Cost-Effective Software Rejuvenation Combining Time-Based and Inspection-Based Policies

Laura Carnevali *Member, IEEE*, Marco Paolieri *Member, IEEE*, Riccardo Reali,
Leonardo Scommegna *Member, IEEE*, and Enrico Vicario, *Member, IEEE*

Abstract—Software rejuvenation is a proactive maintenance technique that counteracts software aging by restarting a system, making selection of rejuvenation times critical to improve reliability without incurring excessive downtime costs. Various stochastic models of Software Aging and Rejuvenation (SAR) have been developed, mostly having an underlying stochastic process in the class of Continuous Time Markov Chains (CTMCs), Semi-Markov Processes (SMPs), and Markov Regenerative Processes (MRGPs) under the enabling restriction, requiring that at most one general (GEN), i.e., non-Exponential, timer be enabled in each state. We present a SAR model with an underlying MRGP under the bounded regeneration restriction, allowing for multiple GEN timers to be concurrently enabled in each state. This expressivity gain not only supports more accurate fitting of duration distributions from observed statistics, but also enables the definition of mixed rejuvenation strategies combining time-based and inspection-based policies, where the time to the next inspection or rejuvenation depends on the outcomes of diagnostic tests. Experimental results show that replacing GEN timers with Exponential timers with the same mean (to satisfy the enabling restriction) yields inaccurate rejuvenation policies, and that mixed rejuvenation outperforms time-based rejuvenation in maximizing reliability, though at the cost of an acceptable decrease in availability.

Index Terms—Software Aging and Rejuvenation (SAR), time-based software rejuvenation, inspection-based software rejuvenation, Markov regenerative process, bounded regeneration restriction, stochastic state classes, stochastic time Petri net.

1 Introduction

1.1 Motivation and challenges

The well-known phenomenon of *software aging* is empirically observed to increase the failure rate and degrade the performance of long-running software systems [17]. It is due to the activation of *Aging-Related Bugs* (ARBs) [15], i.e., software faults that manifest their effects over time, affecting a variety of software systems like cloud infrastructures [31], operating systems, database management systems and middleware [12], Software-Defined Networking (SDN) [38], and cyber-physical systems [23]. As time goes by, ARBs can be activated and propagated [2], leading the system into error states with increasing failure probability, until an aging-related failure occurs, e.g., a process crash due to a memory allocation failure, caused by a memory leak that progressively reduced the available resources.

ARBs are a subclass of Mandelbugs [15], i.e., bugs that are difficult to isolate and systematically reproduce due to complex activation and error propagation, often depending on interactions with execution environments and third-party integrated libraries [40]. Therefore, ARBs are hard to detect and remove before deployment and operation, due to the limited increase of detection probability with respect to testing efforts [18]. To avoid or mitigate disruptive failures, unnecessary resource consumption, and other aging effects, a viable alternative to fault removal is *software rejuvenation*, a preventive and proactive maintenance technique that, in its basic form, consists in repeatedly stopping the system, cleaning its internal state, and restarting it [22]. This approach improves the system reliability (i.e., the probability of operation without failures) by mitigating error accumulation and propagation, while reducing the system availability (i.e., the probability of being available for service) due to downtimes. Thus, the selection of rejuvenation times subtends a crucial trade-off between software qualities of reliability and performance efficiency [28].

1.2 Related works

In the literature on Software Aging and Rejuvenation (SAR), various stochastic models have been used to compute optimal rejuvenation times [13] for different categories of software systems and rejuvenation policies, thus yielding different classes of underlying stochastic process. The concept is illustrated by the Venn diagram shown in Fig. 1.

In the class of SAR models with an underlying Continuous Time Markov Chain (CTMC), a two-step failure process is taken into account in the seminal work [22], computing the expected system downtime and its steady-state availability to derive the conditions under which rejuvenation is useful. In [21], a condition-based maintenance model is presented for systems subject to both deterioration failures (i.e., failure due to gradual worsening of system parameters) and Poisson failures (i.e., abrupt failures), performing minimal or major restoration (i.e., reduction of accumulated deterioration of an unfailed system) based on inspections results, and minimal or major repair (i.e., recovery of a failed system) depending on whether a Poisson or a deterioration

L. Carnevali, R. Reali, L. Scommegna, and E. Vicario are with the University of Florence, Department of Information Engineering, Via di Santa Marta 3, 50139 Firenze, Italy. E-mail: {laura.carnevali, riccardo.reali, leonardo.scommegna, enrico.vicario}@unifi.it
M. Paolieri is with the University of Southern California, Department of Computer Science, 941 Bloom Walk, Los Angeles, CA 90089, USA. E-mail: paolieri@usc.edu
Manuscript received Month Day, Year; revised Month Day, Year.

failure has occurred, respectively. The model is specified through Generalized Stochastic Petri Nets (GSPNs) and numerically solved to compute an optimal inspection policy and an optimal mean time between inspections with respect to maximizing the system throughput. Closed-form formulas of the steady-state availability and the mean time to failure for this model are derived in [9]. The impact of different rejuvenation policies on the availability of cluster systems is studied in [8] by analyzing models specified by Stochastic Reward Nets (SRNs), using a hypo-Exponential distribution for the time-to-failure and approximating a deterministic rejuvenation period by an Erlang distribution to guarantee that the underlying stochastic process is a CTMC. A hypo-Exponential time-to-failure distribution is used also in [43], deriving the optimal rejuvenation rate that maximizes an analytical expression of the system availability.

Expressivity is improved by models with an underlying Semi Markov Process (SMP) [25], making the system evolution dependent not only on the current logical state, as in a CTMC, but also on the sojourn time in the state, which can be characterized by a general (GEN, i.e., non-Exponential) distribution. Typically, these models are defined by directly identifying the SMP states, and they are analyzed to derive the closed-form expression of the system availability, which is maximized to compute the optimal rejuvenation schedules, e.g., as performed in [14] for an extension of the model of [22]. The SMP of a virtualized server is used in [29] to derive the optimal rejuvenation schedule, minimizing both the server steady-state availability and the completion time of a job running on it. In [4], a two-level hierarchical model is presented where a low-level CTMC degradation model provides failure rate analysis to a high-level SMP proactive fault management model, which, in turn, derives the optimal rejuvenation schedule. A similar solution toolchain is defined in [45] to implement software rejuvenation for systems subject to memory leaks. A condition-based maintenance problem for a system subject to both deterioration and Poisson failures is formulated in [10] in terms of a Semi-Markov Decision Process (SMDP), performing joint optimization of the inspection rate and the maintenance policy, and assuming that all general variables take a deterministic value (in particular, duration of maintenance and repair).

Expressivity is further improved by models with an underlying Markov Regenerative Process (MRGP, often abbreviated as MRP) [25], not requiring that the Markov condition is satisfied at each state transition, as in an SMP, but only requiring that it is eventually satisfied with probability 1 at specific times termed *regeneration points*. The expressive power of MRGPs is crucial to model aging phases between consecutive rejuvenations, although, in almost all papers, it is strictly limited by the *enabling restriction* [11], requiring that at most one GEN timer be enabled in each state. In the seminal work [16], time-based rejuvenation is modeled by a Markov Regenerative Stochastic Petri Net (MRSPN) using a deterministic (DET) transition to represent the rejuvenation period and Exponential (EXP) transitions for the remaining durations (see Fig. 1). SAR models in this class are used to represent various software systems with different aims, e.g., to schedule rejuvenation of virtual machines and virtual machine monitors in server virtualized systems [27], to compare time-based and prediction-based rejuvenation in cluster systems [35], and, to improve performability of clustered systems with varying workloads [39].

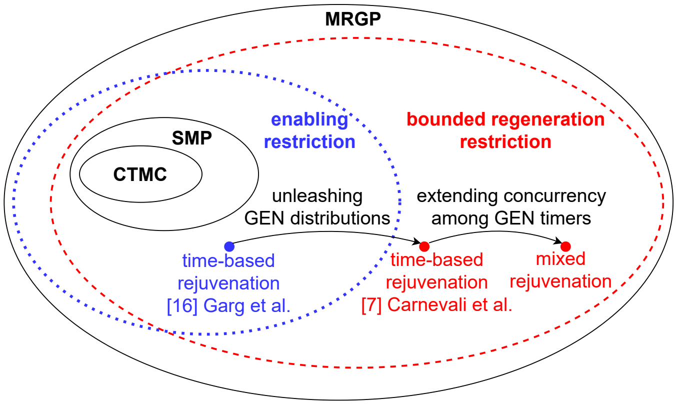

MRGP
SMP
CTMC
enabling restriction
bounded regeneration restriction
unleashing GEN distributions
extending concurrency among GEN timers
time-based rejuvenation [16] Garg et al.
time-based rejuvenation [7] Carnevali et al.
mixed rejuvenation

Fig. 1. SAR models: classes of underlying stochastic process.

Recently, time-based rejuvenation is specified in [7] by a variant of the model of [16] having an underlying MRGP under the *bounded regeneration restriction* [5]. This condition is satisfied if a regeneration is always reached in a bounded number of discrete events, and if GEN timers have DET values or exponomial (often termed expolynomial) distributions, thus breaking the enabling restriction by allowing for concurrent GEN timers. Preliminary experiments show that representing GEN timers impacts the evaluation of the optimal rejuvenation period. An attempt of integrating time-based and inspection-based rejuvenation is also performed in [7], though considering only up to two inspections and without optimizing the times at which they are performed.

1.3 Contribution

We define a novel *mixed* rejuvenation policy that combines the time-based and inspection-based approaches [1]. Specifically, timers are used to trigger inspections (i.e., diagnostic test) of the system state; then, the inspection results are used to decide whether rejuvenation is triggered immediately (as in inspection-based approaches) or postponed; finally, after at most a fixed number of inspections, rejuvenation is performed anyway, regardless of the system state (as in time-based approaches). We define a SAR model with an underlying MRGP under the bounded regeneration restriction, exploiting prior knowledge about the system behavior, such as insights on the aging steps before failure, as well as information derived from measurements or historical data, such as statistics on rejuvenation times, and specificity (i.e., percentage of correctly predicted non-aged states) and sensitivity (i.e., percentage of correctly predicted aged states) of diagnostic tests. Then, we define an efficient quantitative approach to optimize the times at which inspections and rejuvenations are scheduled. For each sequence of observed inspection outcomes, we derive the time to the next inspection or rejuvenation as the maximum time for which the transient unreliability per time unit, given the inspection outcomes since the last rejuvenation, is not larger than a given threshold. In turn, this probability can be computed by performing semi-symbolic transient analysis of the SAR model by the method of stochastic state classes [19].

The proposed rejuvenation policy can be applied to any software system where frequent inspections are costly and, consequently, optimization of inspection times is needed to achieve a tradeoff between reliability and availability, as occurring in SDN networks due to severe resource constraints. In this reference scenario, we consider an SDN-controller subject to software aging, which acts as a monitored slave sending aging-related indicators to a monitoring master, e.g., another SDN-controller, responsible of implementing the rejuvenation policy. Experiments are performed to compute reliability and availability measures by varying the number of inspections, and the sensitivity and specificity of diagnostic tests, considering both constant and time-varying values. The obtained experimental results show that mixed rejuvenation outperforms pure time-based rejuvenation in maximizing reliability, though at the cost of an acceptable decrease in availability. By referring to our SDN scenario, we also show that approximation of GEN timers with Exponential timers having the same expected value (to comply with the enabling restriction) yields inaccurate rejuvenation.

An artifact supporting replication of the experimental results presented in this paper is available open-source under the AGPLv3 license at http://github.com/oris-tool/sar. The artifact uses the SIRIO Java library[1] of the ORIS tool[2] for specification and analysis of the considered SAR models and was used to obtain preliminary results in [7].

In the rest of the paper, first, we illustrate the motivating scenario and we derive its stochastic parameters (Section 2). Then, we illustrate the SAR models of time-based rejuvenation under and beyond the enabling restriction, respectively, comparing the optimal rejuvenation period obtained in the two cases and the corresponding availability and reliability measures (Section 3). Next, we present the SAR model of mixed rejuvenation, combining time- and inspection-based rejuvenation, comparing the achieved experimental results with those obtained through the model with pure time-based rejuvenation (Section 4). Finally, we draw our conclusions and we discuss future research directions (Section 5). A summary of the analysis methods used to compute the dependability metrics is provided in the Appendix.

2 Emerging computing scenario

In this section, we define an emerging computing scenario in the SDN context (Section 2.1) and we illustrate how its stochastic parameters can be derived (Section 2.2).

2.1 System model

In the ever-evolving landscape of networked systems and network softwarization, software aging has become a critical issue. As networks have embraced paradigms like Software-Defined Networking (SDN) [38], Network Function Virtualization (NFV) [33], Multi-access Edge Computing (MEC) [3], and Internet of Things (IoT) [26], the need to detect and mitigate aging effects has become increasingly important.

Measurement-based software rejuvenation usually implements continuous monitoring of many aging indicators, i.e., system parameters that might indicate its aging state. However, in a softwarized network, continuous monitoring would require continuous extraction and transmission of data over the network, increasing bandwidth usage and resource consumption. Since the number of network connections is likely to grow[3] [24], continuous monitoring of softwarized network elements might soon be unfeasible. Thus, reducing the monitoring overhead, in particular the polling rate, is a challenge faced nowadays in various softwarized network contexts like SDN [42] and IoT networks [34].

We consider SDN as exemplary use case scenario. SDN is a network concept that enables centralized and intelligent management and control of individual hardware components through software. As shown in Fig. 2, the network structure consists of three distinct planes, i.e., the *application*, *control*, and *data* planes, which are interconnected by two interfaces, i.e., the northbound interface, between the application and control planes, and the southbound interface, between the control and data planes. Specifically:

- The application plane provides a platform that enables network operators to configure the desired network control logic via the northbound interface.
- The control plane is the core of the SDN network. On the one hand, it simplifies the process of network decision-making in the application plane by abstracting the network state and resource information. On the other hand, the control plane converts the control logic into basic flow instructions and implements them into the data plane through the southbound interface. The control plane is also responsible for promptly adapting to network topology alterations due to data plane failures, ensuring its consistency.
- The data plane is only responsible for directing packets based on the implemented flow instructions.

In particular, the control plane consists of one or more SDN-controllers that interact with the SDN-switches of the data plane, providing instructions and managing the traffic flow based on the network policies and conditions specified

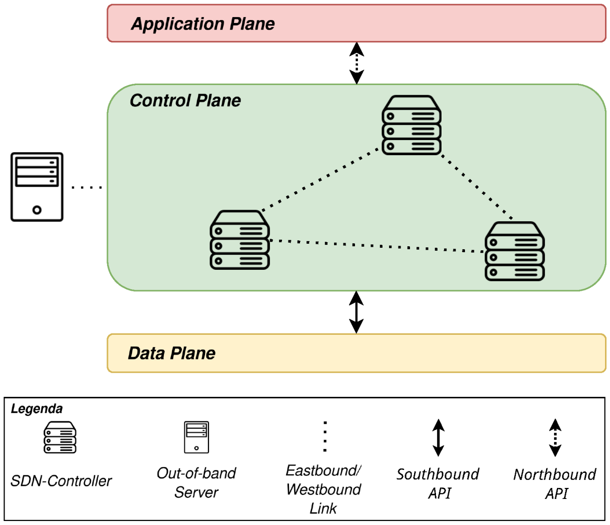


Fig. 2. Graphical representation of the considered system model.

[1] https://github.com/oris-tool/sirio
[2] https://www.oris-tool.org/
[3] https://www.statista.com/outlook/tmo/internet-of-things/worldwide#investment, accessed on March 10, 2024

TABLE 1
Stochastic parameters of the system model depicted in Fig. 2.

Duration x	**Duration Statistics**	**Duration PDF**
Time to error	$\mu = 240\,\mathrm{min} \ \wedge \ \sigma = 180\,\mathrm{min}$	$f(x) = \textsc{HypoEXP}(0.00615, 0.01289)$
Time from error to failure	$\mu = 840\,\mathrm{min} \ \wedge \ \sigma = 600\,\mathrm{min}$	$f(x) = \textsc{HypoEXP}(0.00208, 0.00277)$
Time to failure detection	$P\{x \in [0,4]\ \mathrm{min}\} = 1$	$f(x) = \textsc{Uni}(0,4)$
Repair time (from failure state)	$P\{x \in [2,16]\ \mathrm{min}\} = 1 \wedge P\{x \in [2,8]\ \mathrm{min}\} = 0.7$	$f(x) = 0.0155193(x-2)(16-x)e^{-0.2505x}$
Rejuvenation time (from working state)	$P\{x \in [1,2]\ \mathrm{min}\} = 1 \wedge P\{x \in [1,1.5]\ \mathrm{min}\} = 0.6$	$f(x) = 29.4711(x-1)(2-x)e^{-1.0805x}$
Rejuvenation time (from error state)	$P\{x \in [1,4]\ \mathrm{min}\} = 1 \wedge P\{x \in [1,2]\ \mathrm{min}\} = 0.7$	$f(x) = 11.4951(x-1)(4-x)e^{-1.8619x}$

by the application plane. Given the crucial role played by SDN-controllers, a crash of even one controller could yield unforeseen behaviors, e.g., impossibility for applications to interact over the network, packet losses, degraded performance and wrong results due to the lack of optimized paths within the data plane [37]. Therefore, it becomes imperative to improve the *reliability* of SDN-controllers (i.e., probability that the controller will continuously perform its intended function during a specified time interval $[0, T]$), in order to reduce their failure risk, while not significantly reducing their *cumulative availability* (i.e., expected time during which the controller is working during a specified time interval $[0, T]$), in order to limit their downtime [32], [44]. Meeting this goal requires achieving a trade-off between reliability and availability, as reliability is typically increased by preventive maintenance policies that reduce the availability.

In this study, we tackle the problem of software aging in an SDN-controller, an issue already demonstrated and addressed in the literature [37], [38]. The aging indicators of the SDN-controller are sent across the network to another element that analyzes them to determine the aging status. As in [44], data are transmitted via the eastbound/westbound interfaces of the SDN-controller, either to another SDN-controller or to an "out-of-band" server. Due to the costs of continuous monitoring, aging tests are performed by a threshold-based approach, relying on observed instantaneous values of the aging indicator rather than on collections of densely acquired measurements (as usual in statistical and machine-learning approaches). Thus, the aging detection test gives a positive result if a predetermined threshold is exceeded by the aging indicator. An aging detection test is characterized by specificity and sensitivity, possibly varying, typically increasing, over time, given that aging effects become more evident over time and thus easier to detect [30]. Specificity and sensitivity remain lower than 1, which is realistic given that an aging indicator may not always accurately reflect the system aging conditions due to transient system loads [15]. Therefore, repeating the test helps reducing the number of false negatives and positives. We thus aim at minimizing the monitoring overhead while providing an optimal trade-off between availability and reliability of the monitored SDN controller, by scheduling a limited number of aging tests and exploiting their outcomes to determine the time interval before executing the next test or activating the rejuvenation process.

2.2 Stochastic parameters

For the SDN-controller, we consider a two-step failure process, as usual in the literature on components subject to software aging [16]. According to this, the SDN-controller is in a safe state at the initial time, and then, due to software aging, it may reach an error state which, in turn, eventually leads to a failure state. In contrast to [16], we distinguish between detected and undetected failures, considering that, when the SDN-controller has failed, first the failure is detected and then the controller is repaired, finally returning to the safe state. In particular, failure detection is performed through periodic checks of the controller state. When it is working or aged, the SDN-controller can be rejuvenated, according to different policies (discussed in Sections 3 and 4).

Table 1 shows realistic statistics for each duration considered in the SAR process, defined so that stochastic parameters are in the same order of magnitude as those reported in [38]. According to the available statistics, durations are fitted by different Probability Density Functions (PDFs) that have already been shown to be suitable for characterizing temporal parameters of software aging phenomena as well as of rejuvenation and repair operations [6]. Specifically:

- If the expected value μ and the coefficient of variation cv are known, then the approximants of [41] can be used. Specifically, if $1/\sqrt{2} < \mathrm{cv} < 1$ is satisfied, as actually occurring in our experiments, then the approximant turns out to be a hypo-Exponential (hypo-EXP) PDF $f(x) = (\lambda_1 \lambda_2 / (\lambda_1 - \lambda_2))(e^{-\lambda_2 x} - e^{-\lambda_1 x})$ where $\lambda_i = (2/\mu)(1 \pm \sqrt{2\mathrm{cv}^2 - 1})^{-1}$ with $i \in \{1, 2\}$, e.g., if the system is working, the duration of software aging has $\mu = 240\,\mathrm{min}$ and standard deviation $\sigma = 180\,\mathrm{min}$ (i.e., $\mathrm{cv} = 0.5$), and it is fitted by a hypo-EXP PDF with rates $\lambda_1 = 0.0021$ and $\lambda_2 = 0.0027$.
- If the duration represents the remaining time to the occurrence of a periodic event having period p, then it is characterized by a uniform PDF over $[0, p]$, this being a special case of the remaining life of a renewal process [25], e.g., failure detection is performed every $4\,\mathrm{min}$, and thus the remaining time to failure detection is associated with a uniform PDF over $[0, 4]\,\mathrm{min}$.
- If the duration has bounded support $[a, b]$ and is lower than $c \in (a, b)$ with probability p, it is modeled by the exponomial PDF $f(x) = \alpha(x-a) \cdot (b-x)e^{-\lambda x}$ with $\alpha, \lambda \in \mathbb{R}_0^+$, $\int_a^b f(x)dx = 1$, and $\int_a^c f(x)dx = p$, preserving integral p over $[a, c]$ and finite support with null values at the extremes, e.g., the repair time of the SDN controller is between $2\,\mathrm{min}$ and $16\,\mathrm{min}$, and lower than $8\,\mathrm{min}$ with probability 0.7, thus having PDF $f(x) = 0.1551\,(x-2)\,(16-x)\,e^{-0.2505x}$ over the time interval $[2, 16]\,\mathrm{min}$.

Quantitative evaluation of the SAR models considered in this paper is performed by regenerative analysis based on the method of stochastic state classes [19], [20] (see Sections 3 and 4). Given that the implementation of this method provided by the SIRIO library accepts durations that are associated with any non-Markovian distribution in the class of exponomial functions, the proposed approach can be easily tailored to any other statistics and approximant.

3 Time-based software rejuvenation

In this section, we illustrate the time-based software rejuvenation policy by referring to our SDN scenario (Section 3.1), and we model the policy through a variant of the SAR model of [16] beyond the enabling restriction (Section 3.2). Then, we analyze the model to derive the rejuvenation period that optimizes a trade-off between dependability-related measures (Section 3.3), showing that using a model under the enabling restriction yields an inaccurate rejuvenation policy, and discussing the impact on the SDN scenario (Section 3.4).

3.1 Time-based software rejuvenation policy

In the time-based policy, software rejuvenation is performed periodically, using a pre-determined period, and it brings the system back to its initial safe state. If rejuvenation is triggered when the system is failed but the failure has not yet been detected, a system repair is started immediately.

In the considered SDN scenario, once the rejuvenation period is determined, the time-based policy could be enforced without performing monitoring actions, potentially eliminating monitoring messages from the control plane. However, completely removing SDN-controller monitoring could lead to longer detection times in case of controller failures. Therefore, it is recommended to implement at least a keepalived mechanism to guarantee a sufficient level of availability of the SDN-controller [44], as in fact considered in our scenario, illustrated in Section 2.2.

3.2 SAR model under bounded regeneration restriction

Fig. 3 shows the models of time-based rejuvenation specified using Stochastic Time Petri Nets (STPNs) [19]. STPNs model concurrent timed systems with stochastic temporal parameters and discrete probabilistic choices. An STPN consists of places, transitions, and arcs: places (in Fig. 3, depicted as circles) containing tokens (depicted as dots inside circles) model the discrete logical state of the system; transitions (depicted as bars) model activities with stochastic duration; directed arcs (depicted as directed arrows) from input places to transitions and from transitions to output places model precedence relations among activities; and inhibitor arcs (depicted as dotted edges) from inhibitor places to transitions represent inhibitor conditions for the execution of activities. A transition is enabled by a marking (i.e., an assignment of tokens to places) if each of its input places contains at least one token, none of its inhibitor places contain any token, and its enabling function (indicated by label "e", not present in the models of Fig. 3) evaluates to true. Upon enabling, each transition samples a time-to-fire from its Cumulative Distribution Function (CDF), i.e., an EXP distribution (EXP transitions are depicted as tick white bars), a GEN distribution (GEN transitions are depicted as tick black bars), or the generalized distribution of a Dirac Delta function[4] centered either at zero (transitions with zero time-to-fire, not present in the models of Fig. 3, are termed immediate (IMM) and depicted as thin black bars) or at a non-zero DET value (transitions with non-zero DET value are depicted as thick gray bars). The transition with minimum time-to-fire is selected to fire, removing one token from each of its input places, adding one token to each of its output places, and applying its update function (indicated by label "u"), i.e., an assignment of tokens to each place, defined by a marking expression. Ties (i.e., limit cases of synchronization among DET transitions with the same time-to-fire, e.g., occurring when transitions with the same DET value are enabled at the initial time) are solved by a random switch determined by probabilistic weights of transitions (indicated by label "w", not present in the models of Fig. 3).

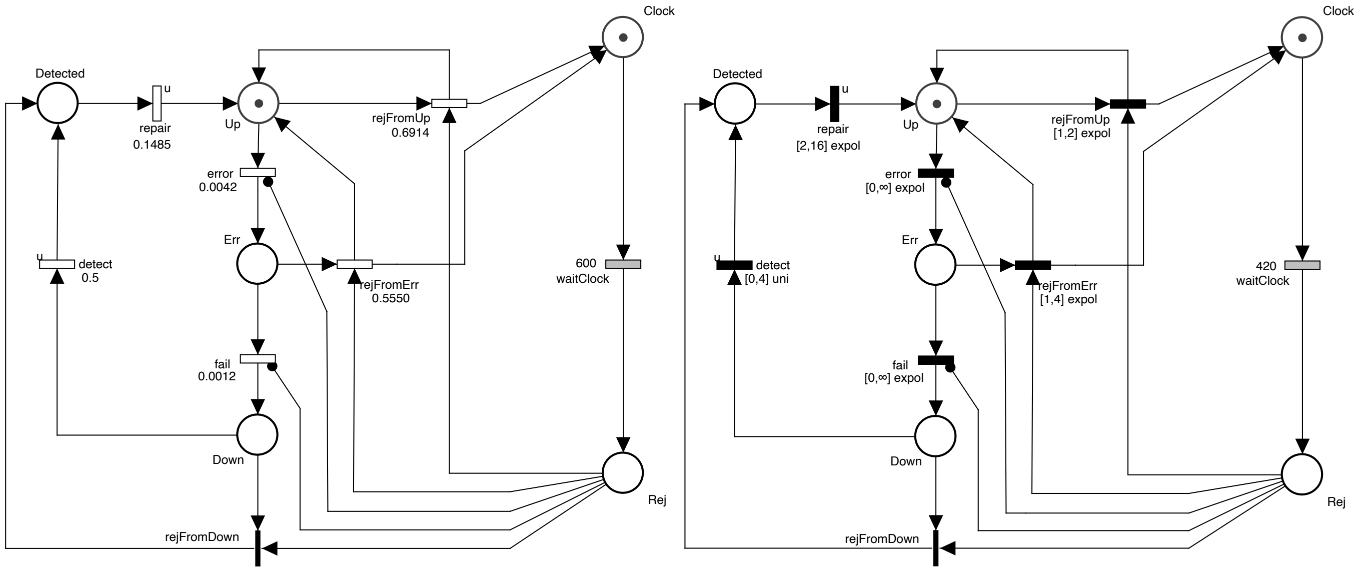
The STPN of Fig. 3a models the two-step failure process illustrated in Section 2.2, which consists of an initial safe state (modeled by place `Up` in Fig. 3a), an error state (modeled by place `Err`), and a failure state (modeled by place `Down`). Concerning failure detection, the model includes a state where a failure has occurred but has not been detected (modeled by place `Down`), and a state where a failure has been detected and the system is repaired (modeled by place `Detected`). Concurrently, the rejuvenation process starts in an initial state (modeled by place `Clock`) waiting until the rejuvenation period has elapsed (modeled by place `Rej`): at this time, error or failure events are inhibited, and rejuvenation is performed in safe or error states (modeled by transitions `rejFromUp` and `rejFromErr`, respectively). If the rejuvenation clock expires when a failure has occurred but has not been detected, then a repair is started immediately (modeled by transition `rejFromDown`). After rejuvenation, the system goes back to the safe state and the rejuvenation process restarts. According to this, transition `detect` has an update function "`Clock=0`" in order to stop the rejuvenation process during system repair, and transition `repair` has an update function "`Clock=1`" in order to restart the rejuvenation process after repair.

In Fig. 3a, the rejuvenation period is modeled by a DET transition while the remaining durations are modeled by EXP transitions fitting the expected values of the PDFs reported in Table 1, so that the underlying stochastic process is an MRGP complying with the enabling restriction. Conversely, the model of Fig. 3b extends the model of Fig. 3a by replacing EXP transitions with GEN transitions having the PDFs of Table 1, so that the underlying stochastic process is an MRGP under the bounded regeneration restriction, going beyond the enabling restriction by supporting the representation of multiple concurrent GEN timers.

3.3 Optimal rejuvenation period

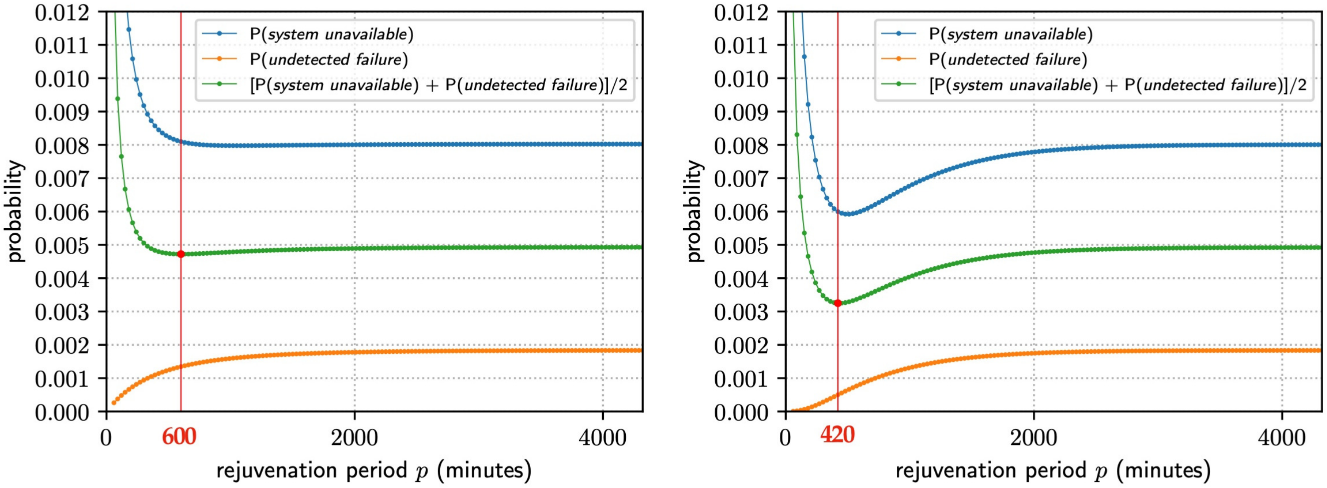
To set a suitable rejuvenation period p, we select the value p^* that optimizes the trade-off between the steady-state unavailability $\bar{a}(p)$, a quantitative measure of the time during which the system is not able to provide service, and the steady-state probability of undetected failure $\bar{r}(p)$, a

[4] Without loss of generality, we do not provide a formal definition of generalized CDF and generalized PDF of a discrete random variable.



(a) (b)

Fig. 3. STPN models of time-based rejuvenation: (a) an extension of the model of [16] to distinguish between detected and undetected failures, having an underlying MRGP under the enabling restriction; (b) an extension of the model of Fig. 3a with multiple concurrent GEN timers, having an underlying MRGP under the bounded regeneration restriction, i.e., beyond the enabling restriction. Temporal parameters are expressed in min.



(a) (b)

Fig. 4. Steady-state unavailability, undetected failure probability, and their average as a function of the rejuvenation period for: (a) the model of Fig. 3a (with underlying MRGP under the enabling restriction) and (b) the model of Fig. 3b (with underlying MRGP under the bounded regeneration restriction). The rejuvenation period that is optimal at minimizing the considered measures is indicated in red.

quantitative measure of interest to the system designers. Specifically, we minimize the average of the two measures:

$$p^* = \arg\min_{p} \left(\frac{\bar{a}(p) + \bar{r}(p)}{2} \right) . \qquad (1)$$

Note that both unavailability $\bar{a}(p)$ and unreliability $\bar{r}(p)$ increase with the probability of system failures; when rejuvenation is more frequent, $\bar{a}(p)$ increases due to the additional periods of unavailability, but system failures are less frequent, reducing $\bar{a}(p)$ and $\bar{r}(p)$.

For both models in Fig. 3, $\bar{a}(p)$ and $\bar{r}(p)$ can be evaluated by performing regenerative steady-state analysis based on the method of stochastic state classes [20], computing the rewards "`If(Down+Detected>0||Rej>0,1,0)`" and "`Down`", respectively (evaluation of the model of Fig. 3a, satisfying the enabling restriction, could also be performed in closed form by inverting the matrix form of the generalized Markov renewal equations, with global and local kernels derived in the Laplace-Stieltjes domain). In particular, we use the SIRIO library to evaluate $(\bar{a}(p) + \bar{r}(p))/2$ for 143 variants of each model in Fig. 3b, obtained by varying the rejuvenation period from 60 min to 4320 min (corresponding to 3 days), with increments of 30 min.

The analysis results are shown in Fig. 4. For the model of Fig. 3a (enabling restriction), as shown in Fig. 4a, both measures have a monotonic trend, which decreases for the steady-state unavailability $\bar{a}(p)$, reaching the value 0.008022, and increases for the steady-state probability of undetected failure $\bar{r}(p)$, reaching the value 0.001833. The rejuvenation period minimizing the average of these measures in Eq. (1) is $600\,\mathrm{min}$, achieving the objective 0.004722. Conversely, for the model of Fig. 3b (bounded regeneration restriction), $\bar{r}(p)$ increases with the rejuvenation period p, converging to 0.001832. At the same time, $\bar{a}(p)$ initially decreases with respect to the rejuvenation period p, reaching the minimum value 0.005921 for $p = 510\,\mathrm{min}$, and then increases, converging to the value 0.008006. The rejuvenation period minimizing the average of $\bar{a}(p)$ and $\bar{r}(p)$ in Eq. (1) is $420\,\mathrm{min}$, achieving the objective 0.003250. Note that, for the optimal period $600\,\mathrm{min}$ obtained through the model of Fig. 3a, the model of Fig. 3b achieves $\bar{a}(600\,\mathrm{min}) = 0.008100$ and $\bar{r}(600\,\mathrm{min}) = 0.001346$, with average equal to 0.004722, pointing out the inaccuracy of rejuvenation policies designed using the model under the enabling restriction.

Note that, in Eq. (1), we select the rejuvenation period p that minimizes a scalar metric, i.e., the average of the steady-state unavailability $\bar{a}(p)$ and the steady-state undetected failure probability $\bar{r}(p)$, thus assigning equal importance to these two metrics. Other functions are supported as well by the approach, e.g., weighted average, geometric mean. Also note that the probability that the system is in a failure state (modeled by place `Down`) contributes to both unavailability (the system is not available for service) and unreliability (the system is down and not functioning). In contrast, the probability that the system is in a rejuvenation (modeled by place `Rej`) contributes only to unavailability. As a subtle aspect observed in Eq. (1), by making the probability that the system is in a rejuvenation higher, the probability that the system is not available becomes lower, yielding the already mentioned tradeoff between unavailability and unreliability.

In contrast to the optimization problem of Eq. (1), a multi-objective optimization problem could be formulated, which would result in a Pareto front of candidate times for the rejuvenation period, with different tradeoffs between unreliability and unavailability, as improving one of these metrics worsens the other one. In the mixed rejuvenation policy (presented in Section 4), where multiple parameters need to be optimized, i.e., the times to the next inspection or rejuvenation, multi-objective optimization would substantially increase the optimization times. Moreover, due to the tradeoff between unreliability and unavailability, separate optimization of these two metrics, one after the other, would not work as well. Therefore, to allow the comparison between the two rejuvenation approaches, we solve the optimization problem with scalar objective of Eq. (1).

3.4 Dependability measures of optimal solutions

3.4.1 Unreliability and cumulative unavailability

We investigate how approximating GEN timers with EXP timers that fit their expected value impacts on the evaluated transient behavior of the system, by considering the cumulative transient unavailability and the transient unreliability. To this end, for both models of Fig. 3 with the optimal rejuvenation periods derived in Section 3.3, we use the SIRIO library to perform regenerative transient analysis based on the method of stochastic state classes [19], using time limit $4320\,\mathrm{min}$ (corresponding to 3 days) and time step $0.1\,\mathrm{min}$. We evaluate the transient unreliability by computing the reward "`Down`" with stop condition "`Down==1`" for the analysis (i.e., the system has encountered at least one failure). Moreover, we derive the cumulative transient unavailability at time t as the integral of the instantaneous unavailability in $[0, t]$, which, in turn is obtained by computing the reward "`If(Down+Detected>0||Rej>0,1,0)`".

Results are depicted in Fig. 5. As illustrated in Figs. 5a and 5b, the transient unreliability increases with time, and, at multiples of the rejuvenation period, the increase is reduced, confirming the importance of rejuvenation. Similarly, as shown in Figs. 5c and 5d, the increase in cumulative transient unavailability near multiples of the rejuvenation period is reduced as time advances, due to the fact that the corresponding sharp peaks in the transient unavailability are distributed over longer periods of time with lower maximum probability values, as an effect of the randomness of repair times (after which the rejuvenation clock is restarted).

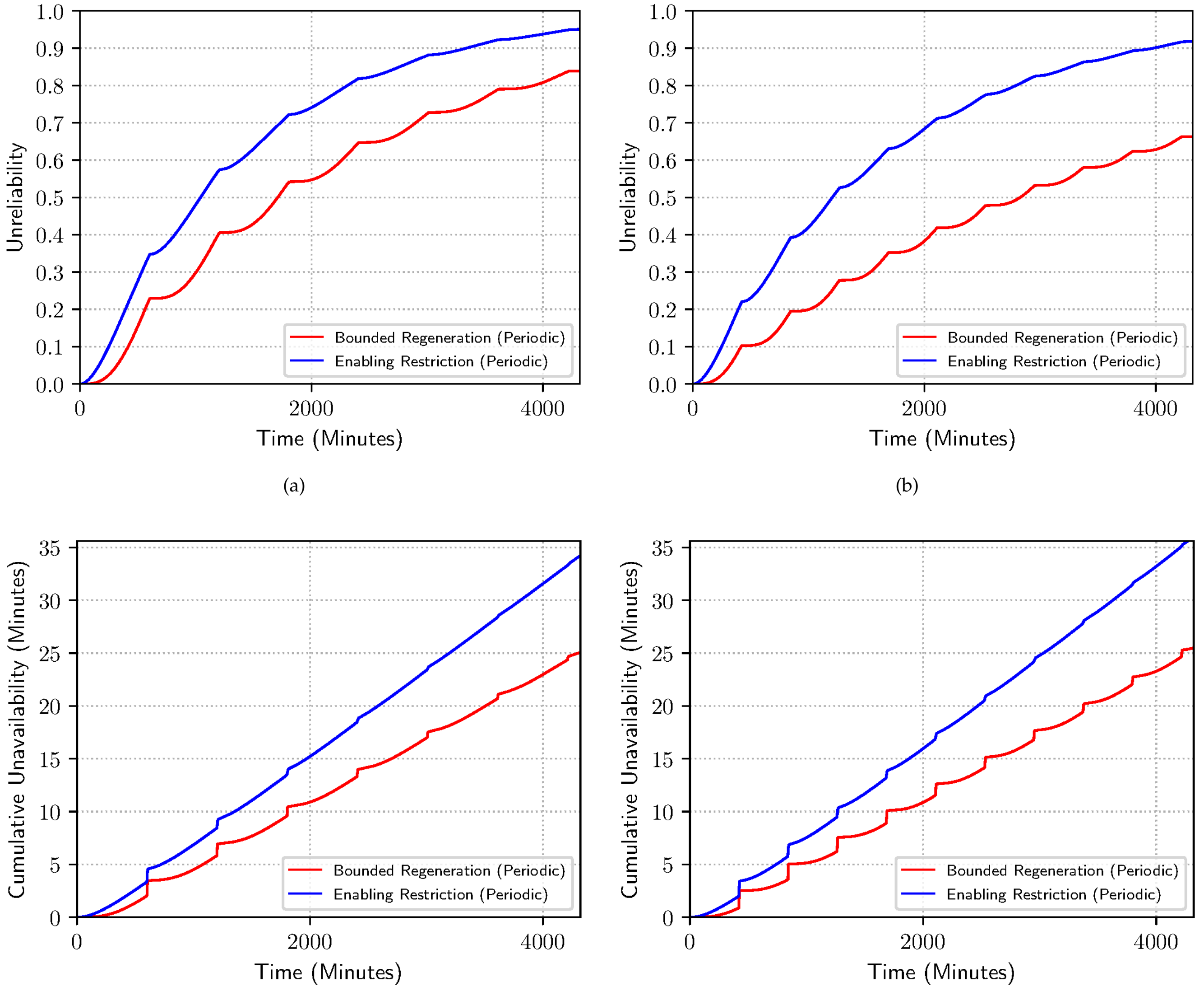
Note that, the model of Fig. 3a (under enabling restriction) achieves larger values of both the considered measures with respect to the model of Fig. 3b (under bounded regeneration restriction), suggesting that, although the two models have transitions characterized by distributions with the same expected value, the different nature of their analytical forms produces significantly different transient behaviors. Also note that, as the rejuvenation period increases, the difference between the computed measures decreases (i.e., blue and red curves become closer), given that beneficial rejuvenation effects are reduced. It is also worth noting that, when the optimal rejuvenation period is computed through the model of Fig. 3a rather than through the model of Fig. 3b, a significant increase in unreliability is observed (by comparing the red curves in Figs. 5a and 5b), thus confirming the inaccurate rejuvenation policy obtained when considering the SAR model under the enabling restriction.

3.4.2 Remarks

When applied in our SDN scenario, the time-based policy rejuvenates the SDN-controller at regular intervals, requiring that only keepalived messages be exchanged, and thus minimizing the number of monitoring messages within the control plane. However, periodic rejuvenation is based on the assumption of constant aging and failure rates, which typically holds if the workload of the system under analysis is mostly constant [15]. Therefore, this policy may not be suitable for SDN controllers subject to variations in traffic intensity [38], which results in aging steps having non-Exponential duration, as occurring in our scenario (i.e., duration of transitions `error` and `fail` in Fig. 3b).

4 Mixed Software Rejuvenation

In this section, we present a novel policy combining time-based and inspection-based software rejuvenation by referring to our SDN scenario (Section 4.1), and we define the policy through a SAR model under the bounded regeneration restriction (Section 4.2). Then, we derive optimal inspection and rejuvenation times (Section 4.3) and we compare dependability-related measures obtained by applying mixed and time-based rejuvenation, discussing the impact on the considered SDN scenario (Section 4.4).



(a)

(b)

(c)

(d)

Fig. 5. For the model of Fig. 3a (blue line), with underlying MRGP under the enabling restriction, and the model of Fig. 3b (red line), with underlying MRGP under the bounded regeneration restriction: transient unreliability with rejuvenation period equal to (a) $600\,\mathrm{min}$ and (b) $420\,\mathrm{min}$, and cumulative transient unavailability with rejuvenation period equal to (c) $600\,\mathrm{min}$ and (d) $420\,\mathrm{min}$.

4.1 Mixed software rejuvenation policy

We define the mixed rejuvenation policy by combining the time-based and inspection-based policies. Specifically, we perform up to a fixed number of inspections on the state of the system, and we use their outcomes to determine the time to the next inspection or rejuvenation, which we term the Time To Wait (TTW). When the TTW elapses, the system state is inspected. As a result of all the inspections executed since the last rejuvenation or repair, the next rejuvenation can be triggered immediately (as in inspection-based approaches) or it can be postponed. After at most a fixed number of inspections, rejuvenation is always triggered, irrespective of the system state (as in time-based approaches). Equivalently, our mixed rejuvenation strategy can be described as a time-based rejuvenation where the rejuvenation period (initially equal to the maximum delay allowed by the sequence of all possible inspections) is reduced at runtime when inspections suggest that the system has aged.

As discussed in Section 2.1, an inspection consists of a threshold-based diagnostic test on some aging indicator, and its result can be either positive, i.e., green (G, meaning that the threshold has not been exceeded), or negative, i.e., red (R, meaning that the threshold has been exceeded). An inspection is characterized by specificity $\gamma(t)$ and sensitivity $\rho(t)$, i.e., $\gamma(t) = \frac{\mathrm{TN}(t)}{\mathrm{TN}(t)+\mathrm{FP}(t)}$ is the fraction of correctly detected negatives and $\rho(t) = \frac{\mathrm{TP}(t)}{\mathrm{TP}(t)+\mathrm{FN}(t)}$ is the fraction of correctly detected positives, where $\mathrm{TP}(t)$, $\mathrm{FP}(t)$, $\mathrm{TN}(t)$, and $\mathrm{FN}(t)$ are the number of true positives, false positive, true negatives, and false negatives, respectively, detected at time t since the completion of the last rejuvenation or repair. Note that both specificity and sensitivity may vary over time, reflecting the fact that aging effects become more evident with time, and thus aging states can be detected more easily [30]. To facilitate the interpretability of results, in Section 4.4, we perform experiments with both constant and time-varying values of specificity and sensitivity.

The proposed mixed rejuvenation policy turns out to be suitable in scenarios like SDN, where monitoring is not continuous, and thus applying trend detection techniques is not appropriate. Rather, in these contexts, it is usually more suitable to implement threshold-based detection techniques on software aging indicators [15]. However, threshold-based techniques suffer from temporary peaks in the system workload, which could temporarily alter the aging indicators and lead to incorrect aging estimations. Therefore, given the dependence of aging indicators on the network transient behavior, the repetition of inspections appears reasonable and convenient to minimize the impact of transient workloads.

4.2 SAR model under bounded regeneration restriction

Fig. 6 provides the STPN model of the mixed rejuvenation policy, whose underlying stochastic process is an MRP under the bounded regeneration restriction. As shown in Fig. 6a, the processes of aging (transition `error`), failure (transition `failure`), failure detection (transition `detect`), repair (transition `repair`), and rejuvenation (transitions `rejFromUp`, `rejFromErr`, and `rejFromDown`) are modeled as in the STPN of time-based rejuvenation shown in Fig. 3b. Conversely, the DET transition `waitClock` modeling the rejuvenation period in Fig. 3b is replaced by a submodel representing the execution of inspections, as detailed in Fig. 6b. Thus, to stop the rejuvenation process during system repair, the update function of transition `detect` assigns zero tokens not only to place `Clock` but also to any other place that is included in the inspection submodel.

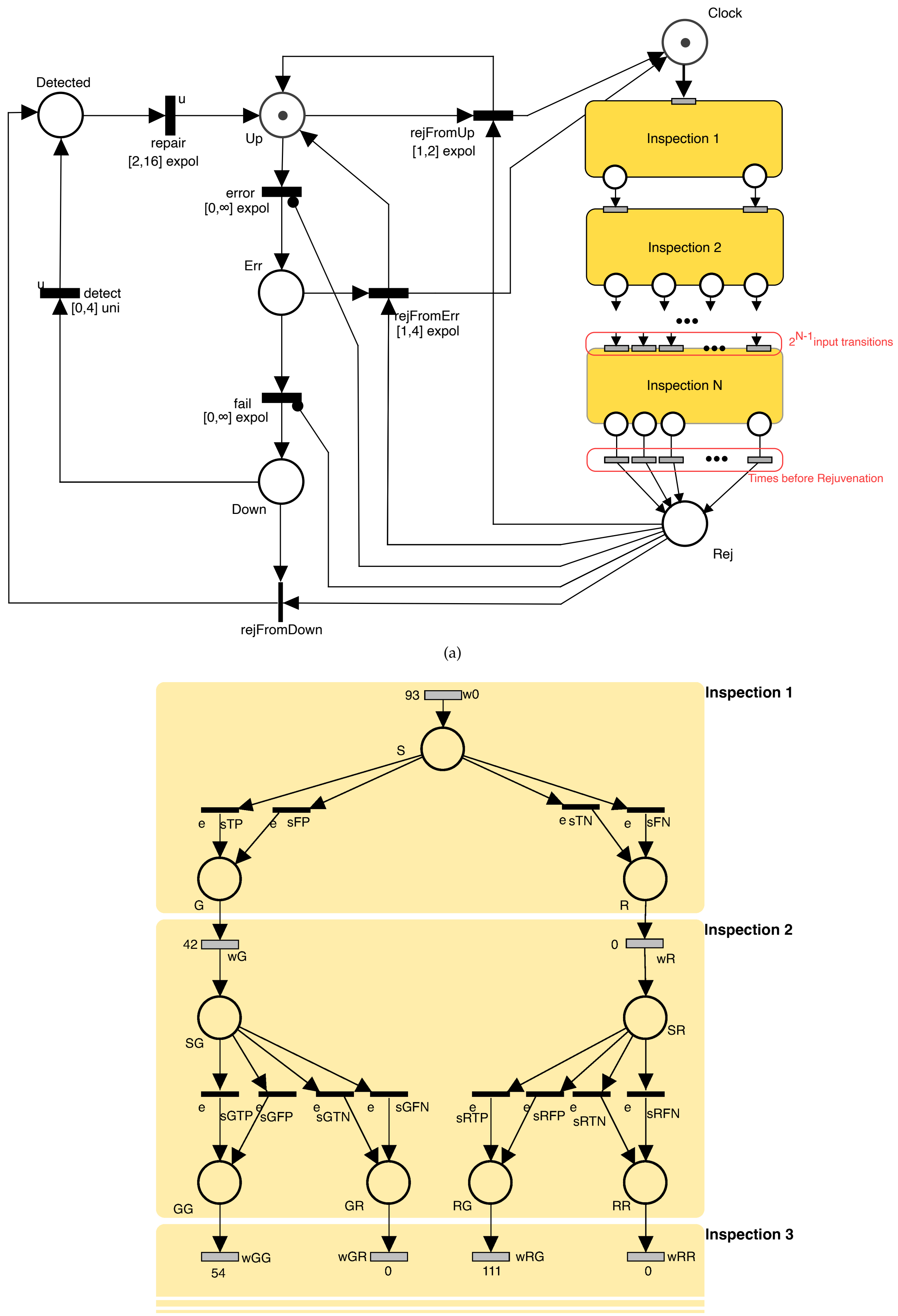
Each inspection has a positive outcome (i.e., true or false positive) or negative outcome (i.e., true or false negative), thus yielding four possible outputs. For each inspection i_n with $n \geqslant 1$ and, if $n > 1$, for each sequence $o_1, \ldots, o_{n-1}$ of outcomes of previous inspections $i_1, \ldots, i_{n-1}$, respectively, the execution of i_n is modeled by a random switch between 4 IMM transitions, defining an STPN submodel consisting of the following elements, as shown in Fig. 6b:

- A DET transition `w0` models the TTW T_0 between the last rejuvenation or repair and the first inspection i_1. For the next inspections, a DET transition models the TTW T_{n-1} between inspections i_{n-1} and i_n $\forall\, n > 1$, and its name encodes the sequence of outcomes o_1, …, o_{n-1}, e.g., `wG` and `wR` model the TTW between i_1 and i_2 if $o_1 = \mathrm{G}$ and $o_1 = \mathrm{R}$, respectively.
- A place models the inspection execution: for the first inspection i_1, place `S`; for subsequent inspections, a place whose name encodes the sequence of outcomes $o_1, \ldots, o_{n-1}$ $\forall\, n > 1$, e.g., `SG` and `SR` for inspection i_2 and outcome $o_1 = \mathrm{G}$ and $o_1 = \mathrm{R}$, respectively.
- Two places model the outcomes G and R of inspection i_n and their names encode such outcomes and, for inspections after the first one, the sequence of outcomes o_1, …, o_{n-1} of previous inspections $\forall\, n > 1$, e.g., places `G` and `R` for inspection i_1, places `GG` and `GR` for inspection i_2 and outcome $o_1 = \mathrm{G}$, and places `RG` and `RR` for inspection i_2 and outcome $o_1 = \mathrm{R}$.
- Four IMM transitions model the inspection outputs (i.e., true/false positive, true/false negative). They have the place modeling the inspection execution as input place, the place modeling the represented inspection outcome as output place, and their names encode the sequence of observed inspection outcomes o_1, …, o_n), e.g., for inspection i_1, transitions `sTP`, `sFP`, `sTN`, and `sFN` represent a true positive, a false positive, a true negative, and a false negative, respectively, while for inspection i_2 and inspection outcome $o_1 = G$, transitions `sGTP`, `sGFP`, `sGTN`, and `sGFN` represent a true positive, a false positive, a true negative, and a false negative, respectively.
  The two concurrent transitions modeling a true positive and a false negative (e.g., `sTP` and `sFN`, respectively) have enabling function "`Up==1`" and probabilistic weight P and $(1 - \mathrm{P})$, respectively, where $\mathrm{P} = \rho(\sum_{k=0}^{n-1} T_k)$ is the sensitivity value at the time of inspection i_n given the outcomes $o_1, \ldots, o_{n-1}$ (T_0 being the TTW between the last rejuvenation or repair and the first inspection i_1, and T_k the TTW between inspections i_{k-1} and i_k $\forall\, k > 1$). Similarly, the two concurrent transitions modeling a true negative and a false positive (e.g., `sTN` and `sFP`, respectively) have enabling function "`Up==0`" and probabilistic weight Γ and $(1 - \Gamma)$, respectively, where $\Gamma = \gamma(\sum_{k=0}^{n-1} T_k)$ is the specificity value at the time of inspection i_n given the outcomes $o_1, \ldots, o_{n-1}$.

Therefore, the submodel of inspection i_n consists of:

- 2^{n-1} DET transitions modeling: *i*) the time between the last rejuvenation or repair and inspection i_1 if $n = 1$, and *ii*) the TTW between i_{n-1} and i_n for each possible sequence of outcomes $o_1, \ldots, o_{n-1}$ for inspections $i_1, \ldots, i_{n-1}$, respectively, if $n > 1$;
- 2^{n-1} places modeling the execution of: *i*) inspection i_1 if $n = 1$, and *ii*) inspection i_n for each possible sequence of outcomes $o_1, \ldots, o_{n-1}$ for inspections i_1, …, i_{n-1}, respectively, if $n > 1$;
- 2^{n+1} IMM transitions modeling the four inspection outputs for each possible sequence of outcomes o_1, …, o_{n-1} for inspections $i_1, \ldots, i_{n-1}$, respectively;
- 2^n places modeling the two inspection outcomes for each possible sequence of outcomes $o_1, \ldots, o_{n-1}$ for inspections $i_1, \ldots, i_{n-1}$, respectively.
- 2^N DET transitions modeling the TTW between inspection i_N and the next rejuvenation for each possible sequence of outcomes o_1, …, o_N for inspections $i_1, \ldots, i_N$, respectively.

Note that, despite the significant number of transitions, the maximum degree of concurrency between non-EXP transitions is equal to two for the inspection submodel, and thus it is equal to three for the whole model, which makes regenerative analysis based on the method of stochastic state classes viable [19], [20]. In fact, along each possible sequence of observed inspection outcomes, a single DET transition modeling a TTW is enabled at a time, and only two IMM transitions modeling a random switch between inspection outcomes are enabled at a time. Note that the analysis is anyway challenging due to the possible significant length of behaviors (in terms of transition firings) between consecutive regeneration points (especially after the firing of transition `error`). Also note that we need to compute all the TTW values only in the case that we need to implement a completely pre-calculated rejuvenation policy.



(a)

(b)

Fig. 6. (a) STPN model of mixed rejuvenation and (b) detail on the submodel representing inspections on the system state.

4.3 Optimal inspection and rejuvenation times

We define an inductive iterative approach to determine the TTWs of the model of Fig. 6 (i.e., the values of the DET transitions). Specifically, we consider the transient unreliability $u(t, O_t)$ up to the next inspection given the sequence of observations since the last rejuvenation or repair:

$$u(t, O_t) := P\,(\text{system failure by } t \,|\, O_t) \tag{2}$$

where O_t is the sequence of observations taken since when the last rejuvenation or repair has been completed up to time t, each observation consisting of an inspection outcome and an inspection time. We derive the TTW as the maximum time T for which $u(t, O_t)$ per time unit is not larger than a given threshold ϵ, modeling the accepted system failure rate:

$$u(t, O_t) \leqslant \epsilon \cdot t. \tag{3}$$

At step 1 (base case of the inductive process), we determined the TTW between the completion of the last rejuvenation or repair and the first inspection i_1, i.e., the value T_0 of the DET transition `w0` in Fig. 6. Given that no observation is taken in the time interval $[0, T_0)$, $u(t, O_t)$ can be derived by performing regenerative transient analysis of the model up to a time limit U with stop condition "`Down`" and computing the reward "`Down`". Then, T_0 is derived as the minimum between U and the maximum time t satisfying Eq. (3).

At step $n > 1$ (inductive case), the TTW between inspections i_{n-1} and i_n is determined for each possible sequence of outcomes $o_1, \ldots, o_{n-1}$ of inspections $i_1, \ldots, i_{n-1}$, respectively. Let $T_1, \ldots, T_{n-1}$ be the sequence of TTWs computed for $o_1, \ldots, o_{n-1}$ during the previous steps of the procedure. Then, $u(t, O_t)$ can be derived by performing regenerative transient analysis of the model up to time limit U with stop condition "`Down||p1||...||p`$n-1$`||s`n" (where `p`i denotes the place encoding the complementary outcome of $o_1, \ldots, o_i \ \forall i \in \{1, \ldots, n-1\}$ and `s`n denotes the place encoding the execution of inspection i_n given the outcome sequence $o_1, \ldots, o_{n-1}$), e.g., if $o_1 =$ G, $o_2 =$ R, and $o3 =$ G, then the stop condition is "`Down||R||GG||GRR ||SGRG`". Finally, T_n is derived as $\min\{U, t_{\max}\} - \sum_{k=0}^{n-1} T_k$ where $t_{\max}$ is the maximum time satisfying Eq. (3).

The analysis time limit U is the time beyond which rejuvenation is expected to be no longer effective. It is estimated as the expected value plus the standard deviation of the time-to-failure. In particular, let μ_{error} and μ_{fail} be the expected values of the PDFs of transitions `error` and `fail`, respectively, and let σ_{error} and σ_{fail} be their standard deviations, respectively. Then, the expected value and the variance of the time-to-failure are $\bar{\mu} = \mu_{\text{error}} + \mu_{\text{fail}}$ and $\bar{\sigma}^2 = \sigma^2_{\text{error}} + \sigma^2_{\text{fail}}$, respectively. Thus, we set the analysis time limit as $U := \bar{\mu} + \bar{\sigma}$, and we guarantee that up to N inspections are performed within time U (since the completion of the last rejuvenation or repair). In fact, if the TTW is selected equal to the analysis time limit, then the TTW of the subsequent inspections are all set equal to 0, indicating that such inspections are not necessary to improve dependability-related attributes of the system. Otherwise, if the TTW is never selected equal to the analysis time limit for a sequence of observations, then the sum of the TTWs (i.e., the time elapsed between the last rejuvenation or repair and the next rejuvenation) is strictly lower than U.

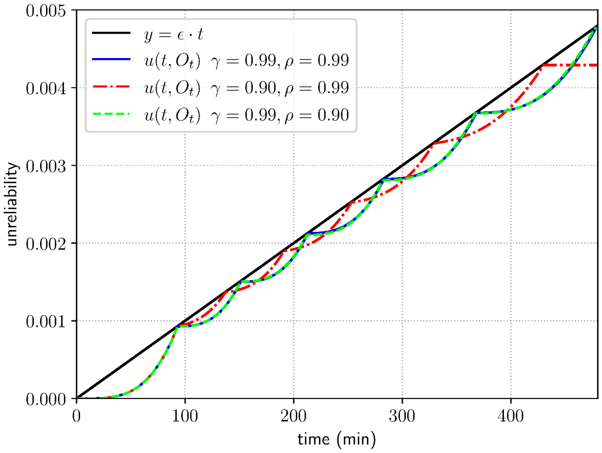


Fig. 7. Line $y = \epsilon \cdot t$ with $\epsilon = 10^{-5}$ and, for the model of Fig. 6, transient unreliability $u(t, O_t)$ given a sequence of five successful observations, for constant values of specificity γ and sensitivity ρ in {0.9,0.99}.

Fig. 7 shows $u(t, O_t)$ for a sequence of five successful observations (i.e., having outcomes $o_1 = \ldots = o_5 =$ G) since the last rejuvenation or repair, and constant values of specificity γ and sensitivity ρ in $\{0.9, 0.99\}$. The angular points of $u(t, O_t)$, i.e., the times at which it intersects the straight line $y = \epsilon \cdot t$, with $\epsilon = 10^{-5}$, represent the optimal selected TTWs. The first TTW T_0 is the same (i.e., $93\,\text{min}$) regardless of the values of γ and ρ, given that no observation has been observed yet. Then, with the same value of ρ, larger values of γ yield larger TTWs, given the increased capacity in detecting aged states, e.g., for $\rho = 0.99$, T_1 is equal to $44\,\text{min}$ for $\gamma = 0.9$ and $58\,\text{min}$ for $\gamma = 0.99$. Conversely, with the same value of γ, larger values of ρ yield equal or slightly larger TTWs, due to the unchanged ability to detect aging, e.g., for $\gamma = 0.99$, T_1 is equal to $58\,\text{min}$ for both $\rho = 0.9$ and $\rho = 0.99$. After the fifth and last inspection, the TTW T_5 (modeling the time to the next rejuvenation) is set equal to difference between $U = \mu_{\text{error}} + \mu_{\text{fail}} + \sqrt{\sigma^2_{\text{error}} + \sigma^2_{\text{fail}}} = 1706\,\text{min}$ and the sum of the previous TTWs, i.e., $\sum_{k=0}^{4} T_k$.

4.4 Dependability measures of optimal solutions

In this section, we derive the optimal inspection and rejuvenation times for both constant and time-varying values of specificity and sensitivity of diagnostic tests (Section 4.4.1), we use them to compute dependability measures (Section 4.4.2), and we discuss the impact on our SDN scenario (Section 4.4.3). In all experiments, we consider $\epsilon = 10^{-5}$.

4.4.1 *Optimal inspection and rejuvenation times*

Table 2 reports the TTWs computed through the approach of Section 4.3 for the model of Fig. 6, with $N = 4$ inspections and constant specificity γ and sensitivity ρ of diagnostic tests taking values in $\{0.9, 0.95, 0.99\}$.

First, we consider γ equal to ρ (first three columns). As γ and ρ increase, a positive test outcome (G) is more likely to be true and consequently the next inspection is postponed by increasing the TTWs until the time bound $U = 1706$ is reached and rejuvenation is triggered. Next, we consider cases where $\gamma \neq \rho$. We observe that, when

$\gamma = 0.90$, TTWs are similar for $\rho = 0.90$ (first column) and $\rho = 0.99$ (fourth column); when $\gamma = 0.99$, TTWs are similar for $\rho = 0.90$ (third column) and $\rho = 0.99$ (fifth column). Sensitivity ρ has lower impact on TTWs than γ since the misclassification of a positive state (i.e., an R inspection outcome when the system has not aged) triggers a rejuvenation, resulting in low unreliability. Independently of the values of γ and ρ, if an inspection yields a negative outcome (i.e., R), a new inspection is immediately repeated (i.e., for transitions ending with "R", the optimal TTW is 0).

Note that γ and ρ could be different for different inspections, or even depend on previous outcomes. We consider two cases with time-varying values, where both $\gamma(t)$ and $\rho(t)$ are equal to $p \in \{0.5, 0.8\}$ up to time $100\,\mathrm{min}$, linearly increase from time $100\,\mathrm{min}$ to $200\,\mathrm{min}$ until reaching 0.95, and are equal to 0.95 after $200\,\mathrm{min}$, i.e.,

$$\gamma(t) = \begin{cases} p & \forall\, t \in [0, 100) \\ p + \frac{t-100}{200-100}(0.95 - p) & \forall\, t \in [100, 200) \\ 0.95 & \forall\, t \in (200, \infty) \end{cases} \tag{4}$$

and $\rho(t) = \gamma(t)$. Note that this trend represents the fact that the effectiveness of diagnostic tests based on aging indicators increases over time [30]. Other functions of time could be analyzed, including different trends for $\gamma(t)$ and $\rho(t)$, with no impact on the computational complexity of the proposed approach. The obtained TTWs are reported in Table 3. When $p = 0.8$ (second column), γ and ρ increment shortly after the initial TTW $T_0 = 93\,\mathrm{min}$, reaching 0.95 after three inspections; compared to a constant $\gamma = \rho = 0.95$ in Table 2, the trend is similar but TTWs are much lower

TABLE 2
TTWs (expressed in min) for the model of Fig. 6 with $N = 4$ inspections and constant specificity γ and sensitivity ρ of diagnostic tests.

Trans.	$\gamma = 0.90$ $\rho = 0.90$	$\gamma = 0.95$ $\rho = 0.95$	$\gamma = 0.99$ $\rho = 0.99$	$\gamma = 0.90$ $\rho = 0.99$	$\gamma = 0.99$ $\rho = 0.90$
w	93	93	93	93	93
wG	42	51	58	44	58
wR	0	0	0	0	0
wGG	54	58	62	54	61
wGR	0	0	0	0	0
wRG	111	219	1613	118	1613
wRR	0	0	0	0	0
wGGG	59	65	70	62	70
wGGR	0	0	0	1	0
wGRG	158	272	1555	162	1555
wGRR	0	0	0	0	0
wRGG	247	1394	0	258	0
wRGR	0	0	0	0	0
wRRG	233	1613	1613	261	1613
wRRR	0	0	0	0	0
wGGGG	74	79	85	76	85
wGGGR	0	0	0	0	0
wGGRG	174	328	1493	185	1494
wGGRR	0	0	0	0	0
wGRGG	378	1290	0	404	0
wGRGR	0	0	0	0	0
wGRRG	493	1562	1555	543	1555
wGRRR	1	1	0	0	0
wRGGG	1255	0	0	1237	0
wRGGR	0	0	0	0	0
wRGRG	1502	1394	0	1495	0
wRGRR	0	0	0	0	0
wRRGG	1380	0	0	1352	0
wRRGR	1	0	0	0	0
wRRRG	398	1613	1613	503	1613
wRRRR	0	0	0	0	0

TABLE 3
TTWs (expressed in min) for the model of Fig. 6 with $N = 4$ inspections and time-varying specificity $\gamma(t)$ and sensitivity $\rho(t)$ of diagnostic tests.

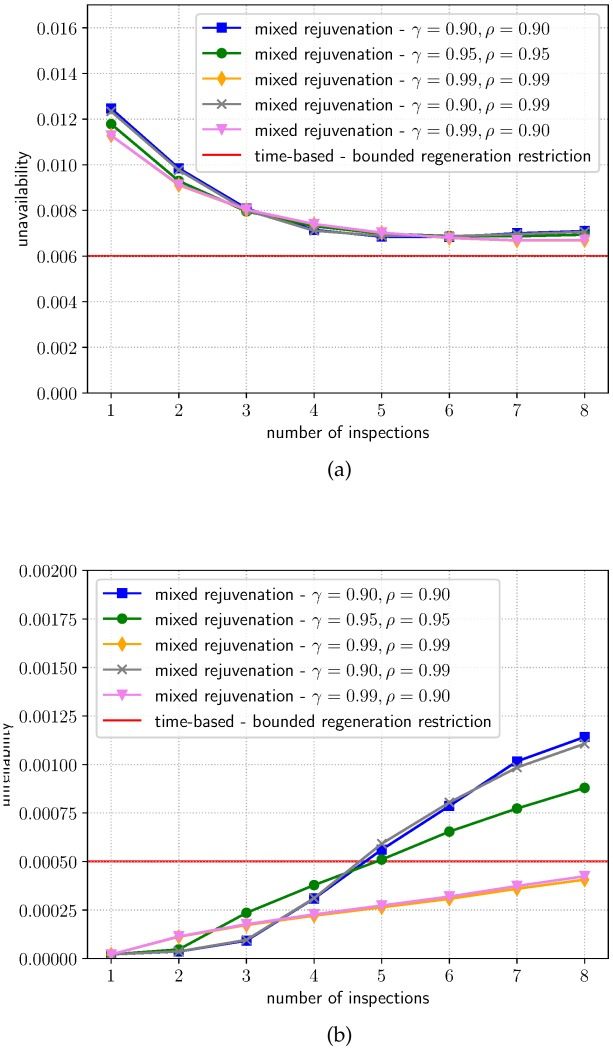
Trans.	$\gamma(t) = \rho(t) \in [0.5, 0.95]$	$\gamma(t) = \rho(t) \in [0.8, 0.95]$
w	93	93
wG	0	21
wR	0	0
wGG	11	54
wGR	11	1
wRG	11	47
wRR	11	0
wGGG	54	66
wGGR	48	1
wGRG	54	124
wGRR	48	2
wRGG	54	140
wRGR	48	1
wRRG	54	91
wRRR	48	0
wGGGG	135	90
wGGGR	2	0
wGGRG	132	204
wGGRR	2	0
wGRGG	135	253
wGRGR	2	1
wGRRG	132	301
wGRRR	2	11
wRGGG	135	262
wRGGR	2	0
wRGRG	132	517
wRGRR	2	0
wRRGG	135	701
wRRGR	2	0
wRRRG	132	160
wRRRR	2	0

due to the inaccuracy of the initial inspection outcomes. When $p = 0.5$ (first column), the TTW after each inspection is nearly the same up to the third test, regardless of the outcome, since G and R are equiprobable up to time $100\,\mathrm{min}$. Then, γ and ρ start increasing, and a trend similar to the case with $p = 0.8$ is observed, though with even lower TTWs due to the higher inaccuracy of the initial test outcomes.

4.4.2 Unreliability and cumulative unavailability

We evaluate the steady-state unavailability and unreliability with mixed rejuvenation, varying the number of inspections in $\{1, 2, \ldots, 8\}$ and considering constant and time-varying specificity γ and sensitivity ρ of diagnostic tests. Fig. 8 shows the results for $\gamma = \rho \in \{0.9, 0.95, 0.99\}$, while Fig. 9 considers $\gamma(t)$ and $\rho(t)$ as defined in Eq. (4) for $p = 0.5$ and $p = 0.8$. To this end, we perform regenerative steady-state analysis [20] of the model of Fig. 6 using the TTWs derived in Section 4.3 through the SIRIO library of the ORIS tool. The steady-state unavailability can be derived by computing the reward “`If(Down+Detected>0||Rej>0,1,0)`”, and steady-state unreliability by computing the reward “`Down`”.

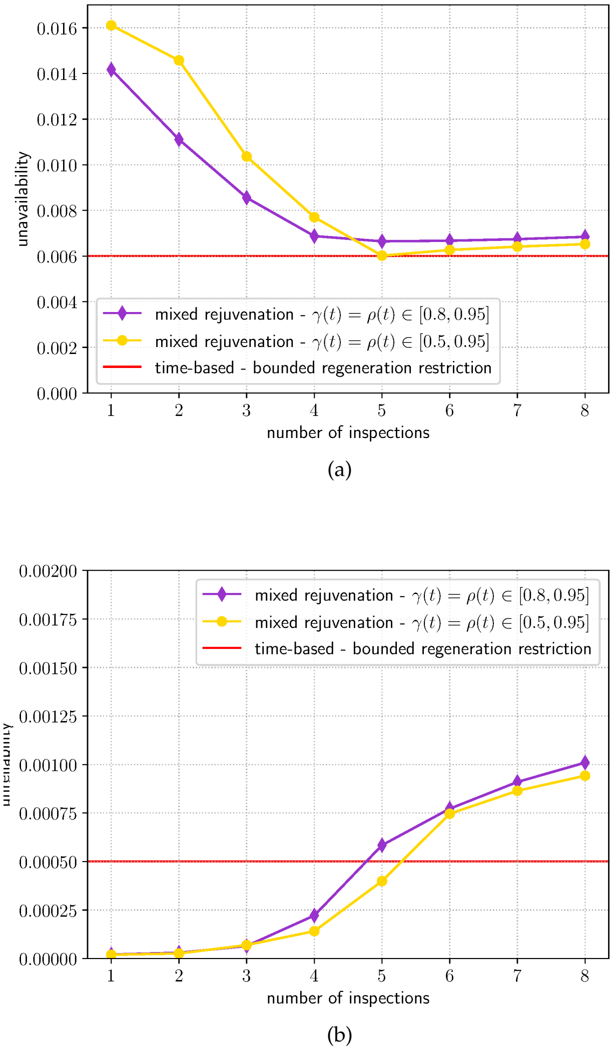
As shown in Fig. 8a, for mixed rejuvenation, the steady-state unavailability almost decreases as the number of inspections increases. In fact, with a larger number of inspections, rejuvenation become more effective, being triggered in very short times if the system is aged, and gradually postponed if it is still correctly working. With $N \leqslant 3$ and $N \geqslant 6$, larger values of γ and ρ yield lower unavailability (blue, green, and orange curves), given the better capacity to detect aging effects and avoid unexpected failures. With $N = 4$ and $N = 5$, lower values of γ and ρ yield lower steady-state unavailability, which can be explained as an attempt to counterbalance the lower confidence in the inspection outcomes, i.e., lower TTWs yield more frequent inspections, thus increasing the probability of detecting aging during the tests and triggering rejuvenation before failure detection and consequent repair. As discussed in Section 4.4.1, with



(a)

(b)

Fig. 8. For mixed rejuvenation (model of Fig. 6), with different number of inspections and constant values of specificity γ and sensitivity ρ of diagnostic tests, and for time-based rejuvenation (model of Fig. 3b): (a) steady-state unavailability and (b) steady-state unreliability.



(a)

(b)

Fig. 9. For mixed rejuvenation (model of Fig. 6), with different number of inspections and time-varying specificity $\gamma(t)$ and sensitivity $\rho(t)$ of diagnostic tests, and for time-based rejuvenation (model of Fig. 3b): (a) steady-state unavailability and (b) steady-state unreliability.

the same value of γ, variations of ρ yield negligible effects (the gray curve nearly overlaps with the blue curve, and the pink curve nearly overlaps with the orange curve).

In Fig. 9a, a trend similar to that of Fig. 8a is observed with $N \leqslant 4$, with slightly larger unavailability (with respect to the case with $\gamma = \rho = 0.95$ of Fig. 8a) mainly due to the initially lower specificity values. With a larger number of inspections, the last inspections are performed in both cases with $\gamma = 0.95$, yielding comparable unavailability.

As shown in Fig. 8b, for mixed rejuvenation, the steady-state unreliability increases with the number of inspections, given that a larger number of inspections yields larger TTW values, delaying rejuvenation and increasing the failure risk. We remark that the transient unreliability per time unit is guaranteed to be lower than a given threshold for any combination of inspection outcomes. From $N = 5$ on, larger values of γ and ρ yield lower unreliability, as expected given the better capacity to detect aging effects. Note that also the unreliability is sensitive to changed of γ rather than ρ.

In Fig. 9b, a trend similar to that of Fig. 8b is observed, with almost slightly larger unreliability (with respect to the case with $\gamma = \rho = 0.95$ of Fig. 8b) mainly due to the initially lower specificity values. With $p = 0.5$ (yellow curve), unreliability is slightly lower than with $p = 0.8$ (violet curve), since lower TTWs yield more frequent rejuvenation.

Overall, using $N = 4$ inspections, mixed rejuvenation reaches a trade-off between steady-state unavailability and unreliability, achieving slightly larger unavailability with respect to time-based rejuvenation while outperforming it in terms of unreliability, which can be nearly halved.

4.4.3 Remarks

In the considered SDN scenario, the proposed mixed rejuvenation policy turns out to be particularly suitable for keeping the monitoring overhead under control, as it enables deciding in advance an arbitrary number of diagnostic tests to be performed on an SDN-controller between two rejuvenation actions. Moreover, unlike the time-based rejuvenation, the mixed rejuvenation is more robust to workload variations that indirectly alter the aging speed and thus also the failure rate of the SDN-controller, which can in fact be managed by the inspection. Furthermore, the possibility to perform multiple diagnostic tests between two rejuvenation operations make this strategy robust also to sudden load peaks, which could in fact lead to altered values of the aging indicators and thus to altered test results.

5 Conclusions

We presented a mixed software rejuvenation policy, which combines time-based and inspection-based rejuvenation, using warnings emitted by diagnostic tests to trigger early rejuvenation. The policy is defined by a non-Markovian model with an underlying MRGP under the bounded regeneration restriction, enabling the definition of an efficient procedure to determine the time to the next inspection or rejuvenation, computed as the time at which the transient unreliability per time unit up to the next test, given the test outcomes since the last rejuvenation, equals a given threshold.

Experimental results show that, with respect to the time-based rejuvenation policy, the mixed rejuvenation policy improves the system reliability while reducing its availability, which anyway comes with an acceptable downtime cost. Results also show that the SIRIO library of the ORIS tool can be effectively used to design and evaluate SAR models with an underlying MRGP beyond the enabling restriction, supporting the definition of exponomial distribution that fit multiple statistics of observed durations (e.g., to preserve the sample mean and sample variance) as well as enabling the selection of optimal inspection and rejuvenation times.

The proposed rejuvenation policy is effective in counteracting software aging in emerging computing systems like SDN. The approach is open to various extensions facilitating applicability to other contexts. In particular, the concurrency structure of the model could be changed to define different aging, failure, and repair processes, e.g., multiple steps from working to failure states. Moreover, the approach could be extended into a hybrid method by integrating measurement-based techniques to estimate stochastic parameters of the model from observations, e.g., time series encoding empirical data on the aging phenomenon.

Acknowledgments

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future"(PE00000001-program "RESTART").

The authors would like to thank Prof. Francesco Chiti (University of Florence) for fruitful discussions about SDNs.

References

[1] J. Alonso, R. Matias, E. Vicente, A. Maria, and K. S. Trivedi. A comparative experimental study of software rejuvenation overhead. *Performance Evaluation*, 70(3):231–250, 2013.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. on Dependable and Secure Computing*, 1(1):11–33, 2004.

[3] J. Bai, X. Chang, F. Machida, L. Jiang, Z. Han, and K. S. Trivedi. Impact of service function aging on the dependability for MEC service function chain. *IEEE Trans. on Dependable and Secure Computing*, 2022.

[4] Y. Bao, X. Sun, and K. S. Trivedi. A workload-based analysis of software aging, and rejuvenation. *IEEE Trans. on Reliability*, 54(3):541–548, 2005.

[5] M. Biagi, L. Carnevali, M. Paolieri, T. Papini, and E. Vicario. Exploiting non-deterministic analysis in the integration of transient solution techniques for Markov regenerative processes. In *Int. Conf. on Quantitative Evaluation of Sys.*, pages 20–35. Springer, 2017.

[6] L. Carnevali, R. German, F. Santoni, and E. Vicario. Compositional Analysis of Hierarchical UML Statecharts. *IEEE Trans. on Software Engineering*, 48(12):4762–4788, 2021.

[7] L. Carnevali, M. Paolieri, R. Reali, L. Scommegna, and E. Vicario. A Markov Regenerative Model of Software Rejuvenation Beyond the Enabling Restriction. In *Int. Symp. on Software Reliability Engineering Workshops*, pages 138–145. IEEE, 2022.

[8] V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. Vaidyanathan, and W. P. Zeggert. Proactive management of software aging. *IBM Journal of Research and Development*, 45(2):311–332, 2001.

[9] D. Chen and K. S. Trivedi. Closed-form analytical results for condition-based maintenance. *Reliability Engineering & System Safety*, 76(1):43–51, 2002.

[10] D. Chen and K. S. Trivedi. Optimization for condition-based maintenance with semi-Markov decision process. *Reliability engineering & system safety*, 90(1):25–29, 2005.

[11] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Markov regenerative stochastic Petri nets. *Performance evaluation*, 20(1-3):337–357, 1994.

[12] D. Cotroneo, R. Natella, and R. Pietrantuono. Predicting aging-related bugs using software complexity metrics. *Performance Evaluation*, 70(3):163–178, 2013.

[13] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo. A survey of software aging and rejuvenation studies. *ACM Journal on Emerging Technologies in Computing Systems*, 10(1):1–34, 2014.

[14] T. Dohi, K. Goseva-Popstojanova, and K. S. Trivedi. Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule. In *Proceedings. 2000 Pacific Rim International Symposium on Dependable Computing*, pages 77–84. IEEE, 2000.

[15] T. Dohi, K. S. Trivedi, and A. Avritzer. *Handbook of Software Aging and Rejuvenation: Fundamentals, Methods, Applications, and Future Directions*. World Scientific, 2020.

[16] S. Garg, A. Puliafito, M. Telek, and K. S. Trivedi. Analysis of software rejuvenation using Markov regenerative stochastic Petri net. In *Proceedings of Sixth International Symposium on Software Reliability Engineering. ISSRE'95*, pages 180–187. IEEE, 1995.

[17] M. Grottke, R. Matias, and K. S. Trivedi. The fundamentals of software aging. In *IEEE Int. Conf. on software reliability engineering workshops (ISSRE Wksp)*, pages 1–6. IEEE, 2008.

[18] M. Grottke, A. P. Nikora, and K. S. Trivedi. An empirical investigation of fault types in space mission system software. In *IEEE/IFIP Int. Conf. on dependable sys. & networks*, pages 447–456. IEEE, 2010.

[19] A. Horváth, M. Paolieri, L. Ridi, and E. Vicario. Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation*, 69(7-8):315–335, 2012.

[20] A. Horváth, M. Paolieri, and E. Vicario. Equilibrium analysis of Markov regenerative processes. In *Proceedings of QEST*, volume 14287 of *LNCS*, pages 172–187. Springer, 2023.

[21] M. M. Hosseini, R. M. Kerr, and R. B. Randall. An inspection model with minimal and major maintenance for a system with deterioration and Poisson failures. *IEEE Trans. on Reliability*, 49(1):88–98, 2000.

[22] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton. Software rejuvenation: Analysis, module and applications. In *Twenty-fifth international symposium on fault-tolerant computing. Digest of papers*, pages 381–390. IEEE, 1995.

[23] K. Jia, X. Yu, C. Zhang, W. Hu, D. Zhao, and J. Xiang. Software aging prediction for cloud services using a gate recurrent unit neural network model based on time series decomposition. *IEEE Trans. on Emerging Topics in Computing*, 2023.

[24] J. M. Kizza. Internet of things: growth, challenges, and security. In *Guide to Computer Network Security*, pages 557–573. Springer, 2024.

[25] V. G. Kulkarni. *Modeling and analysis of stochastic systems*. Chapman and Hall/CRC, 2016.

[26] J. Liu and L. Meng. Integrating artificial bee colony algorithm and bp neural network for software aging prediction in IoT environment. *IEEE Access*, 7:32941–32948, 2019.

[27] F. Machida, D. S. Kim, and K. S. Trivedi. Modeling and analysis of software rejuvenation in a server virtualized system with live vm migration. *Performance Evaluation*, 70(3):212–230, 2013.

[28] F. Machida and N. Miyoshi. An optimal stopping problem for software rejuvenation in a job processing system. In *IEEE Int. Symp. on Softw. Rel. Eng. Workshops*, pages 139–143. IEEE, 2015.

[29] F. Machida, V. F. Nicola, and K. S. Trivedi. Job completion time on a virtualized server with software rejuvenation. *ACM Journal on Emerging Technologies in Computing Systems*, 10(1):1–26, 2014.

[30] R. Matias, A. Andrzejak, F. Machida, D. Elias, and K. Trivedi. A systematic differential analysis for fast and robust detection of software aging. In *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, pages 311–320. IEEE, 2014.

[31] R. Pietrantuono and S. Russo. A survey on software aging and rejuvenation in the cloud. *Software Quality Journal*, 28(1):7–38, 2020.

[32] L. V. Ruchel, R. C. Turchetti, and E. T. de Camargo. Evaluation of the robustness of SDN controllers ONOS and ODL. *Computer Networks*, 219:109403, 2022.

[33] B. Tola, Y. Jiang, and B. E. Helvik. Model-driven availability assessment of the NFV-MANO with software rejuvenation. *IEEE Trans. on Network and Service Management*, 18(3):2460–2477, 2021.

[34] D. Trihinas, G. Pallis, and M. D. Dikaiakos. Low-cost adaptive monitoring techniques for the internet of things. *IEEE Trans. on Services Computing*, 14(2):487–501, 2018.

[35] K. Vaidyanathan, R. E. Harper, S. W. Hunter, and K. S. Trivedi. Analysis and implementation of software rejuvenation in cluster systems. In *ACM SIGMETRICS Int. Conf. on Measurement and modeling of computer systems*, pages 62–71, 2001.

[36] E. Vicario, L. Sassoli, and L. Carnevali. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Trans. on Software Engineering*, 35(5):703–719, 2009.

[37] P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, and C. M. Machuca. Characterization of failure dynamics in SDN controllers. In *Int. work. on resilient net. design and model.*, pages 1–7. IEEE, 2017.

[38] P. Vizarreta, C. Sieber, A. Blenk, A. Van Bemten, V. Ramachandra, W. Kellerer, C. Mas-Machuca, and K. Trivedi. ARES: A Framework for Management of Aging and Rejuvenation in Softwarized Networks. *IEEE Trans. Net. and Serv. Manag.*, 18(2):1389–1400, 2020.

[39] D. Wang, W. Xie, and K. S. Trivedi. Performability analysis of clustered systems with rejuvenation under varying workload. *Performance Evaluation*, 64(3):247–265, 2007.

[40] C. Weng, J. Xiang, S. Xiong, D. Zhao, and C. Yang. Analysis of software aging in Android. In *Int. Symp. on Software Reliability Engineering Workshops (ISSREW)*, pages 78–83. IEEE, 2016.

[41] W. Whitt. Approximating a Point Process by a Renewal Process, I: Two Basic Methods. *Op. Res.*, 30(1):125–147, 1982.

[42] H. Yahyaoui, M. F. Zhani, O. Bouachir, and M. Aloqaily. On minimizing flow monitoring costs in large-scale software-defined network networks. *International Journal of Network Management*, 33(2):e2220, 2023.

[43] M.-L. Yin, J. E. Angus, and K. S. Trivedi. Optimal preventive maintenance rate for best availability with hypo-exponential failure distribution. *IEEE Trans. on reliability*, 62(2):351–361, 2013.

[44] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, J. Yang, L. Zhang, K. Cheng, and X. Xiao. Fault management in software-defined networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(1):349–392, 2018.

[45] J. Zhao, Y. Wang, G. Ning, K. S. Trivedi, R. Matias Jr, and K.-Y. Cai. A comprehensive approach to optimal software rejuvenation. *Performance Evaluation*, 70(11):917–933, 2013.

Laura Carnevali is Associate Professor of Computer Science at the University of Florence, Italy, where she received the Ph.D. in Informatics, Multimedia, and Telecommunications Engineering (2010). Her research activities are mainly focused on quantitative evaluation of stochastic models, with specific application to the analysis of dependability-related attributes of concurrent timed systems in various domains.

Marco Paolieri is a Senior Research Associate at the University of Southern California, Los Angeles, USA. He received his Ph.D. in Computer Science, Systems, and Telecommunications (2015) and his M.S. in Computer Engineering (2011) from the University of Florence, Italy. His research interests focus on stochastic modeling and quantitative evaluation of performance and reliability in concurrent and distributed systems.

Riccardo Reali is a post-doctoral researcher at the University of Florence, where he received the bachelor and master degrees in Information Engineering and the Ph.D. degree in Smart Computing. His scientific activity is focused on quantitative analysis of non-Markovian through compositional and heuristics methods, software engineering methodologies and software architectures.

Leonardo Scommegna is Assistant Professor of computer science with the Department of Information Engineering, University of Florence, Italy, where he obtained his PhD in Smart Computing in 2023. His research interests lie in the area of software engineering, with particular emphasis on software architectures, software testing, and model-based development.

Enrico Vicario is a Professor of Computer Science and Engineering and Head of the Department of Information Engineering at the University of Florence, Italy. His research is in the area of Software Engineering, with a present focus on quantitative evaluation of stochastic models, software architecture and methodologies, and on their connection through Model Driven Engineering practices.

Appendix: Analysis of MRGP Processes

This appendix summarizes the use of stochastic state classes for the transient and steady-state analysis of STPN models with underlying MRGP processes [19], [20].

Stochastic State Classes. A *stochastic state class* [36] encodes the marking of an STPN (i.e., the logical state of the underlying stochastic process) and the joint PDF of the remaining times to fire of enabled transitions (determining the next change of the logical state) and of the absolute time of the previous transition firing (which we call *age*). Starting from an initial stochastic state class in which times to fire are distributed independently, a *transient tree* [19] is computed by adding an edge $\Sigma \xRightarrow{\gamma,\mu} \Sigma'$ for each transition firing with nonzero probability: the edge is labeled with the transition name γ and with the firing probability μ; it connects the current stochastic state class Σ with a new one Σ', in which the times to fire of persistent transitions are decreased by the previous sojourn time, and thus become mutually dependent random variables. In so doing, a transient tree unfolds the state class graph of an STPN and endows it with a probability measure, providing the stochastic characterization of a continuous set of executions.

Transient Analysis. From stochastic state classes, quantitative measures can be computed for the transient regime of the STPN. Given an initial stochastic state class Σ_0 and a sequence of transition firings $\gamma_1, \gamma_2, \dots, \gamma_n$, the sequence of stochastic state classes $\Sigma_0 \xRightarrow{\gamma_1,\mu_1} \Sigma_1 \xRightarrow{\gamma_2,\mu_2} \cdots \xRightarrow{\gamma_n,\mu_n} \Sigma_n$ with $\Sigma_i = \langle m_i, D_i, f_i \rangle$ for $i = 1, \dots, n$ provides the probability μ_i of each firing event, and the resulting marking m_i and joint PDF f_i (over the support D_i) for τ_{age} and for the times to fire $\vec{\tau}$ after the firing. The probability of reaching Σ_n within time t is

$$p_{reach}(\Sigma_n, t) = \left(\prod_{i=1}^{n} \mu_i \right) \int_{D_n^{\leqslant t}} f_n(x_{age}, x_1, \dots, x_n)\, dx_{age}\, d\vec{x} \tag{5}$$

where $D_n^{\leqslant t} = \{ \langle x_{age}, \vec{x} \rangle \in D_n \mid x_{age} \leqslant t \}$ imposes that the firing of γ_n happened within time t. Similarly, the probability that, at time t, the STPN has fired all and only the transitions $\gamma_1, \gamma_2, \dots, \gamma_n$ leading to Σ_n, is

$$p_{in}(\Sigma_n, t) = \left(\prod_{i=1}^{n} \mu_i \right) \int_{D_n^{@t}} f_n(x_{age}, x_1, \dots, x_n)\, dx_{age}\, d\vec{x} \tag{6}$$

where $D_n^{@t} = \{ \langle x_{age}, \vec{x} \rangle \in D_n^{\leqslant t} \mid x_k + x_{age} > t \text{ for all } k \neq age \}$ imposes that γ_n fired within time t and the next transition firing happened strictly after time t (the marking at time t is thus m_n).

Regenerative transient analysis [19] is a technique available in the ORIS tool that is able to exploit these measures and the repetitive structure of MRGP processes. Instead of unfolding all the sequences of events within time t_{max} in a single transient tree, multiple smaller trees are computed to analyze the transient behavior between each pair of *regeneration points* [25], selected transition firings after which future evolution is independent of past history and determined by the specific *regeneration* value. Transient measures between regeneration points are encoded in a *local kernel* and a *global kernel*, and then combined through a system of Markov renewal equations to compute transient probabilities $P_{ij}(t)$ for each marking $j \in \mathcal{M}$, initial regeneration i, and time $0 \leqslant t \leqslant t_{max}$. In particular, if $\mathcal{R}$ is the set of reachable regenerations, and $\textsc{Inner}(i)$ and $\textsc{Leaves}(i)$ are, respectively, the stochastic state classes of inner nodes and leaf nodes in the transient tree enumerated from regeneration $i \in \mathcal{R}$ and limited to the next regeneration, then

$$\mathbf{P}(t) = \mathbf{L}(t) + \int_0^t d\mathbf{G}(u)\, \mathbf{P}(t-u) \tag{7}$$

where $P_{ij}(t) := P\{M(t) = j \mid X_0 = i\}$, and, denoting the initial regeneration as X_0, the next one as X_1, and the time between the two as T_1, the local and global kernels are evaluated, respectively, as

$$\begin{aligned} L_{ij}(t) &:= P\{M(t) = j, T_1 > t \mid X_0 = i\} \\ &= \sum_{\substack{\Sigma \in \textsc{Inner}(i) \text{ s.t.} \\ \Sigma \text{ has marking } j}} p_{in}(\Sigma, t) \\ G_{ik}(t) &:= P\{X_1 = k,\, T_1 \leqslant t \mid X_0 = i\} \\ &= \sum_{\substack{\Sigma \in \textsc{Leaves}(i) \text{ s.t.} \\ \Sigma \text{ has regeneration } k}} p_{reach}(\Sigma, t) \end{aligned}$$

for all $i, k \in \mathcal{R}$, $j \in \mathcal{M}$, and $0 \leqslant t \leqslant t_{max}$. The system of integral equations of Eq. (7) is solved by the ORIS Tool in the time domain using Newton–Cotes formulas: for a given step size h, $\mathbf{P}$ is evaluated numerically at all $t = 0, h, 2h, \dots, \lceil \frac{t_{max}}{h} \rceil h$ from the value of $\mathbf{L}(t)$ and the values of $\mathbf{G}$ and $\mathbf{P}$ at previous time instants [19].

Steady-State Analysis. When the underlying stochastic process of the model is an MRP (possibly with concurrent, generally-distributed timers) that reaches regenerations in a bounded number of transition firings, also steady-state analysis can be performed using the ORIS tool [20]. Visit frequencies ν_i are computed for each regeneration $i \in \mathcal{R}$ by solving the linear system $\vec{\nu} = \vec{\nu}\, \mathbf{G}(\infty)$ and then combined with mean sojourn times $T_{ij} = \int_0^\infty L_{ij}(t)$ for all $i \in \mathcal{R}$ and $j \in \mathcal{M}$ to obtain steady-state probabilities $\pi_j = (\sum_{i \in \mathcal{R}} \nu_i T_{ij}) / (\sum_{i \in \mathcal{R},\, m \in \mathcal{M}} \nu_i T_{im})$ [25].