Sequential Performance Analysis of Systems that Age and Rejuvenate

Leonardo Miranda, Cabral Lima, Daniel Sadoc Menasché
Federal University of Rio de Janeiro (UFRJ), Brazil
Email: {lleomiranda, cabrallima}@ufrj.br, sadoc@dcc.ufrj.br

Guilherme Domingues
State University of Rio de Janeiro (UERJ), Brazil
domingues@iprj.uerj.br

***Abstract*—Sequential performance analysis aims at evaluating performance indicators in an online fashion. The process stops in accordance with a pre-defined stopping rule, as soon as an anomaly that should produce an alarm is observed. Traditional sequential performance analysis techniques include CUSUM and sequential probability ratio test (SPRT). More recent techniques include the bucket algorithm, wherein tokens are accumulated into buckets when the system degrades, and removed when the system naturally recovers. If the number of tokens in the system reaches a threshold, an alarm is triggered. In this paper, we analyze sequential performance analysis algorithms applied to a system that is subject to rejuvenation. Among our results, we indicate how rejuvenation impacts the time until false alarms, and how to set the optimal rejuvenation rate accounting for the fact that systems can recover from transient performance degradation either naturally, as in standard sequential performance analysis models, or due to rejuvenation.**

***Index Terms*—Software aging, rejuvenation, Markov models**

I. Introduction

The performance of computer systems, e.g., measured through user response time or throughput, varies dynamically as a function of the load, aging and attacks, where the former may be impacted by the latter. Sequential performance analysis aims at tracking system performance to trigger alarms when performance reaches levels that require some sort of stringent intervention. Until alarms are not triggered, the system may naturally recover from transient performance degradation or may be subject to rejuvenation as scheduled by admins.

There are a number of sequential performance analysis algorithms, including CUSUM and its variations [1], and the bucket algorithm [2]. One of the fundamental tradeoffs involved in those algorithms weights detection time against false positive rate. Such tradeoff has been studied, for instance, in [2], accounting for systems that age and naturally recover, but without accounting for rejuvenation. In this work, our aim is to extend [2] to analyze the interplay between natural recovery and rejuvenation under sequential performance analysis.

We aim at answering the following questions

1) in the absence of anomalies, i.e., under the baseline workload, how long does it take for a system running sequential performance analysis, subject to rejuvenation, to trigger a false alarm?
2) in the presence of anomalies, what is the interplay between aging, natural recovery, and rejuvenation, impacting metrics such as the system downtime and time to alarms?

Table I
Our contribution

previous work on	natural recovery	rejuvenation
CUSUM and bucket algorithm [2]	✓	
aging and rejuvenation of networked systems [6]		✓
this work: bucket algorithm with rejuvenation	✓	✓

While answering the first question, we leverage literature on $M/M/1$ [3] and $M/M/1/K$ [4] queues with catastrophes to study the joint impact of natural recovery and rejuvenation on metrics of interest for sequential performance analysis. While answering the second question, we extend classical models for aging and rejuvenation [5] to account for natural recovery.

Our key contributions are twofold. First, we show efficient ways to compute the mean time until a false alarm, and indicate conditions under which a geometric approximation for the time until false alarm is accurate. Second, we introduce a Markov model to account for aging, natural recovery and rejuvenation, and show how different parameters affect system unavailability.

The remainder of this paper is organized as follows. Section II discusses related work. Section III characterizes the system under consideration, and discusses the relationship between system and model parameters. Then, Sections IV and V focus on our two main questions, respectively. Finally, Section VI concludes.

II. Related work

In this paper, we model sequential performance analysis of systems under rejuvenation using birth-death processes with catastrophes. Birth-death processes with catastrophes have been used, for instance, to analyze the spread of viruses in networks and natural disasters. To the best of our knowledge, this work is the first to establish a relationship between catastrophes and rejuvenation in the realm of sequential performance analysis (see Table I).

The relation between rejuvenation and catastrophes, in the realm of G-Networks, has been studied in [7]. In this work, we also leverage the relationship between catastrophes and rejuvenation, but in the scope of birth-death processes as opposed to G-Networks.

In the Physics literature, catastrophes are referred to as resets [8]. In [6], [9] we considered rejuvenation after anomalies, accounting for aging. In this work, we extend [6], [9] to account not only for rejuvenation [10], but also for natural system recovery, giving rise to a birth-death model with catastrophes that occur when rejuvenation is applied and the system recovers its original state.[1]

CUSUM and the bucket algorithm have been studied under the general realm of sequential performance analysis algorithms [11]. In previous works, it has been assumed that the systems where those sequential performance analysis algorithms are deployed naturally recover from transient performance degradation. In this work, in contrast, we also consider the impact of rejuvenation interventions on the dynamics of sequential performance analysis.

Romagnoli [12] considers the problem of determining the optimal rejuvenation rate, accounting for its benefits in terms of increased safety and security. It is natural to assume that in those systems a sequential hypothesis test will be continuously running to detect anomalies. Then, the work presented in this paper is complementary to [12], as it serves to shed insights on the impacts of rejuvenation on the false alarm rate.

III. System description

Next, we illustrate a simple setup wherein the contributions of this paper are applicable. To this aim, we consider a flow of arrivals towards a system, wherein the arrivals are characterized by a Markov modulated Poisson process (MMPP) [1], [13], [14].

Our goal is to detect change points in the MMPP traffic, i.e., we aim at detecting when and if the arrival intensity has changed. The only observed quantity is the queue state, that serves as a proxy for 1) the user perceived delay in an open system [1] or 2) throughput of a finite queue and/or of a closed system [15]. The key challenge relates to the fact that the queue may grow either due to transient degradation (that will naturally resolve) or due to an anomaly (that we assume corresponds to a change in the state of the MMPP). The setup is similar to that considered in [1], except that we consider that the system is subject to rejuvenation, that in the context of this work occurs at random times unrelated to the current state of the system. The resulting system, together with its parameters, is illustrated in Figure 1.

A. System characterization

Let the states of the MMPP process be characterized by G, A and R, corresponding to the good, anomalous and rejuvenation states, respectively (see Figure 1). In the good (resp., anomalous) state, the arrival rate towards the central server queue is λ_G (resp., λ_A). We assume $\lambda_A > \lambda_G$, where $\lambda_A = \lambda_G + \lambda_B$ where λ_B is the arrival rate of background packets that consume system resources. In the context of anomalies due to attacks, the background packets correspond to bad packets injected by the attacker, e.g., to cause a denial of server (DoS) attack. Let β be the rejuvenation rate.

[1]Note that in the context of this work catastrophes correspond to events with a positive connotation, namely, rejuvenation.

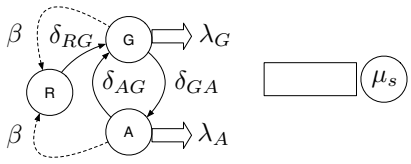


Figure 1. MMPP process

We denote by $1/\delta_{RG}$ the mean rejuvenation time, and by δ_{AG} and δ_{GA} the rates from state A to G and G to A, respectively. Finally, μ_S is the queue service rate.

We analyze the process described in Figure 1 in two regimes. First, in Section IV we consider the case wherein there are no anomalies (e.g., state A in the above figure is never reached). In this case, the main question relates to the mean time until a false alarm. The system transitions between states R and G, but an alarm may still be triggered when the queue grows beyond a certain value which causes the sequential performance analysis algorithm to trigger an alert.

Then, in Section V we consider the case where anomalies are present (e.g., all states of the above system are reachable). In this case, we aim at studying system unavailability due to anomalies and rejuvenation, assuming that both are causes of system downtime. In the above figure, states R and A lead to unavailability, and state G corresponds to an available system. Clearly, increasing the rejuvenation rate β causes the system to spend less time in the anomalous state, but at the cost of potential unavailability while being rejuvenated. One of our goals is to study the tradeoffs involved in the adjustment of rejuvenation, in light of natural recovery (captured through δ_{AG} and through the dynamics of the queue size while the source is at state G) and anomalies (captured through δ_{GA}).

B. Sequential performance analysis and the bucket algorithm

Throughput samples. Next, we briefly introduce the sequential performance analysis algorithm considered in this work, namely, the bucket algorithm. Every time a new sample is collected from the system, it is compared against a threshold. From now on, we assume that the samples correspond to (instantaneous) throughput measures collected from good packets, and that small values may be due to anomalies. From an analytical standpoint, the decrease in throughput of good packets in the presence of bad packets may be due to 1) the fact that at the bad state we have $\lambda_A > \mu_S$ in an open system as in Fig. 1 and [1]; 2) finite queue, i.e., $M/M/1/K$ as opposed to $M/M/1$ in Fig. 1; or 3) a closed system as in [15].

There are B buckets of maximum depth D each. The current bucket is referred to as b, and the current depth of the current bucket, i.e., the number of tokens in it, is d. To each bucket there is a corresponding threshold, whose value is a function of b. If a sample is smaller than the current threshold, a token is added to the current bucket (small throughput). Otherwise, a token is removed (normal throughput).

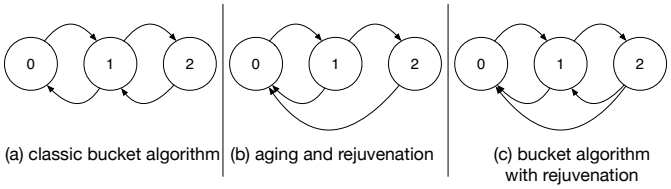


Figure 2. Bucket algorithm

If the current bucket overflows, the algorithm sets the next bucket as the current bucket. Correspondingly, if the current bucket underflows, i.e., if the number of tokens becomes negative, the algorithm sets the previous bucket as the current bucket, or remains unchanged if $b = 1$ and $d = 0$. *When $b = 1$ and $d = 0$, the system has recovered its original state. In contrast, at the extreme when all buckets overflow, $b = B$ and $d = D + 1$, an alarm is raised signaling an anomaly.*

Relation between system state and bucket state. Note that the system state is characterized by the state of the queue, i.e., an integer representing the number of packets in the queue, and the state of the MMPP process. The state of the bucket algorithm, in contrast, is simply the current bucket and the number of tokens in the bucket. The system state and the bucket state are closely related to each other: the number of tokens d should reflect larger queue sizes and/or a transition of the source to the anomalous state A. In case the source is still at the good state G, an increase in d corresponds to an increased probability of issuing a false alarm.

Bucket mechanics. For the purposes of this paper, we assume that system dynamics is captured through a single parameter, namely, the probability that the upcoming sample will cause a token to be added to the current bucket. We refer to this probability as the transient loss probability, or TLP for short. In addition, inspired by CUSUM, except otherwise noted we assume $B = 1$. We also assume that after rejuvenation the system recovers its original state and then $d = 0$.

C. From system to model

Next, we indicate how to parametrize an aging and rejuvenation model in light of the characterized system.

In Section IV, when $\delta_{GA} = \delta_{AG} = \lambda_A = 0$, and $\delta_{RG} = \infty$, the model comprises three parameters: the rate at which tokens are added to a bucket of the so-called bucket algorithm, the rate at which tokens are removed from the bucket and the rejuvenation rate. We refer to those three parameters as λ, μ and ξ. Indeed, the first two grow as a function of λ_G and μ_s, respectively. The third is a function of β. Our goal is to compute the time until the bucket of the bucket algorithm overflows, in the absence of anomalies, which corresponds to a false alarm.

In Section V we consider the case where anomalies are present. In this case, we can again parametrize the model from systems. In this scenario, we also leverage λ_A and δ_{AG} to set the model parameters, and account for the fraction of time at which the system is at states R and A to characterize unavailability.

Modeling working assumptions. As in any modeling exercise, we have to tradeoff between model complexity and expressive capacity. In our continuous time models, we make the following simplifying assumptions: 1) the times between events are all exponentially distributed, irrespectively of the model used to characterize the system, e.g., MMPP presented in Figure 1; 2) rejuvenation can occur from any system state, and except otherwise noted occurs with a constant rate independent of system state.

Summary: In this section we have indicated how the model parameters considered in this paper relate to a simple system. The discussion serves to illustrate how the model is applicable in a simple setting. We leave a methodology to produce a direct functional relationship between the system and model parameters as subject for future work.

IV. Baseline setup without anomalies: modeling false alarm rate in light of rejuvenation

In this section we consider sequential performance analysis of systems that age and rejuvenate, under the assumption that no anomalies will occur during the time horizon of interest. This corresponds to golden runs in [2].

As we assume that no anomaly occurs, alarms correspond to false alarms. We aim at answering the following question:

Question 1: *How long does it take for a false alarm to be triggered, when running sequential performance analysis, e.g., using the bucket algorithm, of a system that ages and rejuvenates?*

Without accounting for rejuvenation, the above question was answered in [2]. In what follows, we 1) indicate how rejuvenation increases the time until a false alarm and 2) show that results from the literature on birth-death processes with catastrophes are instrumental for the analysis of the bucket algorithm under rejuvenation.

A. Birth-death processes with catastrophes are instrumental to study sequential performance analysis with rejuvenation

Birth death processes with catastrophes have been studied under different setups, e.g., accounting for $M/M/1$ with catastrophes [3] and $M/M/1/K$ with catastrophes [4]. In what follows, we show that those results are instrumental for characterizing the bucket algorithm for sequential performance analysis under rejuvenation.

Let state k correspond to the alarm state. The time until a false alarm is the time to reach state k starting from state 0. Let $\overline{T}_{0,k}$ be the mean time to reach state k from state 0. Equation (15) in [16] characterizes the mean time to reach a given state in a birth-death process with catastrophes, from any given state. Interestingly, it follows from Equation (15) in [16] that

$$\overline{T}_{0,k} = \frac{1 - \hat{\gamma}_{0,k}(\xi)}{\xi \hat{\gamma}_{0,k}(\xi)} = \frac{1}{\xi \hat{\gamma}_{0,k}(\xi)} - \frac{1}{\xi} \tag{1}$$

where

$$\hat{\gamma}_{0,k}(\xi) = \prod_{i=0}^{k-1} \frac{\alpha_i}{\alpha_i + \xi} \tag{2}$$

and α_i is the i-th eigenvalue of the negative truncated infinitesimal generator matrix of the birth-death process without catastrophes [17], [18].

Actually, it follows from [18] that the distribution of the hit time is also a sum of exponentially distributed random variables, where the i-th random variable has rate equal to the i-th eigenvalue of the negative truncated infinitesimal generator matrix of the process with catastrophes (see also [19]).

B. Toy example: a three state model corresponding to bucket depth of one

Figure 2 shows the Markov chains corresponding to the original bucket algorithm under natural recovery, (Figure 2(a)), and to aging and rejuvenation (Figure 2(b)). The two are combined in Figure 2(c), where state 0 is the initial state, and states 1 and 2 represent stages of aging.

Let λ, μ and ξ be the rate of degradation, naturally recovery and rejuvenation, and $\rho = \lambda/\mu$.

Bucket algorithm: Figure 2(a). The steady state solution of the bucket algorithm is given by

$$\pi_0 = \frac{1}{1+\rho+\rho^2}, \quad \pi_1 = \pi_0\rho, \quad \pi_2 = \pi_0\rho^2 \tag{3}$$

and the mean time to reach state 2 from state 0 is $T^{(a)} = 1/\tilde{\pi}_2 - 1$ where

$$\frac{1}{\tilde{\pi}_2} = \frac{\mu}{\lambda^2} + \frac{2}{\lambda} + 1. \tag{4}$$

Aging and rejuvenation: Figure 2(b). The steady state solution is given by

$$\pi_0^{(b)} = \frac{1}{1+\frac{\lambda}{\lambda+\xi}+\frac{\lambda}{\lambda+\xi}\frac{\lambda}{\xi}}, \quad \pi_1^{(b)} = \pi_0^{(b)}\frac{\lambda}{\lambda+\xi}, \tag{5}$$

$$\pi_2^{(b)} = \pi_0^{(b)}\frac{\lambda}{\lambda+\xi}\frac{\lambda}{\xi} \tag{6}$$

and the mean time to reach state 2 from state 0 is $T^{(b)} = 1/\pi_2^{(b)} - 1$ where $\pi_2^{(b)}$ is given by (4) replacing μ by ξ.

Bucket algorithm with aging and rejuvenation: Figure 2(c). The steady state solution is

$$\pi_0^{(c)} = \pi_2^{(c)}\left(\left(\frac{\mu+\xi}{\lambda}\right)^2 + \frac{1}{\lambda}\right), \quad \pi_1^{(c)} = \pi_2^{(c)}\left(\frac{\mu+\xi}{\lambda}\right)$$

$$\pi_2^{(c)} = \frac{1}{1+\frac{\mu+\xi}{\lambda}+\frac{\xi}{\lambda}+\left(\frac{\mu+\xi}{\lambda}\right)^2} \tag{7}$$

and the mean time to reach state 2 from state 0 is

$$T^{(c)} = \frac{1}{\tilde{\pi}_2^{(c)}} - 1 \tag{8}$$

where $\tilde{\pi}_2^{(c)}$ is given by (4) replacing μ by $\mu+\xi$.

Note also that the negative truncated infinitesimal generator matrix of the birth-death process without catastrophes is given by

$$\tilde{Q} = \begin{pmatrix} \lambda & -\lambda \\ -\mu & \lambda+\mu \end{pmatrix} \tag{9}$$

whose eigenvalues are given by

$$\alpha_{1,2} = \lambda + \frac{\mu}{2} \pm \frac{\sqrt{\mu(4\lambda+\mu)}}{2} \tag{10}$$

Then, it follows from (1) and (8) that

$$T^{(c)} = \frac{(\alpha_1+\xi)(\alpha_2+\xi)}{\xi\alpha_1\alpha_2} - \frac{1}{\xi} = \frac{\mu+\xi}{\lambda^2} + \frac{2}{\lambda}. \tag{11}$$

Summary: in this section, we have considered a simple setup wherein the bucket algorithm is subject to rejuvenation, in the absence of anomalies (failures or attacks). Through a simple example, we have illustrated how results on birth-death processes with catastrophes are applicable for the analysis of rejuvenation. In what follows, we consider the discrete time model corresponding to the continuous time model considered in this section, and indicate how a parameterized geometric distribution can be instrumental to approximate the time until a false alarm.

C. Discrete time case

Up until now, we considered the mean time to false alarm measured for a continuous time process. A discrete time model was considered in [2]. Next, we illustrate how rejuvenation impacts the behavior of the bucket algorithm under the discrete time setting.

Question 2: *How does the solution to the continuous time model relates to that of a discrete time model?*

The mean number of transitions N until a false alarm relates to the mean continuous time T until a false alarm as

$$N_0\frac{1}{\tau_0} + N_{>0}\frac{1}{\tau_{>0}} = T \tag{12}$$

$$N_0 + N_{>0} = N \tag{13}$$

where N_0 is the mean number of visits to state 0, before absorption, and $N_{>0}$ is the mean number of visits to other states. We have two equations and three unknowns. To solve the above system of equations, we need to account for additional balance equations, as illustrated through our toy example.

1) Back to the toy example: Next, we revisit our toy example (Figure 2(c)) in light of the discrete time model. We have:

$$N_0\frac{1}{\lambda} + N_{>0}\frac{1}{\lambda+\mu+\xi} = \frac{\mu+\xi}{\lambda^2} + \frac{2}{\lambda} \tag{14}$$

$$N_0 + N_{>0} = N \tag{15}$$

$$1 + N_{>0}\frac{\xi+\mu}{\lambda+\mu+\xi} = N_0 \tag{16}$$

The first two equations above correspond to (12) and (13). The last equation is the balance equation for state 1.

It follows from the solution of the above set of equations that the number of samples until a false alarm under the toy example is given by

$$N = 2\left(\frac{\mu+\xi+\lambda}{\lambda}\right). \tag{17}$$

2) Numerical evaluation and geometric approximation: Next, we consider a simple model to approximate the number of samples until a false alarm, under the discrete time model.

Question 3: *Can we approximate the number of samples until a false alarm through a geometric distribution?*

To answer the above question, we consider a simple simulation of the system with aging and rejuvenation. Then, we indicate a regime wherein the geometric model accurately captures the distribution of the number of samples until a false alarm, and illustrate how to parametrize it for the purposes of estimating the mean number of samples until the false alarm is triggered.

Simulation setup. We developed a sequential performance analysis simulator that implements the logic of the Bucket Algorithm, applying rejuvenation (see Algorithm 1). We work with one bucket ($B = 1$), varying its depth from 1 to 28 ($1 \leq D \leq 28$). For the purposes of this section, we consider the environment free of anomalies, aiming at answering questions related to how long it takes for a false alarm to be triggered, varying the rejuvenation probability, R.

Recall that the Transient Loss Probability (TLP) refers to the probability of adding a token to the bucket, due to aging, and one minus TLP is the probability of removing a token. Motivated by [2], in our experiments we consider a TLP of 0.46 (this is the probability of adding tokens to the bucket parametrized based on real data from the `TPCx-V` benchmark). We then extend [2] by accounting for rejuvenation. The rejuvenation probability ranged from 1% to 10%.

Each iteration of Algorithm 1 corresponds to a new sample (e.g., of throughput) being collected. In line 2 of Algorithm 1, we sample a number between 0 and 1 uniformly at random, to determine whether rejuvenation will take place. If so, the bucket depth is reset to zero (line 4). Otherwise, we sample another number between 0 and 1 uniformly at random, to determine whether a token should be added or removed, corresponding to the cases where the sampled throughput is below or above the current threshold, respectively (lines 7-10). Note that for the purposes of the algorithm, the threshold is indirectly captured through the TLP denoted by L. Finally, when d surpasses D, an alarm is raised (line 14).

Exponential distribution against distribution of time to false alarm. To compare the behavior of the exponential distribution against time to false alarm as obtained by simulations, we consider the quantiles of the two distributions.

The samples of time until false alarm wer obtained using Algorithm 1. The exponential distribution was parametrized using the simulated data and least square error (LSE), using SciPy functions `expon.fit` and `expon.rvs`. The pdf of the exponential distribution is given by

$$f(x) = e^{-(x-l)/s}/s \tag{18}$$

where l and s are the location and the scale of the distribution, derived using LSE. Note that there is no direct physical meaning to the parameters of the exponential distribution in our domain of interest. We circumvent this shortcoming using the geometric distribution in the following section.

We let $D = 12$ and note that the exponential distribution closely approximates the behavior of the system (Figure 3). Figure 3(a) considers the case without rejuvenation, and Figure 3(b) with rejuvenation of 5%. Similar behavior was observed at other depths and with different rates of rejuvenation, as far as $D \leq 28$ and TLP is less than 0.5. We observed that for larger values of TLP and D (not shown in the figure), the error of the exponential distribution becomes more significant.

A geometric model to mean time to false alarm. We consider the following geometric model to approximate the mean time until a false alarm:

$$\widetilde{T} \approx \frac{\kappa_1}{(L \cdot (1-R))^{\kappa_2 \cdot D}} \tag{19}$$

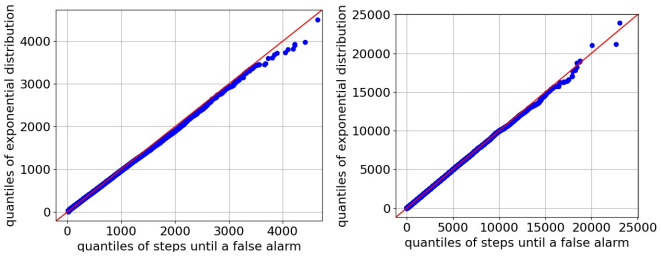
where κ_1 and κ_2 are constants parametrized to fit the model to the simulation data. The rationale behind the model consists in assuming that the average time until a false alarm can be roughly approximated by the average time until a sequence of D tokens are added to the bucket, where each token is added with probability $L \cdot (1-R)$, i.e., the probability that a transient loss probability occurs and no rejuvenation takes place.

In the scenario considered above, we have $L = 0.46$ and R varying between 0 and 10%. Table II shows the values of κ_1 and κ_2 that best fit our data, and Fig. 4 indicates that indeed a geometric model can capture the system behavior for $D > 3$.

Take away message. Figure 4 shows the impact of rejuvenation on the mean time until false alarm. As expected,

Algorithm 1 Simulation: Bucket Algorithm with rejuvenation

```
1:  while d ≤ D do (depth smaller than maximum depth)
2:      x1 ← U[0, 1] (uniformly at random let x1 ∈ [0, 1])
3:      if x1 < R (rejuvenation probability) then
4:          d ← 0
5:      else
6:          x2 ← U[0, 1] (uniformly at random let x2 ∈ [0, 1])
7:          if x2 < L (transient loss probability) then
8:              d ← d + 1 (add token to bucket, due to loss)
9:          else
10:             d ← d − 1 (remove token from bucket, due to success)
11:         end if
12:     end if
13: end while
14: Raise alarm
```



(a) without rejuvenation (b) with rejuvenation

Figure 3. Exponential distribution against time to false alarm: (a) quantiles of steps until a false alarm versus quantiles of geometric distribution with $D = 12$ and $R = 0$. (b) $D = 12$ and $R = 0.05$.

Table II
ANALYTICAL MODEL

Rejuvenation	κ_1	κ_2	Rejuvenation	κ_1	κ_2
1%	33.7894	0.3129	6%	20.0881	0.5092
2%	24.7297	0.3727	7%	9.33475	0.5664
3%	12.7911	0.4425	8%	13.9233	0.5709
4%	15.5557	0.4648	9%	9.96429	0.6068
5%	12.7366	0.5033	10%	14.1693	0.6097

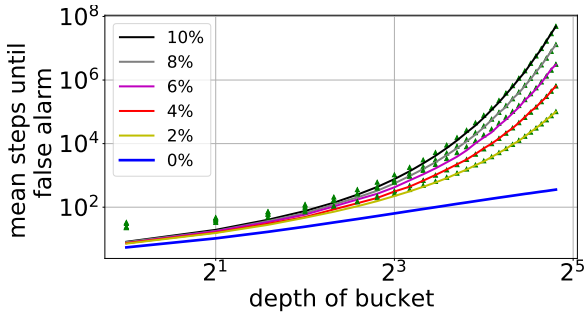


Figure 4. Rejuvenation helps to avoid false alarms: the figure shows an increase in mean number of steps until the bucket overflows (false alarm) and contrasts simulation against an analytical model that approximates the behavior of simulations through a geometric distribution.

the more aggressive is the rejuvenation, the greater is the time until a false alarm. Increasing the rejuvenation probability from 0% to 10%, for instance, when $D = 12$, we observe a four-fold increase in the mean number of samples until a false alarm. In the following section, we also consider the potential side effects of rejuvenation in terms of system unavailability while being rejuvenated.

V. ACCOUNTING FOR ANOMALIES

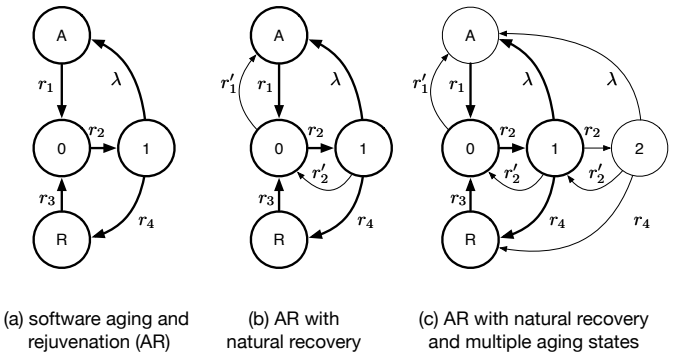
Next, we introduce a Markov model to capture the integrated impact of aging, natural recovery, rejuvenation and attacks on system behavior. In particular, we extend [5] (which corresponds to Figure 5(a)) to account for natural recovery and the possibility of anomalies at the original state (which corresponds to Figure 5(b)).

Question 4: *How to account for anomalies, e.g., due to attacks, while analyzing the role of aging, rejuvenation and natural recovery under sequential performance analysis?*

A. Model description

Our model is obtained from the continuous-time Markov chain proposed by [5], and illustrated in Figure 5(a). States 0, 1, A and R correspond to the initial state, an aging state that is prone to failure, the anomalous state where failure has occurred and the rejuvenation state, respectively. Aging occurs at rate r_2, and after reaching state 1 the anomaly and rejuvenation processes compete with each other, occurring with rates λ and r_4, respectively. Finally, once an anomaly (resp., rejuvenation) occurs, it takes on average $1/r_1$ (resp., $1/r_4$) to complete.

One of the key objectives is to analyze the impact of anomalies and rejuvenation of system unavailability. To that aim, it is



(a) software aging and rejuvenation (AR) (b) AR with natural recovery (c) AR with natural recovery and multiple aging states

Figure 5. Bucket algorithm with overhead due to rejuvenation and anomalies

assumed that system unavailability is given by $U = \pi_A + \pi_R$ where π_A and π_R are the steady state probabilities of states A and R.

First, we extend the above model in two ways, by 1) adding the possibility that the system naturally recovers from aging, which occurs with rate r'_2, and 2) allowing an anomaly to occur when the system is at state 0, which occurs with rate r'_1. The corresponding model is shown in Figure 5(b).

Second, we further extend the model by allowing multiple aging states, that capture the number of tokens in the bucket algorithm. Although there may not be a one-to-one mapping between number of tokens and aging state, we assume that the latter is a proxy to the former. In that case, the rate parameters are the same as those described above, but the number of states captures the multiple aging levels (Figure 5(c)).

B. Model solution

Next, we analytically solve the models in Figure 5(a) and Figure 5(b). The model in Figure 5(c) is numerically evaluated in the upcoming section, and compared against the first two.

The solution of the Markov model presented in Figure 5(b) is given by:

$$\pi_0^{-1} = 1 + \frac{r_2}{r'_2 + r_4 + \lambda}\left(1 + \frac{r_4}{r_3} + \frac{\lambda}{r_1}\right) + \frac{r'_1}{r_1} \tag{20}$$

$$\pi_1 = \pi_0 \frac{r_2}{r'_2 + r_4 + \lambda}, \quad \pi_R = \pi_0 \frac{r_2}{r'_2 + r_4 + \lambda}\frac{r_4}{r_3} \tag{21}$$

$$\pi_A = \pi_0 \left(\frac{r'_1}{r_1} + \frac{r_2}{r'_2 + r_4 + \lambda}\frac{\lambda}{r_1}\right) \tag{22}$$

Then, system unavailability is given by $\pi_A + \pi_R$,

$$U = \frac{\frac{r'_2+r_4+\lambda}{r_2}\frac{r'_1}{r_1} + \frac{r_4}{r_3} + \frac{\lambda}{r_1}}{\frac{r'_2+r_4+\lambda}{r_2}\left(1 + \frac{r'_1}{r_1}\right) + 1 + \frac{r_4}{r_3} + \frac{\lambda}{r_1}} \tag{23}$$

The derivative of U with respect to r_4 is

$$\frac{dU}{dr_4} = \kappa\left(r_1\left(1 + \frac{r'_2 + r_2}{\lambda}\right) - r_3\left(1 - \frac{r'_1}{\lambda}\right)\right) \tag{24}$$

where κ is a positive constant. The above equation reduces to

$$\frac{dU}{dr_4} = \kappa\left(r_1\left(1 + \frac{r_2}{\lambda}\right) - r_3\right) \tag{25}$$

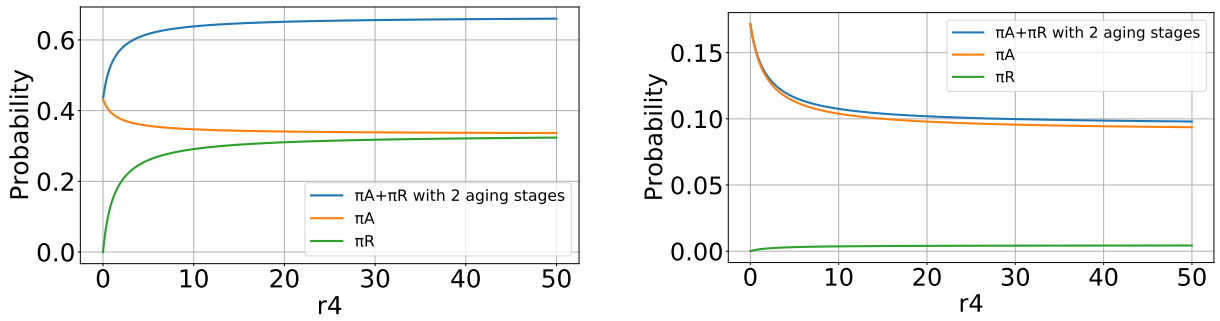


Figure 6. Illustrating a scenario wherein rejuvenation (a) is not beneficial and (b) one where it is beneficial. (a) $r_1 = 0.1, r_1' = 0.1, r_2 = 0.1, r_2' = 0.1, r_3 = 0.1, \lambda = 0.05$ and (b) decreasing r_1' and λ to 0.01 and 0.03 and increasing r_3 to 20. The results are in agreement and illustrate the applicability of (24).

when $r_2' = r_1' = 0$ that corresponds to Figure 5(a).

Note that if $dU/dr_4 < 0$ rejuvenation should be applied at maximum rate. Otherwise, rejuvenation should not be applied.

C. Discussion: comparing model with natural recovery against original aging and rejuvenation model

Next, we further discuss the impact of the different system parameters on the optimal rejuvenation rate r_4. According to (25), as r_1 or r_2 increase, i.e., as it becomes easier to resolve an anomaly or as the system becomes more prone to aging, rejuvenation becomes less attractive. Accordingly, as the rate at which anomalies occur, λ, increases, or the mean time to execute a rejuvenation, $1/r_3$, decreases, rejuvenation becomes more attractive.

To extend the above analysis for the case with natural recovery, we further assess the role of r_1' and r_2'. According to (24), r_2', which is the rate of natural recovery, plays an additive role with respect to the aging rate r_2 when determining the attractiveness of rejuvenation. Indeed, if natural recovery already resolves most of the aging effects, rejuvenation may not be needed. Finally, an increase in r_1' makes rejuvenation less attractive, as it increases the chance that a rejuvenation is immediately followed by a new anomaly.

Numerical evaluation. Next, we illustrate the points discussed in the previous paragraph, on the role of different system parameters on the optimal rejuvenation rate. To that aim, we numerically evaluate the Markov chain in Figure 5(c) with $r_1 = 0.1, r_1' = 0.1, r_2 = 0.1, r_2' = 0.1, r_3 = 0.1$ and $\lambda = 0.05$. For those parameters, we observe in Figure 6(a) that U increases with r_4 in agreement with (24). Then, we increase r_3 to 20 and reduce r_1' and λ to 0.01 and 0.03. All those changes favor the benefits of rejuvenation, as indicated in Figure 6(b), again in agreement with (24).

In the scenarios of Figure 6 it turned out that using one or two aging stages had a negligible impact on the probabilities of interest. In what follows, we illustrate a scenario wherein the number of aging stages impacts our metrics of interest.

Accounting for multiple aging stages. To illustrate the impact of the number of aging stages, let $r_1 = 0.1, r_1' = 0.01, r_2 = 4, r_2' = 2, r_3 = 20$, and $\lambda = 0.05$. This corresponds to the latter scenario as baseline, and increasing the aging, r_2, and natural recovery rate, r_2'. Figure 7 shows the system unavailability, as a function of r_4. With 2 aging stages the unavailability for $r_4 = 0$ is greater than for 1 aging stage: the rate of failures at states different from 0 is greater than at 0, and the system with 1 aging stage visits state 0 more often than the system with 2 aging stages. As r_4 grows the unavailability converges to an asymptotic value that does not depend on the number of stages, given by $3/13 \approx 0.23$ (see (23)).

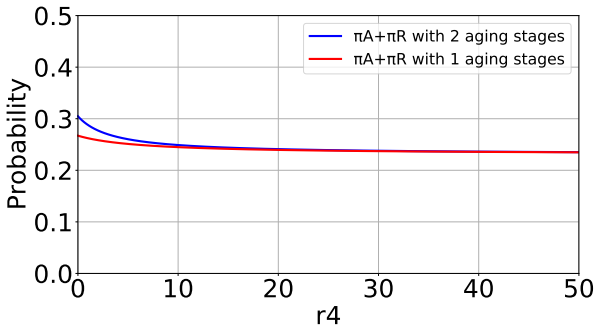


Figure 7. Impact of the number of aging states.

VI. Conclusion

In this work we have extended aging and rejuvenation models to account for natural recovery. In the realm of sequential performance analysis, in the absence of anomalies we have computed the mean time until a false alarm. In the presence of anomalies, we have computed system unavailability accounting for aging, natural recovery and rejuvenation.

As future work, we plan to apply the methodology proposed in this work to a real case study, as well as to further analyze the MMPP workload in light of aging, rejuvenation and natural recovery. We also aim at studying networked nodes subject to aging, rejuvenation, and natural recovery, accounting for a cooperative multithread environment [20], [21] in a scalable fashion and extending product form networks for that matter [22]–[24].

Acknowledgment: This research is co-financed by CAPES, CNPq, and FAPERJ under Grants E-26/203.215/2017, E-26/211.144/2019, and E-26/201.376/2021.

REFERENCES

[1] Y. Burkatovskaya, T. Kabanova, and O. Tokareva, "Sign CUSUM Algorithm for Change-Point Detection of the MMPP Controlling Chain State," in *International Conference on Information Technologies and Mathematical Modelling*, pp. 18–33, Springer, 2016.

[2] C. F. Gonçalves, D. S. Menasché, A. Avritzer, N. Antunes, and M. Vieira, "A model-based approach to anomaly detection trading detection time and false alarm rate," in *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 1–8, IEEE, 2020.

[3] B. K. Kumar and D. Arivudainambi, "Transient solution of an m/m/1 queue with catastrophes," *Computers & Mathematics with applications*, vol. 40, no. 10-11, pp. 1233–1240, 2000.

[4] W. Böhm, "A note on queueing systems exposed to disasters," 2008.

[5] 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*, pp. 381–390, IEEE, 1995.

[6] M. Grottke, A. Avritzer, D. S. Menasché, L. P. de Aguiar, and E. Altman, "On the efficiency of sampling and countermeasures to critical-infrastructure-targeted malware campaigns," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 4, pp. 33–42, 2016.

[7] J.-M. Fourneau and Y. A. El Majhoub, "Processor sharing g-queues with inert customers and catastrophes: A model for server aging and rejuvenation," *Probability in the Engineering and Informational Sciences*, vol. 31, no. 4, pp. 420–435, 2017.

[8] J. M. Meylahn, S. Sabhapandit, and H. Touchette, "Large deviations for markov processes with resetting," *Physical Review E*, vol. 92, no. 6, p. 062148, 2015.

[9] E. Altman, A. Avritzer, R. El-Azouzi, D. S. Menasche, and L. P. de Aguiar, "Rejuvenation and the spread of epidemics in general topologies," in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pp. 414–419, IEEE, 2014.

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

[11] A. Avritzer, R. Tanikella, K. James, R. G. Cole, and E. Weyuker, "Monitoring for security intrusion using performance signatures," in *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pp. 93–104, 2010.

[12] R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Design of software rejuvenation for cps security using invariant sets," in *2019 American Control Conference (ACC)*, pp. 3740–3745, IEEE, 2019.

[13] S. K. Acharya, C. E. Villarreal-Rodríguez, *et al.*, "Change point estimation of service rate in an M/M/1/m queue," *Int. J. Math. Oper. Res.*, vol. 5, no. 1, pp. 110–120, 2013.

[14] S. K. Singh and S. K. Acharya, "A Bayesian inference to estimate change point for traffic intensity in M/M/1 queueing model," *OPSEARCH*, vol. 59, no. 1, pp. 166–206, 2022.

[15] A. Marin, M. Meo, M. Sereno, and M. A. Marsan, "Modeling service mixes in access links: Product form and oscillations," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 312–318, IEEE, 2022.

[16] A. Di Crescenzo, V. Giorno, A. Nobile, and L. Ricciardi, "On the first-visit-time problem for birth and death processes with catastrophes," *arXiv preprint math/0307206*, 2003.

[17] J. A. Fill, "The passage time distribution for a birth-and-death chain: Strong stationary duality gives a first stochastic proof," *Journal of Theoretical Probability*, vol. 22, no. 3, pp. 543–557, 2009.

[18] M. Brown and Y.-S. Shao, "Identifying coefficients in the spectral representation for first passage time distributions," *Probability in the Engineering and Informational Sciences*, vol. 1, no. 1, pp. 69–74, 1987.

[19] O. Jouini and Y. Dallery, "Moments of first passage times in general birth–death processes," *Mathematical Methods of Operations Research*, vol. 68, no. 1, pp. 49–76, 2008.

[20] S. HomChaudhuri, "Implementation of micro-rejuvenation with cooperative stack leasing," in *Handbook Of Software Aging And Rejuvenation: Fundamentals, Methods, Applications, And Future Directions*, pp. 327–351, World Scientific, 2020.

[21] V. Sundaram, S. HomChaudhuri, S. Garg, C. Kintala, and S. Bagchi, "Improving dependability using shared supplementary memory and opportunistic micro rejuvenation in multi-tasking embedded systems," in *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, pp. 240–247, IEEE, 2007.

[22] S. Balsamo, P. G. Harrison, and A. Marin, "A unifying approach to product-forms in networks with finite capacity constraints," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 25–36, 2010.

[23] X. Chao, "A queueing network model with catastrophes and product form solution," *Operations Research Letters*, vol. 18, no. 2, pp. 75–79, 1995.

[24] J. R. Artalejo, "G-networks: A versatile approach for work removal in queueing networks," *European Journal of Operational Research*, vol. 126, no. 2, pp. 233–249, 2000.