; all other (inactive) `ph int` clocks are irrelevant, as each will have some fixed default value. Second, `ph real` transition clocks can be ignored because of the memoryless property. Third and most important, `ph int` clocks, which are active in state s , can be identified as being *proper*, i.e., defined in such a way as to have distinguished probability distributions. Proper clock readings that can lead to regeneration times are identified in the following way.

Let $\text{origin}(t)$ be the time at which transition $t \in \mathcal{T}_{\text{real}} \cup \mathcal{T}_{\text{int}}$ was inserted into the event list (of the discrete-event simulator), which schedules the time, C_t , until its next phase advance and possibly its firing. Once C_t is inserted in the event list, $\text{origin}(t)$ will also give the time that C_t (hence, the firing rate of t , if $t \in \mathcal{T}_{\text{real}}$) was last updated with another time $C'_t \neq C_t$ as a consequence of some model event. Let $\text{next}(\theta)$ be the time of the next (earliest scheduled) `ph int` transition event (phase advance, possibly followed by a transition firing) immediately after the current event changes the process state at time θ . Finally, we write $t \triangleright t'$ to mean that the firing of t causes the firing time of t' to be resampled, either because t' becomes enabled or because this action is explicitly requested by the modeler.

Referring to Fig. 1, assume that $r = (r_0 r_1 r_2 \dots) \in \mathcal{T}_{\text{real}} \mathcal{T}_{\text{zero}}^*$ is a firing sequence occurring at time θ such that $r \triangleright t$ where $t \in \mathcal{T}_{\text{int}}$. Suppose also that $\theta \in [n\tau, n\tau + b]$ for some $n \in \mathbb{N}$, where $b = \text{next}(\theta) - n\tau \in [0, \tau]$ is the time scheduled for the first `ph int` event after $n\tau$. Note that τ is the fixed time between phase changes shared by all `ph int`

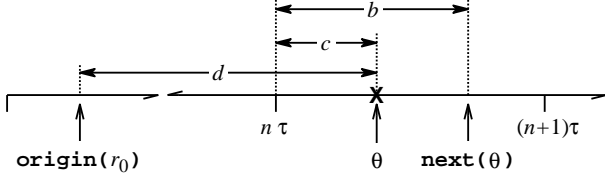


Figure 1. The timeline of a proper reset event.

delays and $0 \leq b \leq \tau$. Finally, assume that $\text{origin}(r_0) \leq n\tau$ for $r_0 \in \mathcal{T}_{\text{real}}$. The value $d = \theta - \text{origin}(r_0)$ is then the time from the last update of the firing time of r_0 to its actual firing. Let Θ , N , B , and C be the random variables associated with θ , n , b , and c , respectively. Thus, with probability

$$\beta_\lambda(b) = \frac{1 - e^{-\lambda b}}{1 - e^{-\lambda \tau}}$$

the probability density of $C = \Theta - N\tau$ is

$$f_\lambda(c) = \frac{\lambda e^{-\lambda c}}{1 - e^{-\lambda \tau}}$$

where λ is the rate of the transition firing sequence r .

The above random variable B is a conservative upper bound for C because any potential conflicts with the nearest scheduled `ph_int` transition event will prevent the firing sequence r from occurring beyond the time interval $[0, B]$. Our assumption that B varies randomly over the interval $[0, \tau]$ holds except in the special case where the clock skews are constant. But, such cases are handled more easily by simply observing the distinguished state, s , at equally-spaced intervals of length τ without the use of our “coin tosses” with probability $\beta_\lambda(b)$ when B takes on value b .

Reset events $r \triangleright t$ that satisfy the condition stated above are said to be *proper*. If `ph_int` transition clock C_t is properly reset by event $r \triangleright t$ at time $\theta = n\tau + c$ with $c \leq b$ and rate λ , then, for as long as the clock C_t is thereafter uninterrupted, the probability distribution $\Pr\{C_t \leq c\}$ at integral times $(n+k)\tau$ for $k \geq 1$ is $F_\lambda(c) = \int_0^c f_\lambda(x) dx$ with positive probability $\beta_\lambda(b)$. Based on the above reasoning, $F_\lambda(c)$ is a *fixed* exponential distribution truncated at τ when the distinguished rate, λ , is *fixed*. Therefore, the distribution function $F_\lambda(c)$ is independent of the GSSMC state, which includes the `ph_int` clock readings. Only the probability $\beta_\lambda(b)$ depends on the process state through b , and this dependency is eliminated through our coin tosses: with probability $\beta_\lambda(b)$ the clock(s) in question are proper, with probability $1 - \beta_\lambda(b)$ the clock(s) are not. The notion of uninterrupted clocks and *clock inheritance*—a `ph_int` transition can inherit the clock properties of another `ph_int` transition that enables it or causes its clock to reset—can be ex-

ploited to *preserve* the distribution $F_\lambda(x)$ of each clock after proper resets: Preserved, that is, until some future time when *all* active `ph_int` clocks are proper *and* the distinguished state $[s, C_1, C_2, \dots]$ is observed at some time $n\tau$.

4. Application

As for an application of our regenerative simulation approach, the following was inspired by [3], which proposes a novel approach for dynamically organizing network nodes into clusters. The goal of clustering is to reduce the far-reaching effects of topological changes, which requires re-routing, and allows the network to better adapt to node mobility. The essential idea is that nodes are grouped together into clusters, as shown in Fig. 2 top, with membership based on proximity, mobility, and the expectation that some lower-bound path availability will be maintained, all to ensure specified performance over a specified service-time interval. Nodes that cannot guarantee sufficient performance are not allowed to join the cluster. Nodes that are already members of a cluster but later discover that they cannot “make the grade” (due to link failures) voluntarily leave the cluster and become “orphaned.” Orphan nodes will attempt to join another cluster, or possibly the same cluster again later, after more favorable (topological, mobility) conditions resume. We assume that the network is limited to K nodes, that cluster sizes are limited to $N < K$ nodes each, and that nodes are indistinguishable in regard to clustering abilities and link capacity. There is no centralized control so nodes can asynchronously and independently join, leave, or create clusters. Each node maintains a deterministic timer that triggers periodic cluster maintenance. We assume that the probability that a node leaves a cluster has a fixed value, ω .

Fig. 2 bottom shows the SPN model that captures the node clustering behavior of the network system discussed above. The number of tokens in place $C(i)$ represents the number of clusters of size i for $2 \leq i \leq N$. Transition $Join(i)$ represents the continuous-time, asynchronous action of an orphan node joining a cluster. As the model shows, this requires that a cluster of size $i - 1$ already exists (place $C(i - 1)$ is not empty) for the orphan node to join before a cluster of size i can be created. The join operation reduces the number of clusters of size $i - 1$ and the number of orphan nodes by one each and increases by one the number of clusters of size i (a token is added to place $C(i)$ after removing one token each from places $C(1)$ and $C(i - 1)$). We assume that orphan nodes will join clusters completely at random and will join larger clusters with higher rates over smaller clusters. Therefore, we let $Join(i)$ be exponentially distributed with rate λi rate that increases linearly with the cluster size, i , which is justified by the following reasoning: A node joins a cluster at random by first coming into

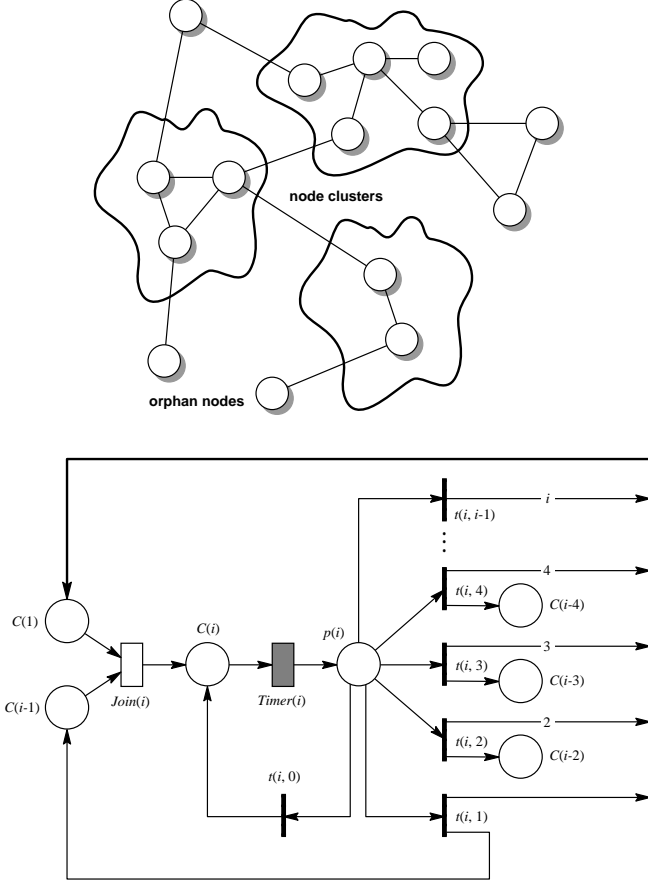


Figure 2. Clustered network and SPN model.

communication range with another node that is already a member of the cluster. The probability that a given node resides within an available cluster of size i is proportional to i/K , hence the linear rate of increase in joining clusters of increasing size.

A new, deterministic timer to trigger cluster maintenance actions is created with each new cluster of two or more nodes. Transition $Timer(i)$ with constant τ delay represents the timer for each cluster of size $i \geq 2$. It is important to note that transition $Timer(i)$ is special in that it must support *multiple enablings*: a new and distinct enabling each time a new token arrives in place $C(i)$. Upon timer expiration, a random number of nodes (fewer than i , possibly none) will exit the cluster as a result of the maintenance procedure. Such actions are modeled by firing any one of the immediate transitions, $t(i, j)$, $0 \leq j < i$. More specifically, the event that j nodes leave a cluster of size i is captured by the firing of immediate transition $t(i, j)$. If transition $t(i, 0)$ fires, meaning that no nodes leave the cluster, the token removed from place $C(i)$ is simply returned there. If transition $t(i, j)$ fires for $0 < j < i$, then j tokens must be deposited in place $C(1)$, representing the j orphaned nodes,

and the original cluster of size i is reduced to a size $i - j$. For $j = i - 1$, this has the effect of depositing i tokens into place $C(1)$ and eliminating the cluster altogether. The firing probabilities of the immediate transitions, all of which are in conflict with each other, is given by a binomial probability mass function, $\Pr\{\text{transition } t(i, j) \text{ fires} \mid \#(p(i)) = 1\} = \binom{i-1}{j} \omega^j (1 - \omega)^{i-1-j}$, with parameter $\omega \in (0, 1)$, $2 \leq i \leq N$, and $0 \leq j \leq i - 1$. We expect that an appropriate value for ω can be estimated from a detailed model of node mobility, network topology, and path availability.

The model is admittedly the simplest form of PDPN, yet it exhibits the fundamental time components, namely, the exponential and constant τ time delays. The multiple enablings of the $Timer(i)$ transitions interacting with the $Join(i)$ transitions for each i^{th} cluster coupled with the contributions by the many immediate transitions exhibits all of the (asynchronous, proper clocks, inheritance) behavior we have discussed. Although the model is not persistently asynchronous (when only one cluster exists) the simple discrete states are visited too infrequently to make good regeneration points. Our approach can instead uncover hidden regenerative structures that recur much more frequently and are associated with properly reset clocks of unsynchronized $Timer(i)$ transitions.

5. Related Work

Haas and Shedler [4] also present regenerative simulation theory in the SPN setting. Their approach is similar to ours in the sense that it, too, exploits special structures within the GSSMC. Like our approach, certain state transitions are identified that leave the process in a fixed probability distribution that depends only on the transition itself, not the history prior to the transition sequence. Rather than focus on *any* transition firing sequence that leaves the (non-memoryless) clocks properly reset before a fixed state is reached, as we do, their approach requires that a *specific* transition firing sequence be observed before a fixed state is reached where the only clocks that are active have just been reset or are exponentially distributed with the mean independent of how the fixed state was entered. Assumptions about the sample paths and firing delays having “new better than used” distributions provide sufficient conditions for Harris recurrence. Consequently, it is guaranteed that the desired state transition into the fixed state occurs i.o., which leads to regeneration. One other minor difference is that their regenerations occur in continuous time whereas ours occur in discrete time.

Our proposed regenerative simulation method was inspired by ideas that address the more customary approach toward determining regeneration times in GSSMCs. As shown in [2] a stronger minorization of the GSSMC can be used in the following way: Given the current state $X_n \in \mathcal{S}$

an m -minorization, $P^m(X_n, \mathcal{X}) \geq \varepsilon(X_n)\nu(\mathcal{X})$ for all $\mathcal{X} \in \Sigma$, some non-vanishing $\varepsilon(X_n) \in [0, 1]$, and fixed probability distribution $\nu(\mathcal{X})$ implies that the chain regenerates with positive probability each time it visits X_n and therefore $P^m(X_n, \mathcal{X}) = \varepsilon(X_n)\nu(\mathcal{X}) + (1 - \varepsilon(X_n))U(X_n, \mathcal{X})$, for some appropriately defined transition kernel U . Sample paths leaving X_n can be generated with $\nu(\mathcal{X})$ when a “coin toss” with a success probability $\varepsilon(X_n)$ indicates that a regeneration has occurred, or with $U(X_n, \mathcal{X})$ otherwise. However, U is typically unknown, so it is far easier to generate sample paths according to P , which is known, and determine regeneration events in a different way: *after* generating portions of a sample path. The trick is to first generate a path from, say, state x to state y in the usual way and, afterwards, determine whether a regeneration has occurred by tossing a *different* coin: one that has probability of success equal to $\varepsilon(x)\nu(dy)/P^m(x, dy)$. The advantage is that (ε, ν) may be defined stronger than in our approach in that ε may be larger and ν may allow the chain to “jump into” a larger space within the continuum. Unfortunately, this method requires knowledge of P^m , which is usually too hard to compute. Our approach avoids having to compute P^m by being more selective in the sample paths between states x and y that offer the possibility of regeneration.

6. Summary

Our regenerative method, applicable to asynchronous PDPNs, is both efficient and general enough to identify a broad variety of regenerative structures embedded within the underlying GSSMC. Our method simply detects the hidden regenerative structures that accompany non-memoryless clocks that may never synchronize but, nevertheless, allow for frequent regeneration times. While good simulation performance has been realized on application models, a heuristic is still needed for choosing a good regenerative structure among many that may be discovered by our approach. This is a topic for future research.

References

- [1] Rob Jones and Gianfranco Ciardo, “On phased delay stochastic Petri nets: Definition and an application,” in *Proc. 9th Int. Workshop on Petri Nets and Performance Models (PNPM’01)*, Reinhard German and Boudewijn Haverkort, Eds., Aachen, Germany, Sept. 2001, pp. 165–174, IEEE Comp. Soc. Press.
- [2] Shane G. Henderson and Peter W. Glynn, “Regenerative Steady-State Simulation of Discrete-Event Systems,” *ACM TOMACS*, vol. 11, no. 4, Oct. 2001.
- [3] A. Bruce McDonald and Taieb Znati, “A Mobility Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks,” *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad-Hoc Networks*, vol. 17, no. 8, pp. 1466–1487, 1999.
- [4] Peter J. Haas and Gerald S. Shedler, “Regenerative stochastic Petri nets,” *Performance Evaluation*, vol. 6, pp. 189–204, 1986.