A Study of Systems with Multiple Operating Levels, Soft Thresholds and Hysteresis

Alexandre Brandwajn, Thomas Begin, Hind Castel-Taleb, Tulin Atmaca

A Study of Systems with Multiple Operating Levels, Soft Thresholds and Hysteresis

Alexandre BRANDWAJN, Thomas BEGIN, Hind CASTEL, Tulin ATMACA

RESEARCH REPORT

N° 9064

27/04/2017

Project-Team DANTE

ISSN 0249-6399

A Study of Systems with Multiple Operating Levels, Soft Thresholds and Hysteresis

Alexandre Brandwajn[1], Thomas Begin[2], Hind Castel[3] and Tulin Atmaca[3]

Project-Team DANTE

Research Report N° 9064 — 27/04/2017 —13 pages.

Abstract: Current architecture of many computer systems relies on dynamic allocation of a pool of resources according to workload conditions to meet specific performance objectives while minimizing cost (e.g., energy or billing). In such systems, different levels of operation may be defined, and switching between operating levels occurs at certain thresholds of system congestion. To avoid rapid oscillations between levels of service, "hysteresis" is introduced by using different thresholds for increasing and decreasing workload levels, respectively.
We propose a model of such systems with non-Poisson arrivals, arbitrary number of servers and operating levels where each operating level may correspond to an arbitrary number of additional servers and soft (i.e. non-deterministic) thresholds to account for "inertia" in switching between operating levels. Additionally, in our model server processing rates may be a function of the current operating level and of the number of requests (users) in the system. We also allow for delays in the activation of additional operating levels. We use simple mathematics to obtain a semi-numerical solution of our model. We illustrate the versatility of our model using several case study examples inspired by features of real systems. In particular, we explore optimal thresholds as a tradeoff between performance and energy consumption.

Key-words: Multi-server systems, multiple operating levels, hysteresis, soft thresholds, non-Poisson arrivals, activation delays.

[1] University of California Santa Cruz, Baskin School of Engineering, USA

[2] Université Lyon 1 / LIP (UMR INRIA, ENS Lyon CNRS, UCBL), Lyon , France

[3] Institut Telecom, Telecom SudParis, Evry, France

RESEARCH CENTRE
GRENOBLE - RHÔNE-ALPES

Inovallée
655 avenue de l'Europe - Montbonnot
38334 Saint Ismier Cedex France

Une étude des systèmes avec plusieurs niveaux de fonctionnement, des seuils non déterministes et avec hystérésis

Résumé : L'architecture actuelle de nombreux systèmes informatiques repose sur l'allocation dynamique d'un ensemble de ressources selon les conditions de charge pour atteindre des objectifs de performance spécifiques tout en réduisant les coûts (par exemple, l'énergie ou la facturation). Dans de tels systèmes, différents niveaux de fonctionnement peuvent être définis et la commutation entre ces niveaux de fonctionnement se produit à certains seuils de congestion du système. Pour éviter les oscillations rapides entre deux niveaux de service, un mécanisme d'«hystérésis» est introduit en utilisant des seuils différents entre deux niveaux de fonctionnement selon que le niveau de la charge augmente ou diminue.
Nous proposons un modèle de ces systèmes avec des arrivées non Poisson, un nombre arbitraire de serveurs et de niveaux de fonctionnement où chaque niveau de fonctionnement peut correspondre à un nombre arbitraire de serveurs supplémentaires et avec des seuils non déterministes pour tenir compte de "l'inertie" lors de la commutation entre deux niveaux de fonctionnement. En outre, dans notre modèle, les taux de service des serveurs peuvent être fonction du niveau de fonctionnement actuel et du nombre de requêtes (utilisateurs) dans le système. Nous permettons également des délais dans l'activation de niveaux de fonctionnement supplémentaires. À l'aide de raisonnements mathématiques simples, nous obtenons une solution semi-numérique de notre modèle. Nous illustrons la polyvalence de notre modèle en utilisant plusieurs exemples d'étude de cas inspirés des caractéristiques de systèmes réels. En particulier, nous étudions les valeurs optimales des seuils afin de trouver le meilleur compromis entre performances et consommation d'énergie.

Mots clés : Systèmes multi-serveurs, niveaux de fonctionnement multiples, hysterésis, seuils non-déterministes, arrivées non-Poisson, délais d'activation.

CURRENT architecture of computer systems and services tends to rely on dynamic allocation of a pool of resources (such as Virtual Machines, processors, storage, etc.) under varying workload conditions in order to meet specific performance objectives while at the same time minimizing cost (e.g., in terms of energy or billing). Examples range from cloud computing [FOX09] and virtualization environments such as VMware [VMW17] to enterprise Operating Systems such as IBMs AIX [AIX17] or Virtual Network Switches in the context of future generation Network Function Virtualization (NFV) [HAW14].

To adapt the number of discrete resources to dynamically varying workloads, different levels of operation are defined and switching between operating levels occurs at certain thresholds. These thresholds may correspond to resource utilization or some other measure of system congestion. To avoid overreacting to spurious workload changes, some "inertia" is introduced through the use of averaging such as sliding window or exponential smoothing, and to avoid rapid oscillations between levels of service, "hysteresis" is introduced by defining different thresholds for increasing and decreasing workload levels respectively.

To our knowledge, the bulk of theoretical analysis of systems with multiple operating levels in the literature is limited to the case where each increase (respectively, decrease) in operating level corresponds to adding (respectively, removing) a single server, there is no inertia in thresholds (deterministic instantaneous thresholds), and request arrivals come from a Poisson source (single or bulk).

Ibe and Keilson [IBE93] derived a closed-form solution for the steady-state distribution of the number of requests in the system through the use of Greens function under the assumption of Poisson arrivals and hard deterministic thresholds where additional servers are allocated immediately as specific values of the number of requests are exceeded. Their numerical results are limited to 3 servers. In 1997, Golubchik and Lui [GOL97] used a combination of the stochastic complementation and matrix geometric methods to derive upper and lower bounds on the performance of such systems with hard deterministic thresholds, Poisson arrivals, and single server allocation per operating level change. Their work accounts for a possible delay in the activation of each operating level. The numerical results in their paper are limited to 5 servers. A couple of years later, the same authors [LUI99] used the stochastic complementation method to obtain an exact solution for homogenous and heterogeneous servers with single and bulk Poisson arrivals. Numerical results in this paper are again limited to 5 servers. In 2000, Le Ny and Tuffin [LEN02] proposed an exact solution for the case of heterogeneous servers with Poisson arrivals. Their cuts method uses simpler mathematics than previous work in the literature to obtain the steady-state distribution of the number of requests in the system. No numerical examples are presented in their paper.

More recently, Mitrani [MIT11, MIT13] considered a model of a system with Poisson request arrivals and two blocks of servers where the reserve block is activated and deactivated according to forward and backward thresholds. The paper focuses on the selection of thresholds and the number of servers in the reserve block so as to optimize a cost function integrating system performance and energy consumption. Aït-Salaht and Castel-Taleb [AIT15a, AIT15b] consider a model of a node in a cloud system with hysteresis where virtual machines are added or removed one at a time. They use stochastic bounding to derive approximate steady-state probabilities of the number of requests in such a node with Poisson arrivals single and bulk.

Our contribution is to propose a model of systems with hysteresis, non-Poisson arrivals, arbitrary number of servers and operating levels where each operating level may correspond to an arbitrary number of additional servers and soft (i.e., non-deterministic) thresholds to account for "inertia" in switching between operating levels. Additionally, unlike in previous work, to account for potential speed degradation as the number of processors increases, in our model request completion rates with multiple servers are not necessarily multiples of single server rates. We also allow for delays in the activation of additional operating levels. We use simple mathematics to obtain a semi-numerical solution of our model.

In the next section we describe in more detail the model considered and we outline the proposed solution approach. Section 3 is devoted to several case studies, which illustrate possible applications of our model. Section 4 concludes this paper.

2 MODEL AND SOLUTION OUTLINE

2.1 Model description

We consider a system with two possible types of request arrival processes (see Figure 1). In the first type, the times between consecutive requests are assumed to be memoryless [ALL90] with a state-dependent rate $\lambda(n)$ when the current number of requests in the system is n. In addition to including the standard Poisson arrivals, such a quasi-Poisson arrival process can be a good representation of arrivals generated by a set of discrete request sources. In the second type of arrivals considered, times between consecutive arrivals are assumed to be independent but can have a general distribution. Specifically, times between request arrivals are distributed according to a phase-type distribution [HAR13] with a exponential phases. We denote by τ_j the probability that the arrival process starts in phase j ($j = 1, \ldots, a$), by λ_j the intensity of phase j and by $\hat{r}_j$ the probability that the arrival process completes after phase j. r_{ij} denotes the probability that the arrival process continues to phase j upon completion of phase k. Such phase-type distributions can represent arbitrarily closely any distribution [JOH88] and readily available methods exist to map theoretical or empirical distributions onto them [BOB05, OSO06].

As shown in Figure 1, the system has a maximum of C homogeneous servers and can accommodate a maximum of N requests (system capacity). Although the capacity of any practical system is finite, we treat also the case where there is no limitation on the number of requests. The system operates at L different levels, each level $\ell = 1, \ldots, L$ corresponds to a given number of active processors c_ℓ with $c_L = C$. We denote by $\mu(n, \ell)$ the rate of request completions when the current number of requests is n and the current operating level is ℓ. This allows us to account for example for the fact that the service rate of each processor may degrade as the

number of active processors increases due to inter-processor interference in multiprocessor environments. We assume that the activation of additional servers corresponding to the next system operating level is not necessarily instantaneous. We denote by $1/\omega$ the mean time to activate a new operating level and we assume that the level activation time is exponentially distributed. Thus, at any time the system could be operating at level ℓ and there could be $k = 0, \ldots, L-\ell$ levels whose activation is pending. For our purposes, we assume that level deactivation is instantaneous.

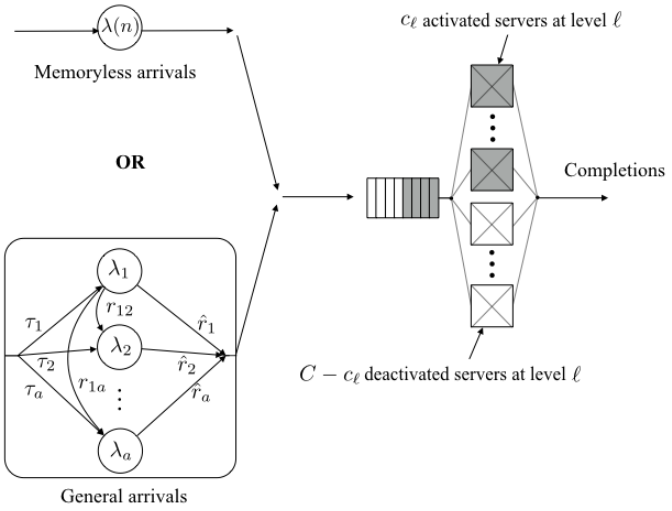


Fig. 1: System with a total of C servers and L operating levels.

The current state of our system (but not that of the arrival process in the case of general arrivals) is defined by the triple (n, ℓ, k) where $n = 0, \ldots, N$ is the number of requests in the system, $\ell = 1, \ldots, L$ is the current operating level and $k = 0, \ldots, L - \ell$ is the number of levels whose activation is pending. With each state (n, ℓ, k) (except those for which $\ell + k = L$ or $n = N$) we associate a probability denoted by $\gamma(n, \ell, k)$ that the activation of an additional operating level will be requested if the current number of requests increases from n to $n + 1$. Similarly, with each state except those for which $\ell = 1$ or $n = 0$ we associate a probability denoted by $\delta(n, \ell, k)$ that the deactivation of an operating level will happen if the current number of requests decreases from n to $n - 1$. Note that we assume that deactivation requests are applied to pending levels (if any) first. If the values of these probabilities are simply 0 or 1, the above description defaults to classical "hard" forward and backward thresholds. The use of other values corresponds to "soft" thresholds allowing one to model "inertia" in activating and deactivating operating levels. Note in passing that such "soft" thresholds are quite common beyond computer applications, e.g., in supermarket policies for adding checkout clerks based on the length of the queue. The performance metrics of interest in our model include the mean response time for a request, attained request throughput, as well as the mean number of active processors, and the fraction of time the system operates at each level. With a finite system capacity, the loss probability may also be of interest. Principal notation used in this paper is summarized in the Appendix.

2.2 Outline of model solution with memoryless arrivals

We start by considering the case of memoryless arrivals with rate $\lambda(n)$ for which the triple (n, ℓ, k) is sufficient to describe the state of our system. We let $p(n, \ell, k)$ be the steady-state probability of the system being in state (n, ℓ, k). It is easy to derive the corresponding balance equations (cf. Appendix). Denote by $p(n)$ the marginal steady-state probability that there are n requests in the system and by $p(\ell, k|n)$ the conditional probability that the current operating level is ℓ and there are k levels pending given n. We then have

$$p(n, \ell, k) = p(\ell, k|n)p(n) \tag{1}$$

The conditional rate of request completions given n, denoted by $u(n)$, can be written as

$$u(n) = \sum_{\ell=1}^{L} \mu(n, \ell) \left(\sum_{k=0}^{L-\ell} p(\ell, k|n) \right) \tag{2}$$

and the probability $p(n)$ can be expressed as

$$p(n) = \frac{1}{G} \prod_{i=1}^{n} \frac{\lambda(i-1)}{u(i)}, \quad n = 0, 1, \ldots \tag{3}$$

In (3) empty products are assumed to be equal to 1 and G is a normalizing constant such that $\sum_{n=0}^{N} p(n) = 1$. Using equations (1) and (3) in the balance equations for $p(n, \ell, k)$, we readily obtain a set of equations for the conditional probabilities $p(\ell, k|n)$. We then use simple fixed-point iteration to solve this set of equations numerically. We refer the interested reader to the Appendix for more detail about the proposed solution approach and its advantages. Having obtained the $p(\ell, k|n)$ and hence the equivalent completion rates $u(n)$ from (2), we compute the steady-state probabilities $p(n)$ using (3). From here, it is a straightforward matter to compute the mean number of requests in the system as $\bar{n} = \sum_{n=1}^{N} np(n)$, the attained request throughput as $\theta = \sum_{n=1}^{N} p(n)u(n)$ and the mean request response time as $R = \bar{n}/\theta$. The fraction of time the system spends on each operating level can be computed as $p(\ell) = \sum_{n=0}^{N} p(n) \left(\sum_{k=0}^{L-\ell} p(\ell, k|n) \right)$ and the mean number of busy servers is given by $\bar{c} = \sum_{\ell=1}^{L} p(\ell)c_\ell$. The state probabilities seen by an arriving request are given by $P_A(n) = \frac{\lambda(n)p(n)}{\sum_{i=0}^{N} \lambda(i)p(i)}$ so that, when the system capacity is finite, the loss probability can be obtained as $P_A(N)$.

2.3 Outline of model solution with general arrivals

We now consider the case where the times between arrivals have a general distribution (phase-type, independent and identically distributed). With such phase-type arrivals, the state description needs to be extended to include the current phase of the arrival process, j, $j = 1, \ldots, a$. We can then define the steady-state probability of the new full system state $p(n, \ell, k, j)$ where n, ℓ and k have the same meaning as before. One can derive balance equations for $p(n, \ell, k, j)$, transform them into equations for the conditional probability $p(\ell, k, j|n)$ and solve the latter using fixed-point iteration in a way quite analogous to the one described before for memoryless arrivals.

As an alternative to this exact solution, we propose an even simpler "divide and conquer" approach, which has the

added advantage of using previously developed solutions. We replace our model with state description $p(n, \ell, k, j)$ by two simpler models. In the first one, phase-type arrivals are replaced by state-dependent memoryless arrival rate $\alpha(n)$ while all other system aspects are fully represented. In effect, this model is the same as the one considered before for memoryless arrivals with state description (n, ℓ, k). In the second model, we use the state description (n, j) to fully represent the phase-type times between arrivals but we replace the remainder of the system by the equivalent conditional rate of request completions $u(n)$ obtained from the solution of our first model. In effect, our second model is a simple *Ph/M/C* type of queue, which can be easily solved using a simple numerically stable recurrence [BRA12]. The solution of this model produces the steady-state probability $p(n, j)$ for the state description (n, j) and the equivalent state-dependent rate of request arrivals $\alpha(n) = \frac{\sum_{j=1}^{a} \lambda_j \hat{r}_j p(n,j)}{\sum_{j=1}^{a} p(n,j)}$. Since we need the $\alpha(n)$ from our second model in order to compute the $u(n)$ from our first model, needed in our second model, we naturally end up with a fixed-point iteration between our two models as shown in Figure 2. We end our iteration when the mean numbers of requests in the system computed from both models are sufficiently close.

Algorithm 1 summarizes this fixed-point iteration between our two simpler models.

Algorithm 1 Solution of model with general arrivals via iteration between two simpler models

1: Initialize the arrival rate values $\alpha(n)$ to the inverse of the mean time between arrivals for all $n = 0, \ldots, N$.
2: Solve the model with state-dependent memoryless arrivals using the current values of $\alpha(n)$.
   a: Obtain current values for $p(\ell, k|n)$ and $p(n)$, as well as the equivalent service rate $u(n)$.
   b: Compute current value of $\bar{n}$ from this model.
3: Solve the *Ph/M/C* queue (our second model) using the current values of $u(n)$ rate from Step 2 as service rates.
   a: Obtain current values for $p(n, j)$ and for $\alpha(n)$.
   b: Compute the current value of $\bar{n}$ from this model ($\bar{n} = \sum_{n=1}^{N} n \left(\sum_{j=1}^{a} p(n, j) \right)$).
4: If the values of $\bar{n}$ from Step 2 and Step 3 deviate by less than $\epsilon > 0$ then stop the iteration, otherwise go to Step 2.
5: Use the values of $p(\ell, k|n)$ and $p(n)$ from last execution of Step 2 as the solution of the model.

Here, the state probabilities viewed by an arriving request are given by $P_A(n) = \frac{\alpha(n)p(n)}{\sum_{i=0}^{N} \alpha(i)p(i)}$. Note that strictly speaking, for our first model to be exact, we would need a rate of arrivals $\alpha(n, \ell, k)$. We introduce an approximation by assuming that the equivalent conditional rate of arrivals depends only on the current number of requests in the system n but not on the current system operating level or the number of pending levels. Similarly, strictly speaking we would need an equivalent service rate $u(n, j)$ for our second model to be exact. Again, we introduce an approximation by using a service rate that depends only on the current number of requests but not on the current phase of the arrival process. To summarize, the approximations introduced by our "divide and conquer" approach are $\alpha(n, \ell, k) \simeq \alpha(n)$ and $u(n, j) \simeq u(n)$. It has been our experience that the inaccuracy introduced by this type of approximation is generally quite small (cf. [ATM16]) and the number of iterations between models required for convergence is typically below 10.

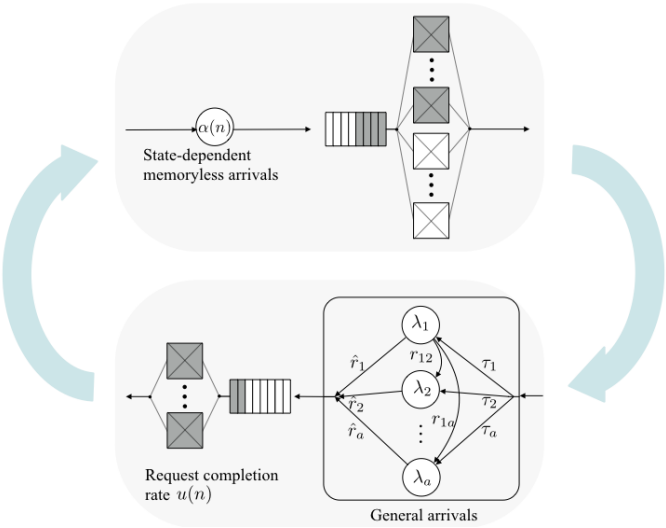


Fig. 2: Iteration between two simpler models in the case of general arrivals.

In the next section, we illustrate the versatility of our model using several examples derived from existing computer and network systems.

3 Case Studies

3.1 Example I: AIX SMT-like system

In our first example we consider a system inspired by the Simultaneous Multithreading feature in IBMs AIX Operating System [SMT17]. We assume that the system can operate at 3 levels where the corresponding numbers of logical processors are 2, 4 and 8. We take the activation delay for levels to be negligible ($1/\omega \simeq 0$). To represent possible processor interference, we assume that the service rate of each processor degrades as the operating level increases. Specifically, we assume that the service rate of each processor degrades by a factor of 0.95 for each consecutive higher operating level, i.e. as the number of processors doubles. The base mean service time with a single processor is taken to be 1. Our system functions with "soft" thresholds chosen so as to switch to the next higher operating level when the relative request response time is around 10 and switch back to the preceding level when the relative response time is around 7 (cf. Appendix). We define the mean relative response time as the ratio of the mean request response time to the mean service time with a single active processor. Arrivals are assumed to come from a set of $K = 256$ exponential request sources yielding a state-dependent arrival rate given by $\lambda(n) = (K - n)\phi$ where ϕ is the unitary request rate of an active source. The system capacity N is taken to be greater or equal to the number of request sources so that there are no lost requests. For this example, we use directly the solution described for memoryless arrivals in Section 2.2.

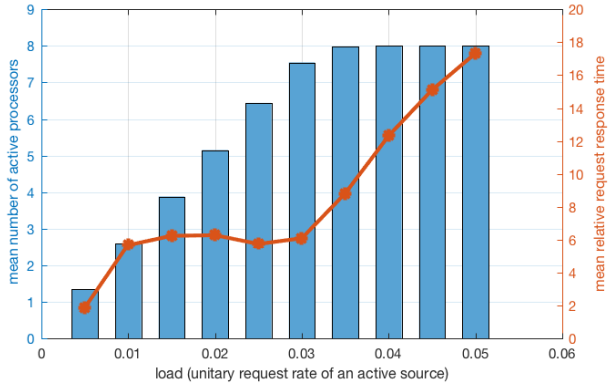


Fig. 3: Example I: mean relative request response time (right y-axis) and mean number of active processors (left y-axis) as a function of load.

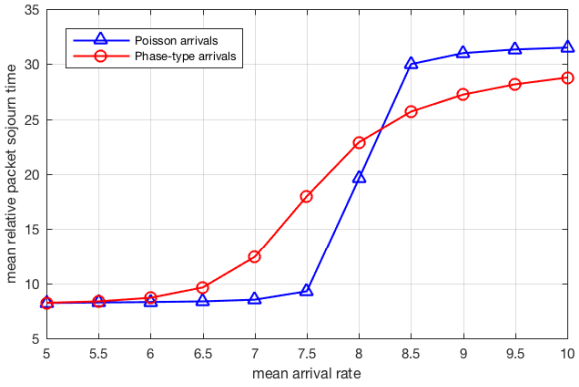


Fig. 4: Example II: mean relative packet sojourn time as a function of load for non-Poisson and Poisson arrival processes.

Figure 3 shows the mean relative response time and the mean number of active processors for this example as a function of the unitary request rate of an active source ϕ. We observe that the response time exhibits a characteristic "dip" and inflexion points as the system switches between its operating levels, while the mean number of processors increases with system load and then reaches its maximum value.

3.2 Example II: Virtual Switch-like system

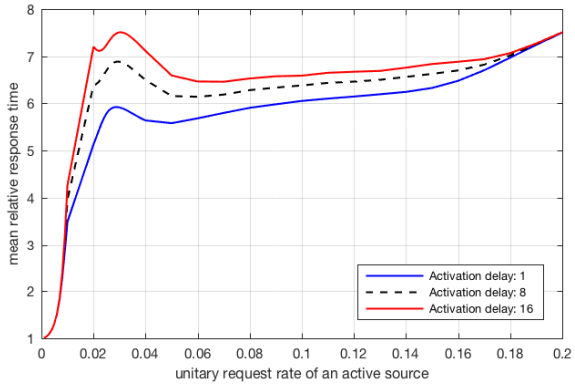
In our second example, we consider a system inspired by Virtual Switching systems (vSwitches) in Network Function Virtualization environments [HAW14]. A class of such vSwitches [EGI13] can dynamically enable processor cores to respond to varying packet workloads. Here, we assume that there are $C = 8$ processors and $L = 8$ operating levels with $c_\ell = \ell$, $\ell = 1, \ldots, L$ i.e. a single processor core is added (respectively, removed) when switching to the next higher (respectively, lower) level. In this example, we use "hard" forward and backward thresholds with negligible level activation delay. Since it is well known that packet arrivals processes in computer networks tend to deviate significantly from a Poisson process [PAX95], the times between request arrivals are given by a phase-type distribution with $a = 16$ phases and a coefficient of variation close to 15 (cf. Appendix). The system capacity is $N = 256$ requests (packets). The mean service time to process a packet is taken to be 1.

Figure 4 illustrates the results obtained from our model. It shows the mean packet sojourn time (request response time) as a function of the mean rate of packet arrivals. For comparison, we have included the results obtained from our model in the case of a Poisson arrival stream. We observe, as could be expected, that the influence of the arrival process on the mean packet sojourn time is most visible at medium loads for which assuming Poisson arrivals would result in a significant underestimation of the mean packet sojourn time. Interestingly, for high load values the mean sojourn time with our phase-type distribution of time between arrivals can actually become lower than with Poisson arrivals. An examination of other performance metrics indicates that this is due to higher loss probabilities than with Poisson arrivals (and thus lower attained throughput, i.e., carried traffic). It is worthwhile noting that, with the threshold values used, there appears to be no visible "dips" in the mean sojourn time.

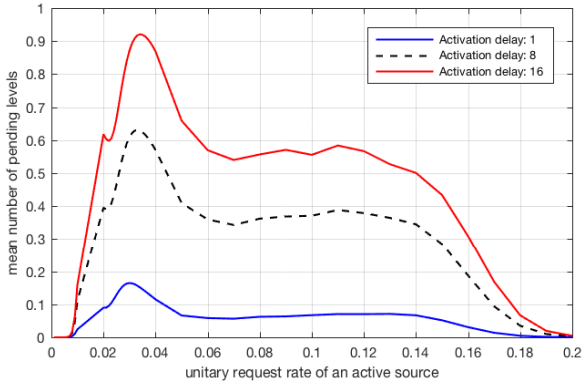
3.3 Example III: cloud-computing-like system

In our third example, we consider a high-level model of a cloud-computing platform in which Virtual Machines (VMs) are added and removed dynamically in response to workload variations (e.g., auto-scaling feature in Amazon Elastic Compute Cloud (EC2)). Here, we assume that there is a sizeable delay when activating additional VMs, and we consider "hard" thresholds to activate (respectively, deactivate) VMs. We assume that request arrivals come from a set of $K = 200$ exponential request sources resulting in a state-dependent arrival rate $\lambda(n) = (K-n)\phi$ where ϕ is the unitary request rate of an active source. The system capacity N is greater than the number of request sources so that there are no lost requests. We assume that the number of VMs varies in the range 2 to 16 ($C = 16$) and VMs are added and removed in groups of 2, i.e., $c_\ell = 2\ell$, $\ell = 1, \ldots, 8$. We fix the forward thresholds so as to switch to the next higher operating level when the relative request response time is around 7 and switch back to the preceding level when the relative response time is around 4. Here, the mean relative response time is defined as the ratio R/T where T is the mean request execution time.

For our numerical study, we take $T = 1$ and we explore the mean relative response time as a function of the unitary request rate ϕ. Figure 5a illustrates the results obtained for three values of the mean activation delay $1/\omega$ =1, 8 and 16. We note the presence of several "dips" and inflexion points corresponding to the VMs being activated and deactivated in the cloud. We observe that, with the parameter values considered, the effect of the activation delay appears most visible near the value of the unitary request rate $\phi = 0.03$. As shown in Figure 5b, it is near this value that the mean number of VMs whose activation is pending is the highest. For much lower loads, the system operates almost exclusively at level 1 and for much higher loads, the system



(a) Mean relative response time as a function of load and activation delay.



(b) Mean number of pending operating levels as a function of load and activation delay.

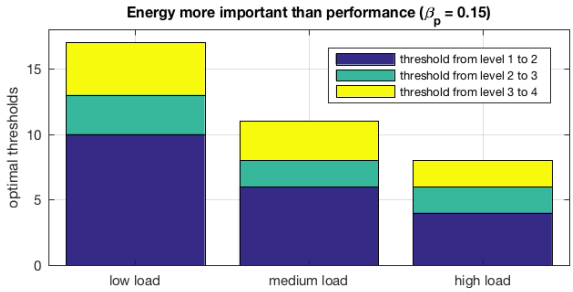
Fig. 5: Example III.

operates almost exclusively at its highest level, so that the value of the activation delay matters little in these operating regions.

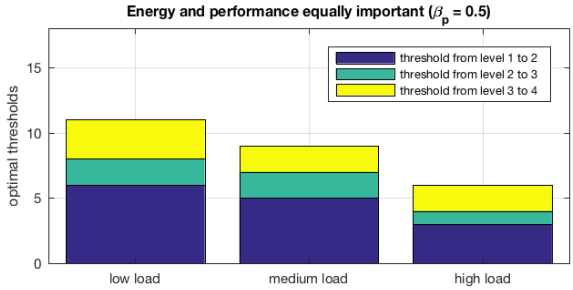
3.4 Example IV: Linux-like system

Our last example is devoted to a system inspired from a server running the Linux Operating System [LIN17]. The server operates at $L = 4$ levels corresponding to from 1 to 4 processor cores made available to execute requests ($c_\ell = \ell$, $\ell = 1, 2, 3, 4$). We assume that the request arrival process is non-Poisson with a coefficient of variation of the time between arrivals equal to 3 (cf. Appendix). We assume that the system capacity is limited to $N = 128$ requests. Based on actual energy consumption measurements in a Dell Power Edge server (with Intel Xeon processors) [DEL17], we assume that the energy consumption per time unit is given by a function of the form $b + c_\ell d$ where b denotes the base power consumption when the system is idle, c_ℓ is the number of processors corresponding to level ℓ and d is the average power consumption of a processor. We also assume that the activation time for adding processors is negligible. Here, we consider "hard" thresholds when activating and deactivating processor cores, i.e. switching operating levels.

Our goal is to determine threshold values that minimize a cost function combining system performance and energy consumption. Thus, our cost function comprises two components: energy cost and performance cost. For the former, since one processor is always active, we focus on the additional power consumption $\sum_{\ell=1}^{L} p(\ell)(c_\ell - 1)d$. Taking this value relative to the maximum additional power consumption $(c_L - 1)d$, we get as our energy cost $\eta_e = \frac{\sum_{\ell=1}^{L} p(\ell)(c_\ell - 1)}{c_L - 1}$. Note that this relative measure has a range of 0 to 1. As a measure of performance cost, we use the relative request response time $H = R/T$ where $T = 1/\mu(1,1)$ is the mean execution time by a single processor core. We denote by H_1 and H_L the corresponding relative response times in a single-level system with c_1 and c_L processors, respectively. Clearly, we have $H_L < H < H_1$. We use $\eta_p = \frac{H - H_L}{H_1 - H_L}$ as our performance cost measure. This relative measure, too, has a range of 0 to 1. Our overall cost function is defined as a linear combination of these two components $\eta = \beta_e \eta_e + \beta_p \eta_p$. The coefficients β_p and $\beta_e = 1 - \beta_p$ correspond to the relative importance we assign to performance versus energy consumption. Note that the first term in our cost function increases as the number of processors increases, while the opposite is true of the second term resulting in a tradeoff between performance and energy consumption.



(a) Optimal forward thresholds for three load levels with $\beta_p = 0.15$.



(b) Optimal forward thresholds for three load levels with $\beta_p = 0.5$.

Fig. 6: Example IV: optimal thresholds.

In our numerical study, we take the mean execution time by a single processor core to be 1 and we consider two sets of values for the weighting factors in our cost function: ($\beta_e = 0.85, \beta_p = 0.15$) and ($\beta_e = \beta_p = 0.5$). In exploring the values of thresholds, we concentrate on forward thresholds ($\gamma(n, \ell, k)$) and we fix backward thresholds ($delta(n, \ell, k)$) at 4/5 of the corresponding forward threshold. We consider three workload levels, referred to as low, medium and high. These workload levels correspond to mean arrival rates of 1.5, 2.5 and 3.5, respectively (recall that we assume a total of 4 processor cores). As illustrated in Figure 6a for $\beta_p = 0.15$, the threshold values that minimize

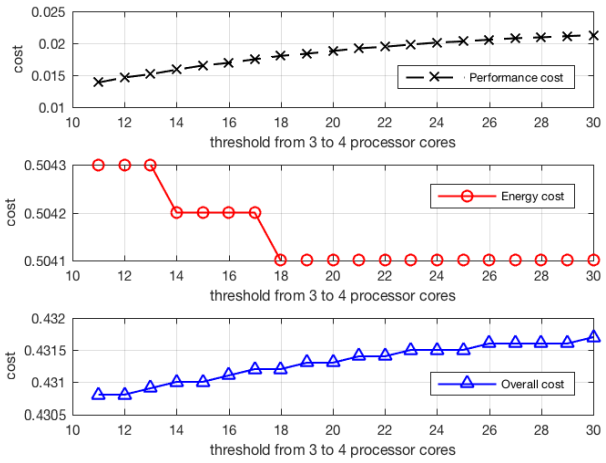


Fig. 7: Example IV: variation of cost components as a function of forward threshold between 3 and 4 processor cores.

our cost function (determined under the constraint of a fixed relationship between forward and backward thresholds) depend on the workload level. The higher the workload, the sooner a switch to higher operating level is required. Similar effect can be observed in Figure 6b for $\beta_p = 0.5$, and, as could be expected, with more emphasis on performance in this case, the threshold values tend to be smaller, i.e., switch to higher operating levels happens sooner. Finally, Figure 7 shows an example of how the components of our cost function vary as we increase the threshold value to switch from 3 to 4 cores, while keeping the other two threshold values constant. We observe that the relatively slow step-wise decrease in energy cost compared to the faster increase in our performance cost component. This figure was obtained with a mean request arrival rate of 2.5 and $\beta_p = 0.15$.

Note that in all our case studies the presence (or not) and location of "dips" and inflexion points depends on the values of thresholds at which switching between operating levels occurs.

4 Conclusions

We have presented a model of a system with multiple operating levels and possibly "soft" (probabilistic) thresholds for switching from one operating level to another. We assume an arbitrary number of servers and operating levels, each corresponding to an arbitrary number of additional servers. Our model accounts for possible delays in level activation, hysteresis in level switching, as well as for general (phase-type) times between request arrivals. We propose a semi-numerical solution for our model where a set of conditional state probabilities is computed using a fixed-point iteration. Our solution approach uses simple mathematics. Although we do not have a proof of convergence of our fixed-point iteration, in the over 60,000 examples we studied the method never failed to converge within typically between a few tens to a few hundreds of iterations.

We use four case study examples, mostly inspired by features of real systems, to illustrate the flexibility of our model. Our example inspired by the SMT feature in AIX accounts for the speed degradation of each server as the number of servers increases, as well as for arbitrary numbers of servers corresponding to different operating levels. Our example of Virtual Switches in the context of computer networks illustrates the potential importance of the arrival process on the performance of such systems. Our results indicate that the influence of the arrival process on the mean packet sojourn time is most visible at medium and high loads. Assuming Poisson arrivals would result in a significant underestimation of the mean packet sojourn time at medium loads, while at high loads it would underestimate packet losses. We use the example of Virtual Machine allocation in a cloud system to illustrate the potential importance of larger activation delays. Interestingly, the influence of activation delays is most visible for medium loads, which is most probably the desired operating region for these systems. Our example of processor core activation in a Linux-like environment clearly shows the dependence of optimal threshold values for switching between levels on system workload (in addition to the obvious dependence on the criterion used to determine such thresholds).

A natural extension of our work would be to attempt to relax the assumption of memoryless service times in our model.

Appendix A
Balance Equations with Memoryless Arrivals

For $n > 1$, we have

$$\begin{aligned}
p(n,\ell,k)[\lambda(n)+\mu(n,\ell)+kw] = \\
p(n-1,\ell,k)\lambda(n-1)[1-\gamma(n-1,\ell,k)] \\
+p(n-1,\ell,k-1)\lambda(n-1)\gamma(n-1,\ell,k-1) \\
+p(n,\ell-1,k+1)(k+1)\omega \\
+p(n+1,\ell,k)\mu(n+1,\ell)[1-\delta(n+1,\ell,k)] \\
+p(n+1,\ell,k+1)\mu(n+1,\ell)\delta(n+1,\ell,k+1) \\
+p(n+1,\ell,k)\mu(n+1,\ell+1)\delta(n+1,\ell+1,k)
\end{aligned}$$

For $n = 1$, we have

$$\begin{aligned}
p(1,\ell,k)[\lambda(1)+\mu(1,\ell)+kw] = \\
p(0,\ell,k)\lambda(0) \\
+p(1,\ell-1,k+1)(k+1)\omega \\
+p(2,\ell,k)\mu(2,\ell)[1-\delta(2,\ell,k)] \\
+p(2,\ell,k+1)\mu(2,\ell)\delta(2,\ell,k+1) \\
+p(2,\ell,k)\mu(2,\ell+1)\delta(2,\ell+1,k)
\end{aligned}$$

In the above equations, impossible terms are assumed to vanish and the last term is present only for $k = 0$. For $n = 0$, we assume that the system operates at level 1 and that there are no pending level activations so that we have $p(0,1,0) = \sum_{\ell=1}^{L}\sum_{k=0}^{L-\ell} p(1,\ell,k)\mu(1,\ell)$.

APPENDIX B
ADVANTAGES OF THE PROPOSED SOLUTION

Transforming the balance equations for our model into equations for the conditional probabilities $p(\ell,k|n)$ we get for $n>1$

$$\begin{aligned}p(\ell,k|n)[\lambda(n)+\mu(n,\ell)+kw]=\\ p(\ell,k|n-1)u(n)[1-\gamma(n-1,\ell,k)]\\ +p(\ell,k-1|n-1)u(n)\gamma(n-1,\ell,k-1)\\ +p(\ell-1,k+1|n)(k+1)\omega\\ +p(\ell,k|n+1)\mu(n+1,\ell)[1-\delta(n+1,\ell,k)]\lambda(n)/u(n+1)\\ +p(\ell,k+1|n+1)\mu(n+1,\ell)\delta(n+1,\ell,k+1)\lambda(n)/u(n+1)\\ +p(\ell,k|n+1)\mu(n+1,\ell+1)\delta(n+1,\ell+1,k)\lambda(n)/u(n+1)\end{aligned}$$

Similarly, for $n=1$ we get

$$\begin{aligned}p(\ell,k|1)[\lambda(1)+\mu(1,\ell)+kw]=u(1)\\ +p(\ell-1,k+1|1)(k+1)\omega\\ +p(\ell,k|2)\mu(2,\ell)[1-\delta(2,\ell,k)]\lambda(1)/u(2)\\ +p(\ell,k+1|2)\mu(2,\ell)\delta(2,\ell,k+1)\lambda(1)/u(2)\\ +p(\ell,k|2)\mu(2,\ell+1)\delta(2,\ell+1,k)\lambda(1)/u(2)\end{aligned}$$

As was the case for the balance equations, impossible terms are assumed to vanish and the last term in the above equations is present only for $k=0$. Note that $u(n)$ is given by formula (2) and we must have $\sum_{\ell=1}^{L}\sum_{k=0}^{L-\ell}p(\ell,k|n)=1$ for all values of n. These equations, considered in the order $n=1,2,\ldots$, can be solved using a simple fixed-point iteration. The use of conditional probabilities has the effect of partitioning the state space into independently normalized probabilities for each value of n. This has the potential of enhancing the numerical stability of iterative solutions by reducing round-off errors. Additionally, in the case when there are no level activation delays, for any values of n for which operating levels dont overlap, we have $p(\ell,0|n)=1$ for the single level ℓ corresponding to n. This has the potential of reducing the computational effort in models without activation delays. Note also that in the case of infinite population size ($N=\infty$), the conditional probabilities $p(\ell,k|n)$ tend to a limiting distribution as $n\to\infty$. In practice, this asymptotic convergence tends to happen for reasonably small values of n thus avoiding arbitrary truncation.

Denote by the superscript the current iteration number in a fixed-point iteration. A simple implementation could start with a feasible set of initial values for the conditional probabilities $p^0(\ell,k|n)$ for $n=1,2,\ldots$ and the corresponding conditional rates of request completions $u^0(n)=\sum_{\ell=1}^{L}\mu(n,\ell)\sum_{k=0}^{L-\ell}p^0(\ell,k|n)$. Let $\pi(\ell,k|n)$ denote non-normalized values corresponding to $p^i(\ell,k|n)$. Then, enumerating system states in the order of increasing values of ℓ for consecutive increasing values of $n=1,2,\ldots,$ at each iteration we can compute

$$\begin{aligned}\pi^i(\ell,k|1)=[\lambda(1)+\mu(1,\ell)+kw]^{-1}[u^{i-1}(1)\\ +\pi^i(\ell-1,k+1|1)(k+1)\omega\\ +p^{i-1}(\ell,k|2)\mu(2,\ell)[1-\delta(2,\ell,k)]\lambda(1)/u^{i-1}(2)\\ +p^{i-1}(\ell,k+1|2)\mu(2,\ell)\delta(2,\ell,k+1)\lambda(1)/u^{i-1}(2)\\ +p^{i-1}(\ell,k|2)\mu(2,\ell+1)\delta(2,\ell+1,k)\lambda(1)/u^{i-1}(2)]\end{aligned}$$

For $n>1$ we have

$$\begin{aligned}\pi^i(\ell,k|n)=[\lambda(n)+\mu(n,\ell)+kw]^{-1}\\ [p^i(\ell,k|n-1)u^{i-1}(n)[1-\gamma(n-1,\ell,k)]\\ +p^i(\ell,k-1|n-1)u^{i-1}(n)\gamma(n-1,\ell,k-1)\\ +\pi^i(\ell-1,k+1|n)(k+1)\omega\\ +p^{i-1}(\ell,k|n+1)\mu(n+1,\ell)[1-\delta(n+1,\ell,k)]\lambda(n)/u^{i-1}(n+1)\\ +p^{i-1}(\ell,k+1|n+1)\mu(n+1,\ell)\delta(n+1,\ell,k+1)\lambda(n)/u^{i-1}(n+1)\\ +p^{i-1}(\ell,k|n+1)\mu(n+1,\ell+1)\delta(n+1,\ell+1,k)\lambda(n)/u^{i-1}(n+1)]\end{aligned}$$

As before, impossible terms are assumed to vanish and the last term in the above equations is present only for $k=0$. In this approach, we immediately use newly computed values $\pi^i(\ell,k|n)$, then, as soon as all $\pi^i(\ell,k|n)$ values for a given $n=1,2,\ldots$ have been obtained, we normalize them to get $p^i(\ell,k|n)=\pi^i(\ell,k|n)/\sum_{\ell=1}^{L}\sum_{k=0}^{L-\ell}\pi^i(\ell,k|n)$. Of course, more sophisticated iterative schemes can easily be devised.

APPENDIX C
CASE STUDY DETAILS

In all examples, the stringency for the convergence of the fixed-point iterative solution of the model with Poisson or quasi-Poisson arrivals was set to 10^{-6} for the maximum relative difference in conditional probabilities. In the case of non-Poisson arrival, we used $\epsilon=10^{-4}$ (cf. Algorithm 1) for the relative difference between $\bar{n}$.

For all examples, only non-zero threshold values are specified.

Example I - AIX SMT inspired example

"Soft" threshold values used:
$\gamma(17,1,0)=1/3$, $\gamma(18,1,0)=2/3$, $\gamma(19,1,0)=1$, $\gamma(35,2,0)=1/3$, $\gamma(36,2,0)=2/3$, $\gamma(37,2,0)=1$, $\delta(14,2,0)=1/3$, $\delta(13,2,0)=2/3$, $\delta(12,2,0)=1$, $\delta(26,3,0)=1/3$, $\delta(25,3,0)=2/3$, $\delta(24,3,0)=1$.

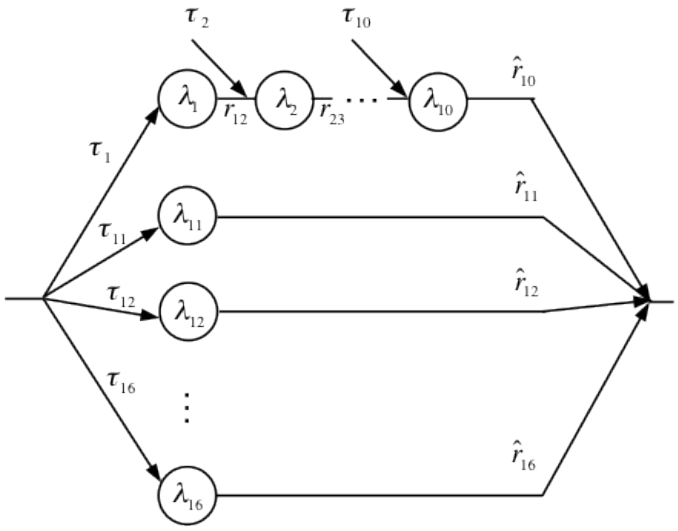
Example II vSwitch inspired example

"Hard" threshold values used:
$\gamma(9,1,0)=1$, $\gamma(19,2,0)=1$, $\gamma(29,3,0)=1$, $\gamma(39,4,0)=1$, $\gamma(49,5,0)=1$, $\gamma(59,6,0)=1$, $\gamma(69,7,0)=1$, $\delta(6,2,0)=1$, $\delta(17,3,0)=1$, $\delta(27,4,0)=1$, $\delta(34,5,0)=1$, $\delta(41,6,0)=1$, $\delta(49,7,0)=1$, $\delta(55,8,0)=1$.

Figure 8 shows the inter-arrival time distribution used for this example.

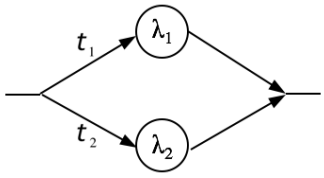
Example III cloud-computing inspired example

"Hard" threshold values used:
$\gamma(14,1,0)=1$, $\gamma(28,\ell,2-\ell)=1$ for $\ell=1,2$, $\gamma(42,\ell,3-\ell)=1$ for $\ell=1,2,3$, $\gamma(56,\ell,4-\ell)=1$ for $\ell=1,\ldots,4$, $\gamma(70,\ell,5-\ell)=1$ for $\ell=1,\ldots,5$, $\gamma(84,\ell,6-\ell)=1$ for $\ell=1,\ldots,6$, $\gamma(98,\ell,7-\ell)=1$ for $\ell=1,\ldots,7$, $\delta(12,\ell,2-\ell)$ for $\ell=1,2$, $\delta(28,\ell,3-\ell)$ for $\ell=1,2,3$, $\delta(32,\ell,4-\ell)$ for $\ell=1,\ldots,4$, $\delta(44,\ell,5-\ell)$ for $\ell=1,\ldots,5$, $\delta(58,\ell,6-\ell)$ for $\ell=1,\ldots,6$, $\delta(72,\ell,7-\ell)$ for $\ell=1,\ldots,7$, $\delta(86,\ell,8-\ell)$ for $\ell=1,\ldots,8$.



Probabilities		Phase rate	
τ_1	4.96299789e-002	λ_1	7.02937615e+000
τ_2	6.55622766e-002	λ_2	5.60291523e+000
τ_3	6.08048526e-002	λ_3	5.01429942e+000
τ_4	3.95306023e-002	λ_4	4.39854222e+000
τ_5	8.43336270e-002	λ_5	4.11181239e+000
τ_6	1.11445442e-001	λ_6	3.09159050e+000
τ_7	4.34658002e-002	λ_7	2.59431915e+000
τ_8	1.13779144e-002	λ_8	2.52635899e+000
τ_9	3.87506920e-002	λ_9	2.45646593e+000
τ_{10}	2.30172016e-001	λ_{10}	2.25907287e+000
τ_{11}	5.27071506e-007	λ_{11}	2.23401704e-005
τ_{12}	8.11915805e-006	λ_{12}	1.69154794e-004
τ_{13}	1.08429620e-004	λ_{13}	9.82553880e-004
τ_{14}	1.42476517e-003	λ_{14}	5.49823548e-003
τ_{15}	1.86758284e-002	λ_{15}	3.05903719e-002
τ_{16}	2.44709129e-001	λ_{16}	1.70040180e-001
$r_{12}, r_{23}, \ldots, = \hat{r}_{10}, \hat{r}_{11}, \ldots, = 1$			

Fig. 8: Pareto-like distribution with $a = 16$ phases for the time between arrivals in Example II.



Probabilities		Phase rate	
τ_1	2.8595933766e-005	λ_1	2.6666667e-003
τ_2	$1 - \tau_1$	λ_2	1.01081081

Fig. 9: Hyperexponential distribution with $a = 2$ phases for the time between arrivals in Example IV.

Example IV Linux inspired example

Figure 9 shows the inter-arrival time distribution used for this example.

TABLE 1: Notation used.

Main notation	
C	Number of servers
N	System capacity
L	Number of operating levels
n	Current number of requests in the system
ℓ	Current operating level
c_ℓ	Number of active processor at level ℓ
$\mu(n, \ell)$	Rate of request completions given n and ℓ
$\lambda(n)$	Rate of request arrivals given n (case of memoryless arrivals)
$1/\omega$	Mean time to activate a new operating level
k	Number of operating levels whose activation is pending
$\gamma(n, \ell, k)$	Probability of activating an additional operating level if n increases to $n + 1$
$\delta(n, \ell, k)$	Probability of activating an additional operating level if n decreases to $n - 1$
Intermediate quantities in the solution	
$p(n, \ell, k)$	Steady-state probability of the system being in state (n, ℓ, k)
$p(n)$	Marginal steady-state probability of having n requests in the system
$p(\ell, k\|n)$	Conditional probability of being at operating level ℓ with k pending levels given n
$u(n)$	Conditional rate of request completion given n
Performance metrics	
$\bar{n}$	Mean number of requests in the system
θ	Attained request throughput
R	Mean request response time
$p(\ell)$	Fraction of time the system spends on operating level ℓ
$\bar{c}$	Mean number of busy servers
$P_A(n)$	Probability that a request finds the system with n requests upon arrival
Additional quantities for general arrivals (phase-type distribution)	
a	Number of exponential phases in the arrival process
j	Current phase of the arrival process
τ_j	Probability that the arrival process starts in phase j
λ_j	Intensity of phase j
$\hat{r}_j$	Probability that the arrival process completes after phase j
r_{kj}	Probability that the arrival process continues to phase j upon completion of phase k
$p(n, \ell, k, j)$	Steady-state probability of the system being in state (n, ℓ, k) and the arrival process in phase j
$\alpha(n)$	Conditional rate of arrivals given that there are n requests in system
Case studies	
K	Number of exponential request sources (case of memoryless arrivals)
ϕ	Unitary rate of an active source (case of memoryless arrivals)
T	Mean request execution time

APPENDIX D
PRINCIPAL NOTATION USED

ACKNOWLEDGMENTS

This work was supported in part by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR). Part of this work was also supported by ANR MARMOTE (ANR-12-MONU-0019) and Télécom SudParis. The authors

wish to thank Dr. Kenneth James for his helpful remarks and suggestions.

REFERENCES

[AIT15a] Aït-Salaht, F., and Castel-Taleb, H. (2015, July). The threshold based queueing system with hysteresis for performance analysis of clouds. InComputer, Information and Telecommunication Systems (CITS), 2015 International Conference on(pp. 1-5). IEEE.

[AIT15b] Aït-Salaht, F., and Castel-Taleb, H. (2015, October). Stochastic bounding models for performance analysis of clouds. InComputer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015 IEEE International Conference on(pp. 603-610). IEEE.

[AIX17] IBM AIX 7.2, http://www-03.ibm.com/systems/power/software/aix/

[ALL90] Allen, A. O. (1990).Probability, Statistics, and Queueing Theory: With Computer Science Applications. Gulf Professional Publishing.

[ATM16] Atmaca, T., Begin, T., Brandwajn, A., and Castel-Taleb, H. (2016). Performance evaluation of cloud computing centers with general arrivals and service.IEEE Transactions on Parallel and Distributed Systems,27(8), 2341-2348.

[BOB05] Bobbio, A., Horvth, A., and Telek, M. (2005). Matching three moments with minimal acyclic phase type distributions. Stochastic models, 21(2-3), 303-326.

[BRA12] Brandwajn, A., and Begin, T. (2012). A Recurrent Solution of $Ph/M/c/N$-like and $Ph/M/c$-like Queues.Journal of Applied Probability,49(01), 84-99.

[DEL17] Dell PowerEdge R815 Rack Server, http://www.dell.com/us/business/p/poweredge-r815/pd

[EGI13] Egi, N., Iannaccone, G., Manesh, M., Mathy, L., and Ratnasamy, S. (2013). Improved parallelism and scheduling in multi-core software routers. The Journal of Supercomputing, 1-29.

[FOX09] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... and Stoica, I. (2009). Above the clouds: A Berkeley view of cloud computing.Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS,28(13), 2009.

[GOL97] Golubchik, L., and Lui, J. (1997, June). Bounding of performance measures for a threshold-based queueing system with hysteresis. InACM SIGMETRICS Performance Evaluation Review(Vol. 25, No. 1, pp. 147-157). ACM.

[HAR13] Harchol-Balter, M. (2013).Performance modeling and design of computer systems: queueing theory in action. Cambridge University Press.

[HAW14] Hawilo, H., Shami, A., Mirahmadi, M., and Asal, R. (2014). NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC).IEEE Network,28(6), 18-26.

[IBE93] Ibe, O. C., and Keilson, J. (1995). Multi-server threshold queues with hysteresis.Performance Evaluation,21(3), 185-213.

[JOH88] Johnson, M.A.and Taaffe, M.R. (1988). The denseness of phase distributions. School of Industrial Engineering, Purdue University.

[LEN02] Le Ny, L. M., and Tuffin, B. (2002).A simple analysis of heterogeneous multi-server threshold queues with hysteresis. In Proceedings of the Applied Telecommunication Symposium, SCS.

[LIN17] Debian GNU/Linux, kernel version 3.x

[LUI99] Lui, J. C., and Golubchik, L. (1999). Stochastic complement analysis of multi-server threshold queues with hysteresis.Performance Evaluation,35(1), 19-48.

[MIT11] Mitrani, I. (2011). Service center trade-offs between customer impatience and power consumption.Performance Evaluation,68(11), 1222-1231.

[MIT13] Mitrani, I. (2013). Managing performance and power consumption in a server farm.Annals of Operations Research,202(1), 121-134.

[OSO06] Osogami, T., and Harchol-Balter, M. (2006). Closed form solutions for mapping general distributions to quasi-minimal PH distributions.Performance Evaluation,63(6), 524-552.

[PAX95] Paxson, V., and Floyd, S. (1995). Wide area traffic: the failure of Poisson modeling.IEEE/ACM Transactions on Networking (ToN),3(3), 226-244.

[SMT17] IBM AIX 7.2 Simultaneous multithreading, https://www.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.genprogc/smt.htm

[VMW17] VMWare ESX Server, http://www.vmware.com/products/esx

RESEARCH CENTRE
GRENOBLE - RHÔNE-ALPES

Inovallée
655 avenue de l'Europe - Montbonnot
38334 Saint Ismier Cedex France

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr
ISSN 0249-6399