Accurate Modeling and Efficient QoS Analysis of Scalable Adaptive Systems under Bursty Workload

Diego Perez-Palacin[a], Raffaela Mirandola[a], José Merseguer[b]

[a]*Dip. di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy*
[b]*Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Zaragoza, Spain*

Abstract

Fulfillment of QoS requirements for systems deployed in the Internet is becoming a must. A widespread characteristic of this kind of systems is that they are usually subject to highly variable and bursty workloads. The allocation of resources to fulfill QoS requirements during the peak workloads could entail a waste of computing resources. A solution is to provide the system with self-adaptive techniques that can allocate resources only when and where they are required. We pursue the QoS evaluation of workload-aware self-adaptive systems based on stochastic models. In particular, this work proposes an accurate modeling of the workload variability and burstiness phenomena based on previous approaches that use Markov Modulated Poisson Processes. We extend these approaches in order to accurately model the variations of the workload strongly influence the QoS of the self-adaptive system. Unfortunately, this stochastic modeling may lead to a non tractable QoS analysis. Consequently, this work also develops an efficient procedure for carrying out the QoS analysis.

Keywords: Adaptability, Quality of Service, Stochastic Petri nets, Markov models, Workload modeling

1. Introduction

The Quality of Service (QoS) offered by systems is an important matter for their successfulness in the marketplace. This is exacerbated nowadays, in the era

Email addresses: diego.perez@polimi.it (Diego Perez-Palacin), raffaela.mirandola@polimi.it (Raffaela Mirandola), jmerse@unizar.es (José Merseguer)

of Internet services and online applications, where the popularity of systems with good functionality may be jeopardized by poor QoS, such as low performance or scarce availability. Examples of site degradation due to high workloads of Internet traffic abound, from legitimate requests, such as "flash crowds" effects, to disruptions due to malicious requests, such as denial of service attacks.

In order to build systems with good QoS, the formal methods community has achieved important advances [4]. Among other results, the QoS analysis leads to the identification of the amount of resources allowing a system to fulfill the required QoS (e.g., [5, 33]). However, in the last years the deployment of software systems has changed. Currently, they are not constrained to execute using a predefined number of resources, on the contrary, they can dynamically adjust their deployment as a response to changes in their execution context, such as changes in the workload. This could be obtained, for example, exploiting the elasticity and auto-scaling properties offered by cloud-computing. Hence, software service providers save costs during periods of time when the workload is low, but can also provide good QoS during workload peaks by provisioning an extra amount of resources temporarily. This type of systems are included in what it is called *self-adaptive systems* [12].

The model-based QoS analysis of a software that adapts its deployment to fulfill the required QoS while allocating the minimum amount of resources, is a challenging research topic that has not yet been completely addressed. Beyond traditional models of software behavior, we need models that represent, among others: adaptation policies, monitoring of the environment (to manage false positive adaptations or lacks of adaptations -false negatives-) and workload variations. In this work, we concentrate on the latter, the modeling of workload variations over time in the Internet, such as the requests supported by services, applications and websites.

It has been previously observed that the workload received by most of the systems operating on the Internet is highly variable and shows bursty behavior [8, 21], i.e., irregular spikes of congestion. Therefore, the models used to analyze QoS should be able to represent these characteristics. Otherwise, the model analysis can lead to optimistic results; e.g., it declares fair resource utilizations and probability of congestion, while in the real setting they would not be guaranteed. Formal methods of the stochastic family that have proved to be useful in modeling workloads with *burstiness* are the Markov Arrival Processes (MAP) and a concrete subtype of them, the Markov Modulated Poisson Process (MMPP) [16]. In particular, work on fitting MMPP [16, 18, 20, 21] and MAP [10, 11, 22] parameters from workload traces with *burstiness* is very useful for the analysis of QoS

properties, such as performance or availability, of a wide range of systems.

Considering workload-aware self-adaptive systems carefully, we can observe that they should adapt (e.g., provisioning or release of resources) when they recognize that the workload is changing. However, the usual techniques of MMPP fitting do not provide an accurate representation of the periods of time when the workload is changing. Although this fact does not prevent an accurate analysis of non-adaptive systems, it hampers the precise analysis of systems that adapt resources due to the variable workload, as we will illustrate later.

In this work, we propose an accurate modeling and an efficient QoS analysis of self-adaptive systems that execute under variable and bursty workloads. The accurate modeling builds on our previous work in [31], where we exploited an MMPP(2) model, now we extend it considering an MMPP(N) model, which allows for dealing with both, *short* and *long term* variability in the workload. However, this outstanding modeling when combined with the self-adaptive system model has a price, it may hamper the analysis, even it can turn into a non-tractable analysis problem. Hence, here we develop an approach based on Markov reward models for carrying out an efficient QoS analysis.

The rest of the paper is organized as follows. Section 2 motivates the need of accurate models for representing bursty workloads affecting adaptive systems. Section 3 shows the usefulness of MMPPs for modeling bursty workloads, but also its limitations for adaptive systems. Sections 4, 5 and 6 address these limitations. Section 7 evaluates the feasibility of the solutions here proposed. Section 8 discusses the benefits and limitations of the solution. Section 9 revises related work. Section 10 offers a conclusion.

2. Motivation

The goal of this section is twofold. Firstly, we want to show the usefulness of the adaptive systems for fulfilling QoS requirements in the context of bursty workloads; later, we recall the current modeling of bursty workloads using advanced stochastic formalisms.

Usefulness of the adaptive systems. Let us consider a service composed of just a single computational activity requiring an average 25 milliseconds of processing time to serve a single request. The system can queue up to ten requests, serving them following a FIFO policy, once the queue is full new requests will be discarded. The QoS requirement is that *at least* 99% *of requests must be served*, i.e., an availability requirement.

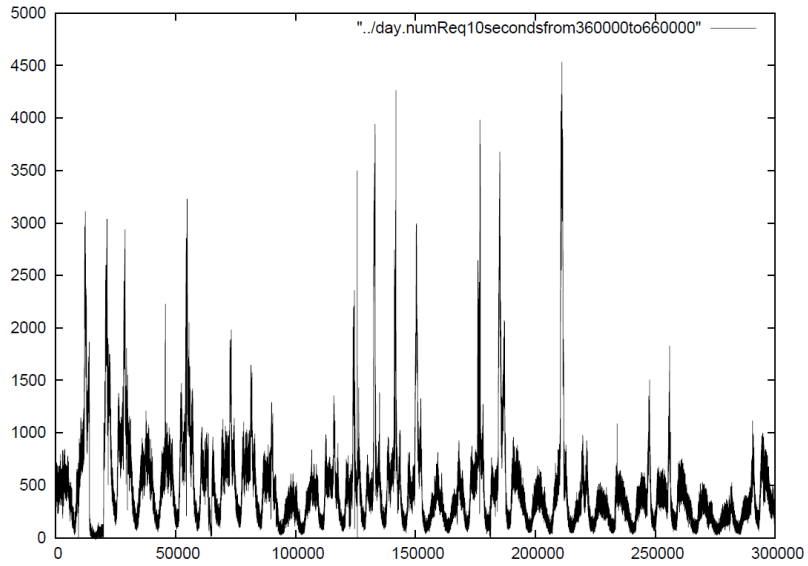

Figure 1: Requests every 10 seconds

We have implemented two Java programs for studying the QoS requirement: one program simulates a *static system* that uses a fixed number of servers to process the requests, and another that simulates an *adaptive system* that dynamically adapts the number of servers based on the current workload. Table 1 shows the results obtained by the execution of the programs using the workload[1] in Figure 1. For the static system, seven servers were required to achieve an availability of 99.07% (the availability obtained using six servers was 98.31%).

For the adaptive system we first calculated, for each possible number of servers, the maximum arrival rate of requests that they can support and still offer 99% of availability. Then, we simulated an adaptive system that adds one more server

[1]Along the paper we use a real bursty workload trace, illustrated in Figure 1, which plots the monitored arrival times of requests to the FIFA 1998 World Cup site [36]. This trace, despite its age, is yet one of the most detailed workload traces of a real service that can be found available for the public on the Web. The y axis counts the requests every ten seconds received by the Paris region server, while the x axis represents the flow of time. The shape of the graph depicts a quite bursty workload: the mean arrival rate of requests is 46.9 per second, but during 90% of the time it is under 86.2 req/s, while there are many peaks of short duration whose arrival rate can easily reach 350 req/s (i.e., around 7 times higher than the mean).

	No. servers used	Availability
Static system	7	99.07%
Adaptive system	3.62	99.943%

Table 1: Simulation results

(i.e., it changes from using c servers to $c+1$) when the workload exceeds the calculated maximum arrival rate for c servers; and releases one server (i.e., it changes from using $c+1$ to c) when the workload is below the calculated maximum arrival rate for $c-2$ servers; i.e., we followed a simple *hysteresis-based* approach to reduce the adaptations that were false positives. We considered that a server needs 1 minute for booting. Under these settings, the second row in Table 1 shows that an average number of 3.63 servers can achieve an availability of 99.94%. The number of servers used at any time is depicted in Figure 2.

As a conclusion we observe that the adaptive system can offer better QoS -more availability with less servers- than the static one. In fact, the former is provisioning servers during the bursty periods while releases them when they are no longer required.

Stochastic formalisms for modeling bursty workloads. Firstly, we modeled the static system as follows. The arrival process used was a Poisson process with $\lambda = 46.9$ req/s, i.e., the average arrival rate in the trace in Figure 1. In this case, the system and the workload exactly correspond to the classical $M/M/c/B$ queuing model. Figure 3 shows the M/M/c/B model using the Petri nets formalism that will be used in forthcoming evaluations along the paper. The service rate was exponentially distributed with mean $\mu = 40$, which is the inverse of the 25ms of the processing time. Secondly, we evaluated the queue for a different number of servers c, being the system capacity $B = c + 10$, in order to represent the ten queued requests. Thirdly, we analyzed such model and obtained that using two servers the 99.91% of requests can be served, result that is pretty far away from the one obtained in the simulation, which required seven servers for satisfying the 99.07% of requests.

The Poisson process as workload model is broadly used in stochastic analysis. However, it does not offer good results for modeling workloads with the bursty phenomenon, as it was previously presented in research works (e.g., [30]). Being the goal of this work the accurate modeling of bursty workloads for the QoS analysis of self-adaptive systems, we pursue more detailed descriptions of workload

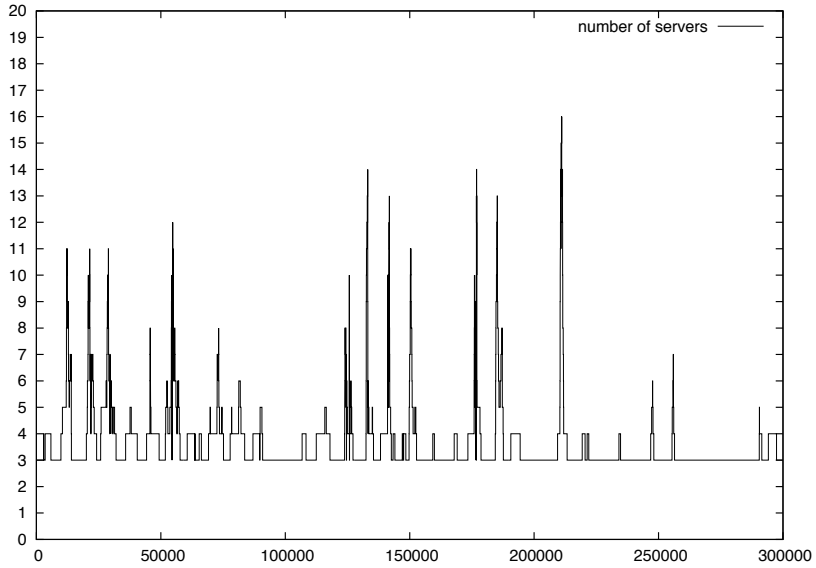

Figure 2: Number of Active Servers

variability. Next section starts the discussion.

3. Bursty Workload Modeling for Adaptive Systems

From the bursty workload trace in Figure 1 we can observe that the arrival rate is highly variable in the *long term*, meaning that the changes in the arrival rates differ by orders of magnitude. However, at the same time it is also highly probable that in the *short term*, i.e., few seconds in the future, the arrival rate will differ from the current one, but not much. These long and short term variabilities are well represented by the concept of *multiple workload states*, where each state is governed by an arrival rate. Since the arrival rate in each workload state is different, then, at a certain point in time it will be completely different from the arrival rate received a long time ago, hence, the *long term* variability is well represented. Regarding the *short term* variability, this is also modeled since the inter-arrival time between each two consecutive requests at each state follows a probability distribution function, instead of being a constant value. When the probability distribution is an exponential one, these concepts are exactly the ones represented by a Markov Modulated Poisson Process (MMPP).

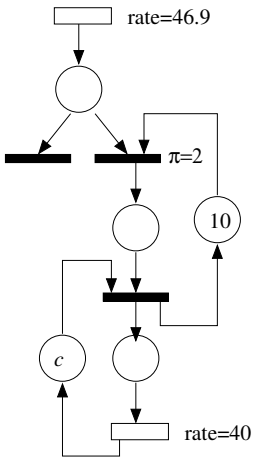

Figure 3: Stochastic Petri net representing the same M/M/c/B model of our example system

3.1. Markov Modulated Poisson Processes and Workload Fitting

MMPPs have been largely and successfully used in the literature as workload models for systems evaluation, mainly network traffic modeling [16, 18, 20]. MMPPs are suitable to model event arrival processes, high variability and autocorrelation for event generation. An MMPP is a stochastic process, the arrival rate at each moment is determined by the states of a continuous-time Markov chain (CTMC). So, when the chain is in state s_n, the arrival process is a Poisson process with rate λ_n. An MMPP with N states is defined by the NxN infinitesimal generator Q, which governs the state changes in the MMPP, and a vector Λ of N components representing the arrival rates in each state.

$$Q = \begin{pmatrix} -q_{11} & q_{12} & \dots & q_{1N} \\ q_{21} & -q_{22} & \dots & q_{2N} \\ & \dots & & \\ q_{N1} & q_{N2} & \dots & -q_{NN} \end{pmatrix}, \Lambda = (\lambda_1, ..., \lambda_N),$$

where $\forall n,\ \lambda_n > 0$ and $\forall n, n', (q_{nn'} \geq 0$ and $q_{nn} = \sum_{n':n' \neq n} q_{nn'})$.

To fit the parameters of the MMPP with N states we follow the algorithm presented in [21] and surveyed in [25]. Appendix A briefly recalls this algorithm. The algorithm requires as input a trace of counts $\mathcal{C}$ as the one that generated Figure 1 and a "width parameter" called a whose value is inversely correlated with the number of states for the resulting MMPP. We have applied the algorithm to the workload trace in Figure 1 with value of $a = 2$. We obtained an MMPP of

34 states and we applied such MMPP(34) as workload model to the *static* system described in Section 2. In this way we obtain an $MMPP/M/c/B$ queueing model whose analysis results show that $c = 7$ servers are necessary to obtain an availability of 99.11%. This is a very good result, pretty close to the 99.07% in Table 1. So the MMPP(34) largely improved the results obtained by the former Poisson process in the $M/M/c/B$ queue, which missed the burstiness characteristic. However, when the purpose is to evaluate a self-adaptive system instead of a static one, we have found limitations in the utilization of MMPPs as workload models. Next subsection describes these limitations and our proposal to deal with them.

3.2. MMPP(N) for Adaptive Systems

In order to easily illustrate the limitations of using MMPP(N) as workload models when evaluating self-adaptive systems, let us consider the basic case where: (1) the workload arrival rate remains constant for a certain time, (2) then it starts increasing up to reaching a certain threshold, say a target arrival rate, (3) it remains for a certain time in this threshold, (4) and then it starts decreasing until reaching the initial arrival rate, and then it cycles from (1) again. Figure 4 graphically depicts this situation, it has been artificially synthesized just for illustrating the limitation of MMPP(N).

Consider the application, over the trace in Figure 4, of the algorithm in Appendix A for MMPP fitting. It yields $N > 2$ states, with arrival rates $\lambda_1 > \lambda_2 > ... > \lambda_N$. In this case, where the variations are always periodic, half of the times the arrival rate in the next state increases and half of times decreases. For instance, in Figure 4, we observe that in I, III and V the next arrival rate is higher (workload is increasing) and in II, IV and VI the next arrival rate is lower (workload is decreasing). For this reason, the algorithm provides a fitted MMPP(N) modelling that, being in s_i, the probabilities to increment (decrement) the arrival rate in the next workload change are the very same. This happens for each intermediate state s_i, $1 < i < N$, which means that states s_{i-1} and s_{i+1} are equiprobable from s_i. This is statistically true indeed. However, this statistical truth does not represent well the actual behavior of the arrival rate variations. Looking at the trace, when the arrival rate is 50 and it comes from a tendency of increments, as in I, III and V, the next arrival is always above 50. However, the MMPP models an equiprobable choice between values above and below 50. The same happens for II, IV and VI, the next arrival rate is always below 50, rather than an equiprobable choice, as the MMPP models.

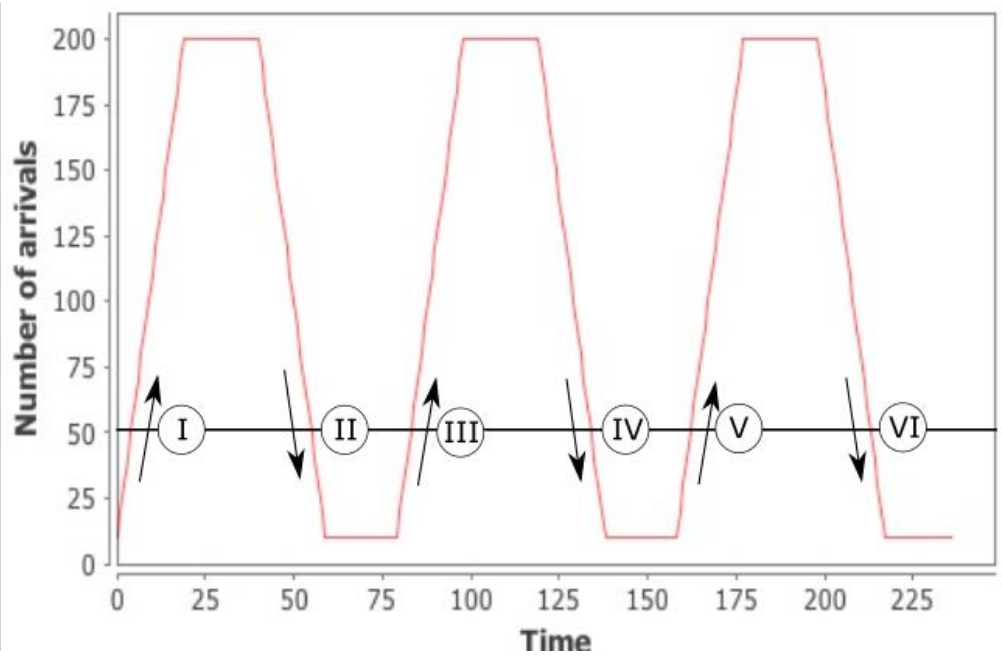

Figure 4: A synthetic workload trace with constant increments and decrements

To illustrate this limitation, we fitted the parameters of an MMPP with the data in the trace synthesized for depicting Figure 4. We obtained an MMPP(6) workload model and we wondered how accurately this model would represent the changes in the workload. We performed several experiments that simulated the arrival rate produced by this MMPP(6). Figure 5 depicts six samples of these experiments, illustrating the obtained variations of the arrival rate along time. Only the top-right chart brought a workload behavior similar to what it was expected. The other experiments failed in producing the expected workload variations due to the following reasons:

- The workload starts increasing soon after it has started to decrease, i.e., before reaching its minimum, as it happens in the top-middle and bottom-left chart.
- The workload starts decreasing before reaching its maximum, as it happens in the bottom-right chart.
- The workload, instead of ranging from the lowest to the highest, keeps its ar-

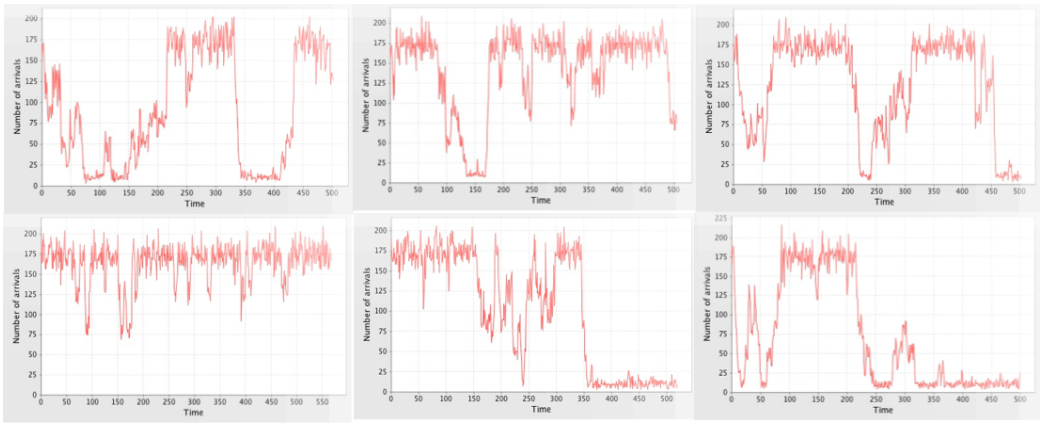

Figure 5: Six samples of the arrival rate generated by the MMPP fitted using the data in Figure 4

rival rate too much time in a medium value, fact that happens in the bottom-middle chart.

Therefore, even though the fitting of the MMPP is correct from a statistical point of view, in terms of arrival rates and transition rates, it is not correct from the point of view of the probability of providing large increments or decrements. For instance, this MMPP(6) has a probability of $0.5^4 = 0.0625$ for completing the sequence $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \rightarrow s_6$, while in the trace (Figure 4) this happens every time the workload starts decreasing from its maximum value. Therefore, *we can observe that this characteristic of the formalism is not very useful for modeling large continuous increments or decrements, say "tendencies", in the workload*. In fact this procedure for fitting MMPP(N) fails in the concept that the MMPP(2) used in [31] represented well which was the complete change between low workloads and bursts of requests.

Considering that we pursue the evaluation of self-adaptive systems that change their configuration depending on the workload they are receiving and that the workload can show "tendencies", then the statistically correct parameterization is not enough because it incurs in a lack of accuracy in some important characteristics of the workload. *Instead, we look forward to a workload model able to represent both, bursts and "tendencies"*. Figure 6 provides evidence of the "tendencies". It represents an excerpt of Figure 1 –concretely, from 132300 to 133000 in the x-axis–. It shows an increasing tendency of the real arrival rate, from around 500 to 3000 requests every 10 seconds, in a time interval of few minutes.

Lastly, a limitation arises if the analysis technique of the system model is based on state-space enumeration, such as those that can be applied to Markov chains or Petri nets. To exemplify this limitation, let us consider the nominal behavior of a self-adaptive system stochastically modeled by $k1$ states, and its workload modeled by a MMPP(k2) which consists of $k2$ states. So, for system analysis, their aggregation produces a complete model that, in the worst case, has $k1 \cdot k2$ states. The reason is simple, for each system state the workload can be in any of its states. Accurate workload models, as the ones proposed in this work, may entail a large number of states. These analysis techniques should be carefully used in order to avoid state explosion, which would make the analysis intractable.

Proposed solution

For a self-adaptive system that adapts due to workload changes, then periods in which the workload is incrementing or decrementing are of special importance since the adaptation should occurr. Let us call them *transient periods*. For this reason, we propose to identify the *stable periods* in the workload (i.e., periods when the workload does not vary much), to keep only the states in the MMPP that represent these periods[2], and to model separately and explicitly the *transient periods*. This will cope with the limitation regarding the lack of accuracy previously discussed. Additionally, in order to avoid the limitation caused by the possible state space explosion when evaluating the whole model that integrates stable and transient periods, we propose to analyze the system QoS for each *workload state* -stable or transient- separately, and to compose the results later so that it is not necessary to analyze the aggregated model.

Details on these steps are given in Sections 4, 5, and 6. In particular, Section 4 details the state space reduction of the MMPP in order to keep only the most important states. Section 5 presents models for *transient periods*. Section 6 provides the theories applied for obtaining the QoS of the system without requiring the evaluation of the whole aggregated model. It is worth noting that, from now on, all experimental results and figures only use real traces, so the synthetic one will be no longer used.

[2]From now on we will refer to *stable periods* as *stable states*, so to identify them with the MMPP states.

4. MMPP State Space Reduction

In this section we discuss methods that can reduce the amount of states yielded by the algorithm in Appendix A. The first solution that comes to mind is to increase the value of the input width parameter a used by the algorithm, which inversely correlates with the number of states of the resulting MMPP. However, according to the algorithm, the mean arrival rates λ_n of the proposed states calculated in such way would only depend on two values in the trace $\mathcal{C}$, the maximum and minimum arrival rate values found in the trace. The rest of values would be irrelevant for the calculation of every λ_n in Λ. In consequence, we prefer to first create an MMPP using a small value for parameter a, which will create a large amount of states, and then apply a filtering process to keep only those states in which the arrival rate is really most *stable*. For reducing the number of states, we have analyzed different options, which are discussed below: a) keep only the most visited states, b) keep only the states with longest "mean sojourn time" and c) keep only the states with longest maximum "sojourn time".

For ease of explanation let us use a trace, $\mathcal{W}s$, with the same number of entries as $\mathcal{C}$, so $|\mathcal{W}s| = |\mathcal{C}|$. Each entry $\mathcal{W}s_i$, $i \in [1...|\mathcal{C}|]$, stores the index of the workload state in the MMPP that can generate count c_i in $\mathcal{C}$. Therefore, $\mathcal{W}s_i \in [1...N]$. This trace can be easily derived from the information in $\mathcal{C}$ using the algorithm in Appendix A and the function $state(c_i)$ there described, which assigns a workload state for each possible count of requests. Consequently, position i in $\mathcal{W}s$, has value n, only if, in $\mathcal{C}$, the count of requests for i was generated by the state s_n. Formally, $\mathcal{W}s_i = n \iff state(c_i) = s_n$.

a) Most visited states: For each state we count how many times it appears in $\mathcal{W}s$. We then could select as *stable* states those with most occurrences in $\mathcal{W}s$. Even if this method is very simple and straightforward, we have observed, exercising the traces in Figures 1 and 13, that it is not convenient to use it. The reason relies on the fact that the states with extreme associated arrival rates (e.g., the set of states with highest associated arrival rate λ_n) are filtered because they usually have few occurrences. Nevertheless, these states are very important, even if they rarely happen, because they represent the worst-case scenarios. Overall this would be an insufficient model, and therefore we did not adopt this option but we continue searching for a better method.

b) States with longest "mean sojourn time": First, for each state we count how many times it appears in $\mathcal{W}s$. Second, for each state we count how many times

it happens that $\mathcal{W}s_i = n \ \wedge \mathcal{W}s_{i-1} \neq n$, which means how many times s_n has been visited from another state. Third, for each state we calculate its "mean sojourn time" (i.e., we divide the previous two counting values). Now we could select a percentage of states with the highest values. It is reasonable thinking that the states with large mean sojourn time are more *stable* than those states with low mean sojourn time. A reason can be that the latter show a low mean sojourn time because they are active only when the arrival rate is in the process of incrementing or decrementing. However, this option shows also a drawback. We have empirically observed from experimentation with traces in Figures 1 and 13 that the problem in this case is caused by the short-term variability of the workload. It specially happens with states whose associated arrival rate is low. The reason is that, according to [21] and the algorithm in Appendix A, the difference between each pair λ_n and λ_{n+1} progressively reduces while n increments. Therefore, for some s_n with low arrival rate, the short-term variability becomes significant, because it represents changes in the active state. These very frequent changes cause that the average sojourn time, in these affected states, are very low, and consequently they are filtered. Consider for instance this small sub-trace of $\mathcal{W}s$ $[..., s_2, s_2, s_2, s_2, s_2, s_3, s_2, s_3, s_3, s_2, s_3, s_3, s_3, s_3, s_3, ...]$. In a long-term view, values in the beginning are continuously s_2 and values in the last part are continuously s_3, so both states should present long mean sojourn times. However, they will appear with short mean sojourn times since in a short-term view the trace presents many transitions from s_2 to s_3 and vice versa, which increments their counts of times they are visited from other states, and hence it drastically reduces the results of their mean sojourn times. Therefore, the filtering of states obtained with this method is highly influenced by what is happening in the short-term arrival rate variability. For this reason this option is also considered not suitable to our needs. From the intermediate results, observed during our research, we envisioned that this drawback could be reduced through the utilization of dynamic values for *a*, the "width parameter" of the fitting algorithm. Increasing its value, when creating states representing lower arrival rate, we could mitigate the effect of the short-term variability. Such solution comes at the price of including additional parameters in the approach, those that will drive the variations for a. Consequently, we discarded this technique at present and we leave its research as future work. However, the main problems of the two presented methods can be eluded by considering for each state its longest sojourn time only, as follows.

c) States with longest maximum "sojourn time": In this case we consider for each state its maximum number of consecutive occurrences in the trace. More

formally, we could define an array $longestVisit$ of size N whose values are calculated as: $longestVisit_n = K$ such that

$$\exists_{i\in[1...|\mathcal{C}|]}\ \forall_{k\in[0...K-1]}\ \mathcal{W}s_{i+k} = n$$

$$\wedge$$

$$\nexists_{i\in[1...|\mathcal{C}|]}\ \forall_{k\in[0...K]}\ \mathcal{W}s_{i+k} = n$$

Consequently, each element in $longestVisit_n$ indicates the longest visit that s_n receives. We sum the values in the array as $sumL = \sum_n longestVisit_n$ and we keep the minimum set $\mathcal{S}$ of states whose sum of longest visits exceeds a given proportion p of $sumL$. Formally, we keep the set of states $\mathcal{S}$ such that

$$\sum_{s_n\in\mathcal{S}} longestVisit_n \geq sumL \cdot p$$

$$\wedge$$

$$\nexists\mathcal{S}'\ ((|\mathcal{S}'| < |\mathcal{S}|) \wedge \sum_{s_n\in\mathcal{S}'} longestVisit_n \geq sumL \cdot p)$$

It is worth noting that, in this manner, the number of resulting *stable* states is not preset by values p and N but it is tailored for each situation based on the relationships among values in $longestVisit$. This method reduces the influence of the short-term variability, but it is not perfect either. There might be stable states that are difficult to identify as such. Consider for instance the sub-trace $[..., s_2, s_2, s_2, s_2, s_2, s_3, s_2, s_2, s_2, s_2, ...]$. In a long-term view, if s_2 appeared in the workload continuously, then the maximum sojourn time of $longestVisit_2$ would be at least ten. However, there is a change to s_3 in the sixth position, which makes $longestVisit_2$ to be at least five. This problem will not cause an error, because if a state is visited for long periods, in the long-term view, some of its visits will actually show a long visit period in the short-term -even if the maximum in each case does not completely hold the same value-. Therefore, if s_2 is usually visited in the long-term view for periods longer than five, even not finding any sub-trace with 10 consecutive values, there can be sub-traces with 9, 8 or 7 s_2 consecutive visits. Then, we can accept this weakness and consider this method suitable enough.

For these reasons, in our approach *we adopt this method to obtain a new MMPP starting from the MMPP created by the algorithm in Appendix A,* as follows. Iteratively we remove from $longestVisit$ the state with lowest maximum "sojourn time". Assuming an iteration where s_n is removed then:

1. λ_n is removed then getting a new Λ.
2. $\mathcal{W}s$ is modified by overwriting to $\mathcal{W}s_i = \mathcal{W}s_i - 1$ all $\mathcal{W}s_i \geq n$ (or $\mathcal{W}s_i > n$ if $n = 1$).

Finally, the modified $\mathcal{W}s$ can be traversed for setting the transition rate values of the stochastic generator matrix Q in the same way as described in Section 3.1.

Applying this reduction to our previous MMPP(34), created from the workload trace in Figure 1 using $a = 2$ and $p = 50\%$, we got an MMPP(12). The number of states has been reduced by 65%, meaning that 35% of the states accounted for more than 50% of the longest sojourn times. Despite these good results, the work presented in this section could be further extended in the future. Better methods could be identified to reduce the state space or improve the presented ones, for example to overcome limitations, as it might happen with a state with only one time occurrence with a very long visit that, at present, would be classified as stable.

5. Modeling Workload Transient Periods

This section describes the stochastic modeling of the *transient periods* of the workload, i.e., those that represent increments or decrements in the arrival rate. Consider that MMPPs model the transition from one state to another as an immediate event, however, not all workload traces show such abrupt behavior, they may have progressive increments and decrements. Consequently, the explicit and accurate modeling of these increments and decrements will improve the system analysis results. Moreover, since system adaptations should occur during these workload variations, an explicit modeling and analysis may provide us with valuable knowledge of the system behavior beyond the QoS.

The idea of this modeling is to represent *large increments and decrements* in the arrival rate while we also keep a representation of the short-term variability. For example, Figure 6, which is an extract of the running example workload trace $\mathcal{C}$ in Figure 1, clearly shows both variabilities: a) a *large increment* in the arrival rate from 500 to 3000 requests every 10 seconds in a period of time around 30 minutes, and b) a *short term* variability where the arrival rate values show a high frequency of oscillation causing that the arrival rate value c_{i+1} is similar to the arrival rate value c_i but it may be higher, equal or even lower. *The large increments in a short time, but not instantaneous, is the type of behavior that lacks the MMPP modeling*.

Consider Figure 7(a), it represents the theory of what an MMPP produces when it changes from state s_n to $s_{n'}$ and vice versa. Although it represents long

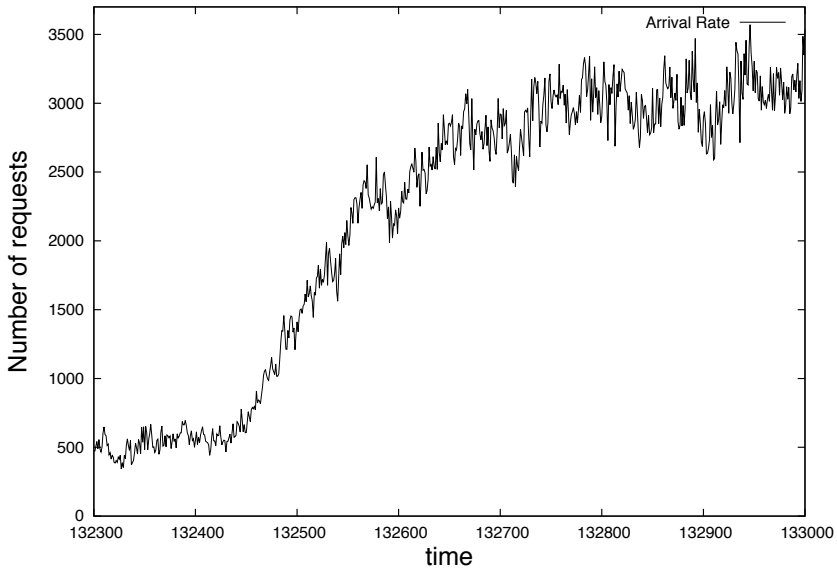

Figure 6: Variation in the arrival rate. Zoom-in from indexes [132300...133000] in Figure 1

term variability, it clearly depicts an abrupt change. However, Figure 7(b) models *transient periods*, which is a view of how real workloads usually change their arrival rate and therefore what it is desired to model. It represents long-term variability between arrival rates λ_n and $\lambda_{n'}$ as a linear increment —if $\lambda_n < \lambda_{n'}$— or as a linear decrement —if $\lambda_n > \lambda_{n'}$—. Note that, although this section deals with both short-term and long-term variability, we have filtered out the short-term one in Figure 7 only for visibility reasons.

For describing a linear increment (decrement) in the arrival rate, we just need the mean time for the workload to change, called $mt_{nn'}$ and $mt_{n'n}$ in Figure 7(b). This variation in the arrival rate (i.e., the change of speed, over time, in the reception of requests) fits the concept of acceleration[3].

The models for the increments and decrements presented in this section will provide even more accurate results than the MMPP models presented so far while

[3]We follow the concept of acceleration in physics, i.e., the second derivative of a given magnitude with respect to the time. Here, the magnitude would be the *requests*. Then, the first derivative would be the concept of *arrival rate* (or velocity of the arrival of requests), whose units are $requests/second$. While, the second derivative would be the acceleration, whose units are $requests/second^2$.

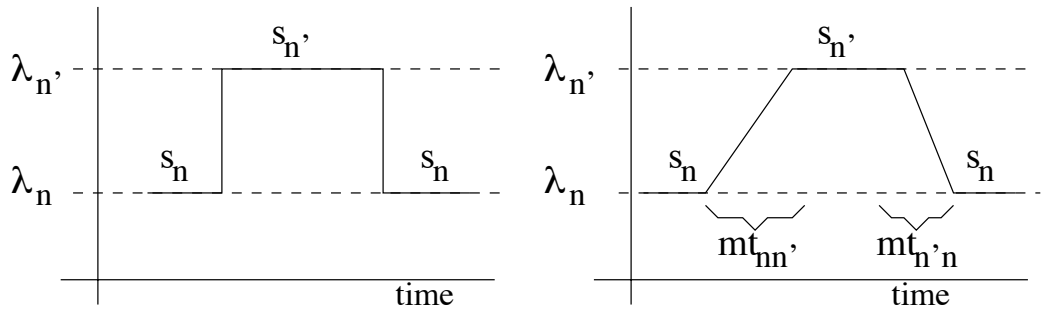

Figure 7: Workload models considering only states of the MMPP (a) and also including *transient periods* (b)

they still keep simplicity. Other approaches, as curve fitting algorithms, could offer more precise formal representations of the workload variability than the linear increments/decrements here modelled. However, such techniques are not directly applicable to the stochastic modelling we are using here to represent the system. Indeed, the more accurate modelling would be at the price of an higher complexity in the transformation to a stochastic workload model that could be effectively attached to the system model.

We devise two algorithms for calculating the mean time for the workload to change from a stable state s_n to another stable state $s_{n'}$, one for the case $\lambda_n < \lambda_{n'}$ and another for $\lambda_n > \lambda_{n'}$. *These algorithms generalize the one we presented in [31], which only dealt with the MMPP(2) case*. Appendix B explains in detail the algorithm we propose for the case $\lambda_n < \lambda_{n'}$, i.e., a period of increment in the arrival rate, the other algorithm would be very similar. The following lines provide a brief summary of the underlying idea of algorithm in Appendix B. The general idea for calculating the mean time that the workload takes to change from a stable state n to stable state n' —$mt_{nn'}$— is to identify every segment in trace $\mathcal{C}$ where the arrival rate shows a continuous increment between λ_n and $\lambda_{n'}$, and get the mean length of each of these segments. The decision whether a segment where the arrival rate changes from λ_n and $\lambda_{n'}$ is a continuous increment is not trivial. The immediate method of assessing if each arrival rate in the segment has a higher value than the arrival rate in the precedent position in the segment does not work well due to the short-term variability present in the workload. Thus, in

order to decide whether a segment represents a real increment, we propose to filter out the short-term variability by grouping the values of some continuous positions in the trace into a single value. The number of consecutive arrival rate values in the segment to include in each group is a parameter. Then, we compare the value of each group with the value of its successive group to asses whether they continuously increment.

5.1. Stochastic Models for Transient Periods

In order to analyze the system behavior during the *transient periods* of the workload we need to accurately model them. To this end we use GSPN [1], which is a language well suited for the modeling of system behavior. We present two GSPN models, which use the previously computed $mt_{nn'}$: one for increments and the other for decrements.

The **GSPN Model for Workload Increment** is represented in Figure 8(a). The arrival of requests is modeled by tokens in place $P_{arrivals}$. The idea is to augment the arrival rate from λ_n to $\lambda_{n'}$ in a mean of $mt_{nn'}$ time units.

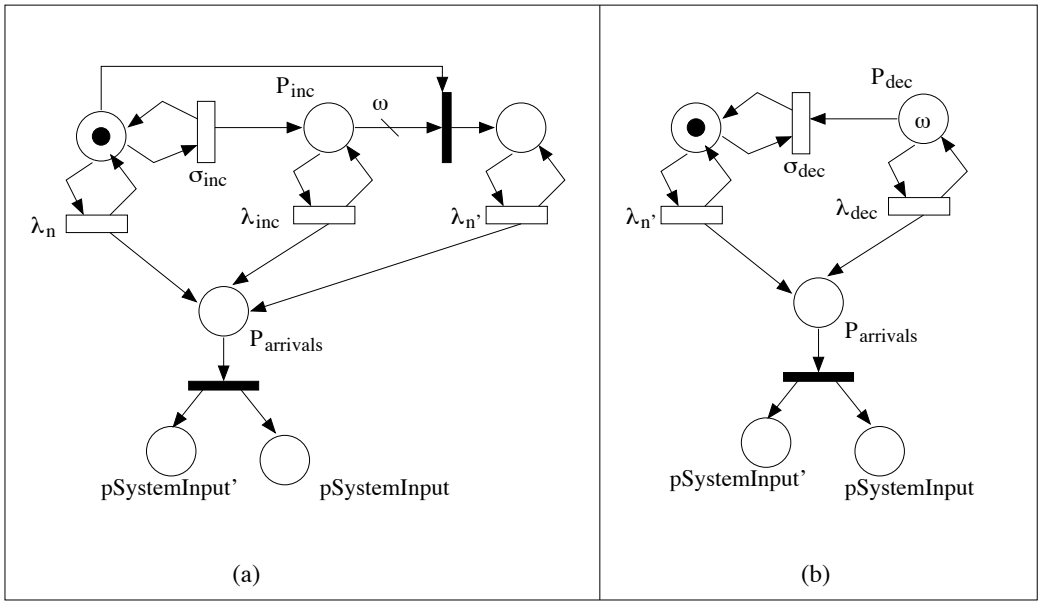

Figure 8: Transient period GSPN models: (a) workload increment and (b) workload decrement

The rate at which tokens are generated is $\lambda_n + \lambda_{inc} \cdot \#P_{inc}$.[4] In the beginning, $\#P_{inc} = 0$, then the arrival rate is λ_n, which is increased until $\lambda_{n'}$ along ω steps[5] . The parameters for achieving the objective of reaching $\lambda_{n'}$ in $mt_{nn'}$ time units are set as follows:

- $\lambda_{inc} = \frac{\lambda_{n'} - \lambda_n}{\omega}$, since we need ω steps to reach $\lambda_{n'}$.
- σ_{inc} represents the rate at which increments in the arrival rate happen. Since ω increments occur in $mt_{nn'}$ time units, then $\sigma_{inc} = \frac{\omega}{mt_{nn'}}$.
- ω is a user's choice. The higher, the more accurate the modeling, however at the cost of increasing the state space. Figure 9 illustrates examples for $\omega = \{5, 10, 20, 50\}$.

The expert reader can argue that the examples in Figure 9 could be equivalent to those produced by a fitted MMPP(X) where, $X = \omega + 1$ with

$$\Lambda = (\lambda_n, \lambda_n + \lambda_{inc}, \lambda_n + 2\lambda_{inc}, ..., \lambda_n + (\omega - 1)\lambda_{inc}, \lambda_{n'})$$

$$Q = \begin{pmatrix} -\sigma_{inc} & \sigma_{inc} & 0 & ... & & & 0 \\ 0 & -\sigma_{inc} & \sigma_{inc} & 0 & ... & & 0 \\ & ... & & & ... & ... & \\ 0... & & & & 0 & -\sigma_{inc} & \sigma_{inc} \\ 0... & & & & & 0 & \end{pmatrix}$$

This is true, and hence it can be also argued that it would be possible to elicit, from the input trace, an MMPP that includes all stable and transient periods while providing accurate modeling of the workload variability. This argument is also true. However, the problem of such solution resides in the implementation of a method to fit the parameters of an MMPP with such characteristics. We are not aware of any method, for such fitting, able to emphasize on gaining accuracy in the representation of the transient times when the system needs to adapt.

The **GSPN Model for Workload Decrement** is represented in Figure 8(b). In this case tokens in $P_{arrivals}$ are created at rate $\lambda_{n'} + \lambda_{dec} \cdot \#P_{dec}$. In the beginning, $\#P_{dec} = \omega$, which decreases at rate σ_{dec}. The parameters are set as follows:

[4] $\#P_{inc}$ means the number of tokens in place P_{inc}.

[5] It is worth noting that we are considering *infinite server* semantic for all transitions.

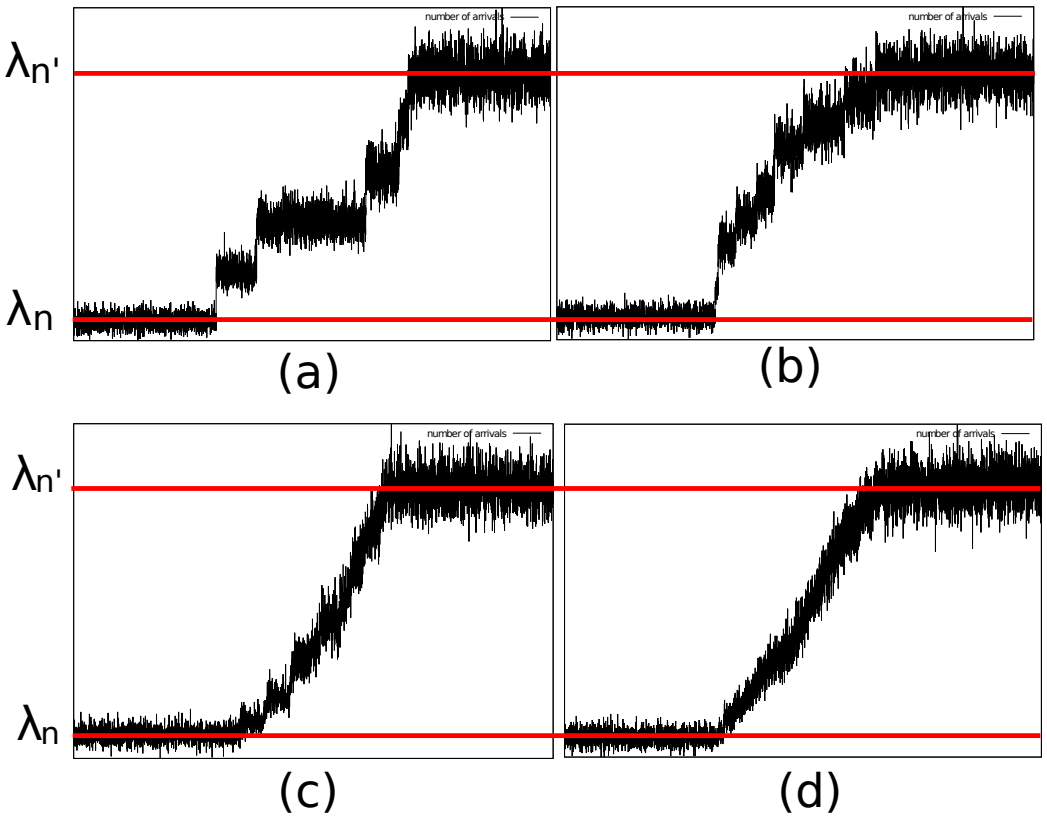

Figure 9: Four examples of workload models using (a) $\omega = 5$, (b) $\omega = 10$, (c) $\omega = 20$, (d) $\omega = 50$

- $\lambda_{dec} = \frac{\lambda_n - \lambda_{n'}}{\omega}$.
- $\sigma_{dec} = \frac{\omega}{mt_{nn'}}$.
- ω is again a user's choice.

6. Efficient QoS Analysis Guided by Workload Models

Once accurate models of both the stable and transient workload states are generated, it is the moment to evaluate the system QoS under these workload models. A method to create a complete workload model that aggregates these types of stables and transient models was proposed in [31]. That work dealt with the aggregation of transient workload models with an MMPP(2) for stable states -one for bursty arrivals and one for non-bursty-. However, if we would follow such

approach then the high number of states created by the current accurate modeling would result in a non-tractable analysis.

To overcome this limitation, rather than analyzing the aggregated model, we propose to analyze the system QoS for each *workload state* -stable or transient- separately, and to compose the results later. *The solution here proposed generalizes the one we proposed in [31]* by applying Markov reward models (MRM) theory [23].

It is worth noting that for our solution to be useful, the inter-arrival time of requests should be much lower than the mean time spent in each *workload state* for each visit. Fortunately, this is what usually happens for Internet services, changes in the workload take several minutes or even hours, while inter-arrival times of requests are in the hundredths of second.

Let us consider the states in the MRM as the union of the states in the MMPP defined in Section 4 (called *stable workload* states) and one state for each of the transient models (called *transient workload* states). Subsections 6.1 and 6.2 tackle the calculation of the reward for *stable workload* states and *transient workload* states respectively. Later, Subsection 6.3 focusses on the generation of the CTMC that governs the MRM and the achievement of QoS results.

6.1. QoS Analysis in Stable Workload States

In *stable workload* states the system only suffers short-term variability, that we model with exponentially distributed inter-arrival times -a standard technique for system QoS analysis-. The presence of short-term variability affects the system QoS, but it should not be a reason to adapt the system. The motivation is that the short-term variability oscillates with a frequency in the order of some seconds while the system adaptation —in terms of activating and booting new resources— may take several minutes. In this case, unless it were performed a very accurate prediction of the future workload values, which is a topic out of the scope of this work, when the system completes its self-adaptation the new system configuration would no longer be necessary. Such a frequent adaptation rate would create a too unstable system that moreover would not work better than a static system. Therefore, if only short-term variability existed in the workload, the system should be in a configuration that allows it to satisfy its QoS even with the inconvenience created by this frequent oscillations in the arrival rate. So, we assume that for each *stable workload* state an expected system configuration exists.

We create a parametric Markovian model called $M_{sys}(R)$ representing the system nominal behavior using *R* resources. As workload model we consider for each state s_n in the MMPP(N) proposed in Section 4, a Markovian model called M_{wk},

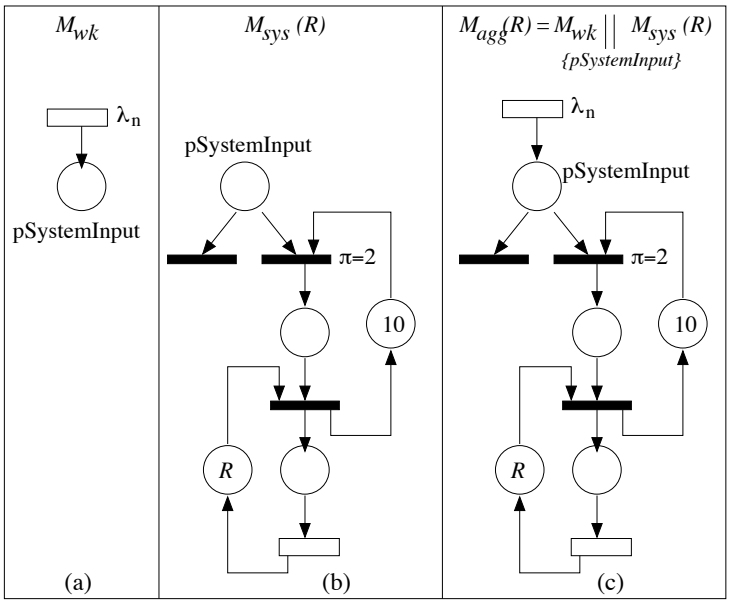

Figure 10: Examples of GSPN models: (a) M_{wk}, (b) $M_{sys}(R)$ of the system in Section 2 and (c) $M_{agg}(R)$

which represents an arrival rate of requests with λ_n. For system QoS analysis in s_n (i.e., whose results will become the reward values of the MRM for s_n) we create an aggregated[6] model $M_{agg}(R) = M_{sys}(R)||M_{wk}$. We can analyze $M_{agg}(R)$ to obtain: 1) the expected number of required resources r_n under workload M_{wk}; and 2) results for the QoS metrics we pursue (e.g., response time).

Figure 10 depicts examples of these models in the GSPN language: part (a) a workload M_{wk}, part (b) the single service system illustrated as running example in Section 2 $M_{sys}(R)$, and part (c) the aggregation of these models $M_{agg}(R)$.

6.2. QoS Analysis in Transient Workload States

In *transient workload* states the long-term variability in the arrival rate is represented and, consequently, the system adaptation for allocating or deallocating resources occurs. These periods are placed between *stable workload* states, as represented in Figure 9. Let us call $s_{nn'}$ the transient period between stable states s_n and $s_{n'}$. We create:

[6]For aggregation we use theory of place composition proposed in [15].

- $M_{sys}(R)$ as in previous subsection.
- A Markovian model, M_{adapt}, that represents the adaptation logic. We follow a simple model for the adaptation logic since this concern is not the main scope of the paper.
- A Markovian model, M_{wk}, that represents the increment (if $\lambda_n < \lambda_{n'}$) or decrement (if $\lambda_n > \lambda_{n'}$) in the workload, as described in Section 5.

For system QoS analysis in $s_{nn'}$ (i.e., the result that will be the reward of the MRM for $s_{nn'}$) we create an aggregated model $M_{agg} = M_{sys}(r_n)||M_{wk}||M_{adapt}$, where $M_{sys}(r_n)$ is an instance of $M_{sys}(R)$, $R = r_n$, r_n calculated as in previous subsection for stable state s_n. The main behavior of M_{agg} represents how the system serves requests while the workload is changing and the number of allocated resources vary from r_n to $r_{n'}$ according to the adaptation logic. Appendix C presents examples of M_{wk}, $M_{sys}(r_n)$, M_{adapt} and also of the final system M_{agg}.

We can analyze M_{agg} to obtain: *output1*) results for the QoS metrics we pursue; and *output2*) the mean time between the moment when the arrival rate starts incrementing until the moment when both the arrival rate has reached $\lambda_{n'}$ and the system is in a configuration to serve requests at such rate.

From *output2* and $mt_{nn'}$ –parameter calculated in Section 5– we can obtain $mt^{adapt}_{nn'} = output2 - mt_{nn'}$. It provides the mean time that passes between the moment when the workload has already reached $\lambda_{n'}$ and the moment when the system has just finished its adaptation and it is ready to appropriately serve requests at rate $\lambda_{n'}$.

6.3. Generation and Analysis of an MRM

We initially consider that the states of the MRM are those of our MMPP(N); the state transition matrix of the MRM, Q^{MRM}, will initially be Q, the CTMC given by the infinitesimal generator of the MMPP. Then, we need to extend the MRM to also consider the *transient periods*.

For including a transient period $s_{nn'}$, we:

1) remove transition $Q^{MRM}_{nn'}$;

2) add a state $s_{nn'}$ in Q^{MRM};

3) add a transition from s_n to $s_{nn'}$ with rate $Q_{nn'}$ in Q^{MRM};

4) add a transition from $s_{nn'}$ to $s_{n'}$ with rate $\frac{1}{mt_{nn'}+mt^{adapt}_{nn'}}$ in Q^{MRM}. Therefore, $Q^{MRM}_{n\to n',n\to n'} = -Q^{MRM}_{n\to n',n'}$ and for each $j \neq n'$, $Q^{MRM}_{n\to n',j} = 0$.

Figure 11 graphically details the inclusion of three *transient states*[7] in a CTMC.

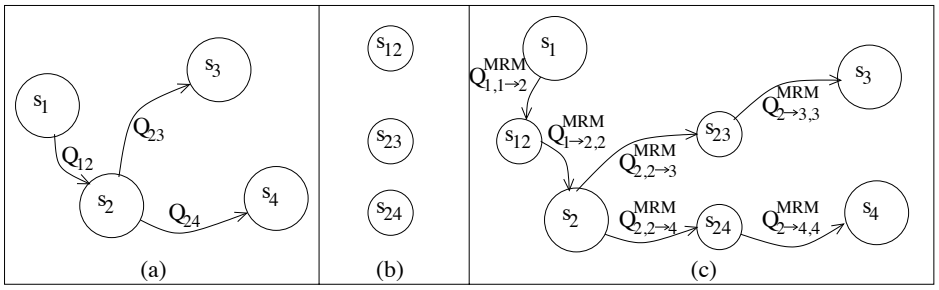

Figure 11: Modification of the CTMC: (a) Q, (b) *transient* states, (c) Q^{MRM}

Q^{MRM} needs to be adjusted because currently the time spent in *transient states* is also represented by the *stable states*, so we have to appropriately reduce the mean time in *stable states*. Figure 12 illustrates an example of this issue. For a stable state s_n, we have to consider the time already included in transient states that s_n reaches and those reached by s_n, as follows:

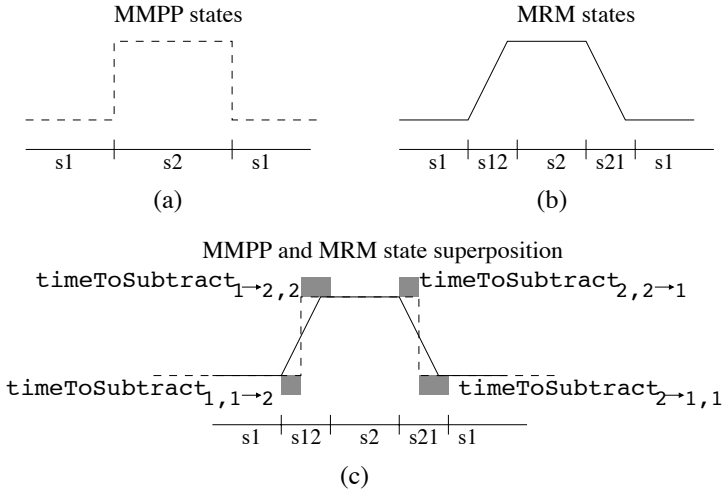

Figure 12: Time modeled in both states, stable and transient

[7]Since we have converted a transient period into a state of the MRM, from now on we will call transient states to transient periods.

- For each transient state s_{xn} that reaches s_n we consider $timeToSubtract_{x \to n,n} = mt_{xn}/2 + mt_{xn}^{adapt}$. The rationale is that mt_{xn} (i.e., mean time incrementing or decrementing) should be represented only by the *transient* state but now this time is also considered in the origin and target *stable* states. In consequence, we subtract half of it from the origin and half from the target. On the other hand, mt_{xn}^{adapt} (i.e., mean time to adapt) is included in the *transient* state and in the target state s_n, hence we subtract it from the latter. For example, in Figure 12(c),

  $$timeToSubtract_{1 \to 2,2} = \frac{mt_{12}}{2} + mt_{12}^{adapt}$$

  For each s_n we consider $meanTTSinputs_n$ as the mean of all $timeToSubtract_{x \to n,n}$.

- For each transient state s_{nx} reached by s_n we consider $timeToSubtract_{n,n \to x} = mt_{nx}/2$. In this case we subtract the half part corresponding to the origin, as explained before. We do not decrement adaptation time here since such a time should only affect the target states. For example, in Figure 12(c),

  $$timeToSubtract_{2,2 \to 1} = \frac{mt_{21}}{2}$$

  For each s_n we consider $meanTTSoutputs_n$ as the mean of all $timeToSubtact_{n,n \to x}$.

Finally, the mean sojourn time in state s_n is

$$\frac{-1}{Q_{nn}} - meanTTSinputs_n - meanTTSoutputs_n$$

and we appropriately update Q^{MRM} as

$$Q_{nn}^{MRM} = \frac{-1}{\frac{-1}{Q_{nn}} - meanTTSinputs_n - meanTTSoutputs_n}$$

We also need to update in Q^{MRM} the state transition rates of each *stable* state s_n to ensure that the sum of all its rates is equal to $-Q_{nn}^{MRM}$, while preserving the transition probabilities. For doing this, all transition rates from s_n are scaled by factor $\frac{Q_{nn}^{MRM}}{Q_{nn}}$.

For example, in Figure 11(c):

$$Q_{22}^{MRM} = \frac{-1}{\frac{-1}{Q_{22}} - (\frac{mt_{12}}{2} + mt_{12}^{adapt}) - (\frac{\frac{mt_{23}}{2} + \frac{mt_{24}}{2}}{2})}$$

and

$$Q^{MRM}_{2,2\to 3} = Q_{23}\frac{Q^{MRM}_{22}}{Q_{22}}$$

Now Q^{MRM} correctly characterizes the CTMC governing the MRM states -stable and transient-. We calculate the steady-state probability distribution of the CTMC, π^T, as the solution of $\pi^T Q^{MRM} = 0$ where $\pi^T \mathbf{1} = 1$. We refer to π_n as the probability of being in a stable state s_n and to $\pi_{nn'}$ as the probability of being in a transient state $s_{nn'}$. The steady-state expected reward, which is the steady-state QoS in this case, is calculated as usually [35, 19]:

$$qos = (\sum_{\forall\ s_n} \pi_n qos_n) + (\sum_{\forall\ s_{nn'}} \pi_{nn'} qos_{nn'})$$

Let us finally note that, although Q^{MRM} may potentially consist of N *stable* states plus $(N^2 - N)$ *transient* states, some *transient* states are not required for analysis. For example, between stable states s_n and $s_{n'}$ we do not need a transient one when:

- There is no one-step transition from s_n to $s_{n'}$, i.e., $q_{nn'} = 0$. Here we benefit from the reduction on the number of stable workload states and state transitions performed, since it helps to have a Q matrix populated with a high proportion of zeros.
- The expected system configuration in s_n is the same as in $s_{n'}$, i.e., $r_n = r_{n'}$, in this case there is no need to adapt the system during the workload change.

7. Evaluation

This section presents evaluations of the proposed approach. Such evaluations use aggregated models, $M_{agg}(R)$, which are compositions of a system model $M_{sys}(R)$, represented by a GSPN, and workload models M_{wk}.

The first evaluation is presented in Subsections 7.1 and 7.2. The first subsection presents experiments for creating a model of the workload that represents the behavior of the real workload trace in Figure 1. The system model used is the GSPN in Figure 10(b) and its workload models those in Figures 10(a) and 8. Subsection 7.2 presents and discusses the quality of the obtained results in terms of system availability evaluation.

The second evaluation is presented in Subsection 7.3. It has been carried out to provide additional validity to our approach.

a	num States	*a*	num States	*a*	num States	*a*	num States
1	67	*2*	34	*3*	23	*4*	17
5	14	*6*	12	*7*	10	*8*	9
9	8	*10*	7				

Table 2: States of the MMPP following the algorithm in [21] using different values for parameter *a*

7.1. Evaluation of the outcome of the algorithms

Here we present the results of our approach for creating the workload model varying the values of its parameters: a, p, L and ω.

Parameter a: As described in Section 3.1, it is a width parameter that is used for calculating the initial number of states in the MMPP and the arrival rate in each state. We have experimented with values from 1 to 10. Table 2 presents the results. It can be seen how, for low values, this parameter has a strong influence in the resulting number of states, while for large values the number of states proposed does not vary much. As proposed in Section 4, we will choose low values of *a* in order to first create a large number of states and then reduce them according to other characteristics of the workload trace beyond its maximum and minimum values.

Parameter p: As described in Section 4 it represents the percentage for reducing the number of states of the initial MMPP following the technique *states with longest maximum "sojourn time"*. We have experimented with values from $p = 100\%$ to $p = 20\%$ in steps of 10%. We have calculated the amount of states of the reduced MMPP when the initial MMPP was calculated with values from $a = 1$ to $a = 4$. These results are represented in Table 3, together with the percentage of reduction in the number of states resulting from pruning the percentage $100 - p$ of lowest maximum sojourn times. In Table 3 it can be seen that the sojourn times of the states are not homogeneous. For example, in the 10% of difference between using *p=100%* and *p=90%*, the number of states is reduced around the 20%. It means that the sojourn time of several of the states initially created by method [21] summarized in Appendix A for the trace in Figure 1 is much under the mean. However, in the 10% of difference between using *p=30%* and *p=20%* the number of states is reduced only around the 8%. This fact justifies the observation that there are states whose visits take much less time than other states, and the arrival rate values produced by these states can be considered as transient.

a=1 *Initial states = 67*								
p	**num States**	**%**	***p***	**num States**	**%**	***p***	**num States**	**%**
1	67	0%	0.7	39	42%	0.4	20	70%
0.9	55	18%	0.6	32	52%	0.3	14	79%
0.8	48	31%	0.5	26	61%	0.2	9	87%

a=2 *Initial states = 34*								
p	**num States**	**%**	***p***	**num States**	**%**	***p***	**num States**	**%**
1	34	0%	0.7	19	44%	0.4	10	71%
0.9	27	21%	0.6	16	53%	0.3	7	79%
0.8	23	32%	0.5	12	65%	0.2	4	88%

a=3 *Initial states = 23*								
p	**num States**	**%**	***p***	**num States**	**%**	***p***	**num States**	**%**
1	23	0%	0.7	11	52%	0.4	6	74%
0.9	18	22%	0.6	9	61%	0.3	4	83%
0.8	14	39%	0.5	7	70%	0.2	3	87%

a=4 *Initial states = 17*								
p	**num States**	**%**	***p***	**num States**	**%**	***p***	**num States**	**%**
1	17	0%	0.7	9	47%	0.4	4	76%
0.9	13	24%	0.6	7	59%	0.3	3	82%
0.8	11	35%	0.5	5	71%	0.2	2	88%

Table 3: States of the reduced MMPP for different values of width parameter a and proportion parameter p

Parameter L: This parameter is used when calculating the mean time that the workload takes to change, process that was summarized in Section 5 and detailed in Appendix B. This parameter represents the number of consecutive arrival rate values in a segment that will be grouped in a single value in order to decide if the segment represents a continuous increment/decrement in the workload or it should otherwise be ignored. We have experimented with a set of values that range from 30 seconds to 2.5 minutes in steps of 30 seconds. That is, when an interval of change in the workload between states is discovered, to filter out the short term variability, we group the number of requests in periods of length *L*. We have also checked the amount of real increments/decrements obtained if parameter *L* would not be used. Table 4 depicts, in function of *L* values (represented in minutes), the number of state changes that existed in the example trace and the number of them that were considered real increments or decrements to calculate the mean changing times between all states. The results of the experiment that did not consider parameter *L* are represented in the table as "No L". We have observed the variation of *L* in the number of states created by setting $a = 2$ and ranging p between 0.7 and 0.4 in steps of 0.1.

It can be concluded that, the larger the value of *L* the more changing intervals are considered real increments/decrements. The reason is that the short-term variability in the arrival rate is filtered in a higher degree, which in turn also entails that there might be considered as real increments/decrements some of the intervals that should have been ignored. It is worth noting that, in some cases, if the *L* parameter is not used, none of the intervals initially found between two states were identified as "real increment" or "real decrement". For instance, this fact happened 5 times for the cases of *p=0.5* and *p=0.4*.

Parameter ω: As described in Section 5.1 it is used to stochastically model the transient increments and decrements in the workload. We refer to the study shown in Figure 9 to show the effect of its different values.

7.2. Quality of the results

The goal is to compare the QoS results obtained by our approach with those in Table 1, which were obtained by using each single value in the trace of Figure 1 to simulate the system behavior along time. We parameterized the model as follows:

- We have chosen $a = 2$ because it is small enough as to generate a high quantity of states in the MMPP. At the same time, this value is also large enough for producing states with arrival rates with sufficient separation.

a=2 p=0.7 allIntervals=48407					
L	**real changes identified**	**L**	**real changes identified**	**L**	**real changes identified**
No L	22967	1	44177	2	46324
0.5	39639	1.5	45602	2.5	46749

a=2 p=0.6 allIntervals=32444					
L	**real changes identified**	**L**	**real changes identified**	**L**	**real changes identified**
No L	15231	1	29337	2	30797
0.5	26350	1.5	30286	2.5	31112

a=2 p=0.5 allIntervals=12990					
L	**real changes identified**	**L**	**real changes identified**	**L**	**real changes identified**
No L	5620	1	11360	2	12071
0.5	10000	1.5	11830	2.5	12220

a=2 p=0.4 allIntervals=3276					
L	**real changes identified**	**L**	**real changes identified**	**L**	**real changes identified**
No L	1300	1	2541	2	2733
0.5	2220	1.5	2657	2.5	2788

Table 4: Number of state changes and number of them that were considered real increments/decrements

The latter is useful in case that two neighbor states are both selected as *stable states* by the state reduction algorithm. If arrivals in each state will be generated following the exponential distribution -as it happens in MMPP-, the value $a = 2$ allows the identification of the state that generated each arrival in the workload log with high confidence during the fitting process.

- For p we have chosen 50% because in Table 3 for *a=2* this is the threshold from which the reduction in the number of states is less than the increment of p (i.e., above $p = 50\%$, each increment of 10% in p creates a reduction lower than 10% in the number of states).

- L is set to one minute since each count in the workload trace represents a time interval of ten seconds, then $L = 6$.

- Parameter ω is set to 10.

With this setting, our approach created 12 stable states (as corresponding to the value given in Table 3 for *a=2* and *p=50%*) and 19 transient states (14 representing an increment in the workload and 5 a decrement) with a corresponding MRM of 31 states. From the MRM evaluation we obtained a system availability of 99.96%. The availability of the real system obtained from the simulation with the actual data in the workload log was 99.94%, as showed in Table 1.

To gain more insight about the behavior of the system, using these results we have also calculated that the workload is in *stable* states the 98.4% of time, while the remaining 1.6% is in *transient* states. Moreover, we have measured that, in stable states, the system reaches an availability of 99.99%, while in transient states the availability is 99.27%.

The importance of considering both the transient workload times and the system adaptation times in their model-based representations is motivated by the following facts:

- *If the model does not represent adaptation times*. It means that the model represents a system that could instantaneously change the amount of used resources in any moment. In other words, this representation is equivalent to assume that the system is always using the correct amount of resources. A similar concept is what our MRM would represent if it would only take into account the results in its *stable* states. The availability evaluation of these states provided us the result of 99.99%. Therefore, if the model did not represent adaptation times, we would fall into an overestimation of the system availability.

- *If the model does not represent transient times*, we can obtain an underestimation of the availability. The reason is that the model represents a system that starts its adaptation to use the correct amount of resources when the workload has completely changed. For example, there would be modeled that the system starts adapting when the workload is already high, then resulting in periods where the availability obtained from the model is lower than the real one. We do not provide a quantity for this result because the straightforward model of this situation incurs in the state-space explosion. However, in [31] is described an example of this kind of underestimation that was possible to quantify because the workload model consisted of only two states and hence it was tractable.

7.3. Application to the MAWI dataset

This section addresses a second evaluation of our approach. Again, we are using a real workload trace, in this case provided by the MAWI[8] dataset [13, 27]. In particular, we report results for a 96-hours trace[9], which contains a sequence of requests during continuous time intervals, then featuring more than 6 billions of events. Figure 13 depicts the traffic variability during these four days, grouped in periods of ten seconds. We can easily distinguish four cycles of day and night. We can also see a heavy increment in the traffic around index 15000 and many traffic bursts along the trace that can reach about 500000 events every ten seconds, which happen suddenly and do not last for much time.

As for the real system to consume the workload, we have experimented with one similar to that used in the previous evaluation, so to match assumptions and results. The assumptions are then as follows. The system can queue 10 packets before start serving them, and a packet will be lost if it finds the queue plenty upon arrival. To manage a reasonable number of resources, we assume that each one can serve 20000 packets per second (note that if using a rate of 40, as for the FIFA case, our system then would need more than a thousand of resources). Finally, the results obtained were that the 99.9968% of packets can be served and that the average number of active servers should be 3.55.

Next, we simulated the adaptive system with the new trace and the proposed *hysteresis-based* approach. As parameters, we used $a = 2$ and $p = 50\%$. We then obtained an MMPP made of 286 states, which was later reduced to 66 after

[8]MAWI offers traces of a backbone, which is a transit link of WIDE to an upstream ISP. The traces go from 1999 to 2017, since 2006 they cover daily traffic.

[9]Link to the traces: `http://mawi.wide.ad.jp/mawi/ditl/ditl2009/`

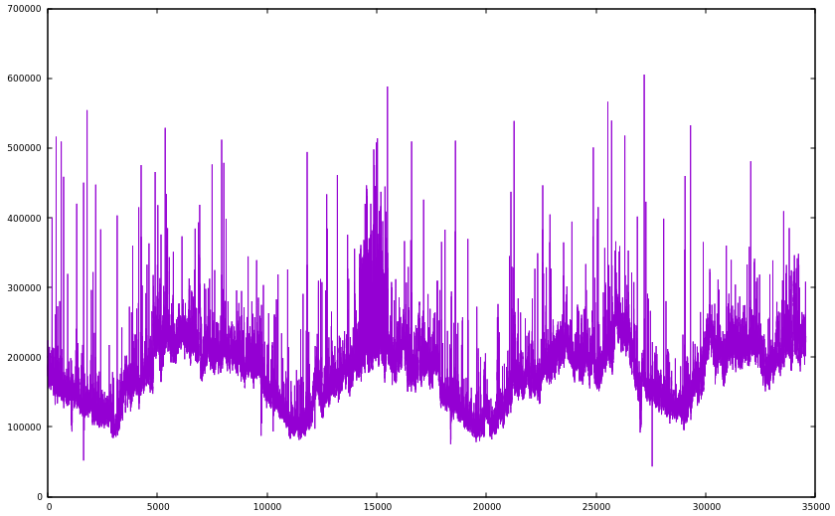

Figure 13: Requests every 10 seconds of MAWI 96-hours trace

applying the proposed state space reduction. The MMPP(66) still had more than two thousand of transitions between states. For pragmatic reasons, we applied again the approach with $a = 4$ and $p = 25\%$, which produced an initial MMPP of 143 states. It was reduced to an MMPP(15), the number of transients states identified for the MRM were 16, where 13 of them correspond to increments in the arrival rate and the other 3 to decrements. Therefore, the MRM consisted of $15 + 16 = 31$ states. The model analysis obtained the following results:

- Percentage of served packets 99.9982%. Therefore, the approximation has been in the fifth significant figure with respect to the real system result.
- The workload remains the 97.518% of the time in the 15 stable states.
- The workload remains the 2.482% of the time in the 16 transient states .
- In the stable states, the system serves the 99.99977% of the packets.
- In transient states, the system serves the 99.93764% of the packets.

These results demonstrate again the importance of modeling the transient periods, when the workload changes but the system is still adapting towards its best configuration. Although only accounting for 2.482% of the time, transient states

are important because they discard packets around 260 times more frequently than the stable states.

8. Discussion of the Approach

This section discusses benefits and limitations of the proposed approach according to six dimensions: the QoS properties to evaluate, the changing traffic, the real-time models, the efficiency of the approach, the effectiveness of the approach, and the upper bound for the workload model.

On the QoS properties to evaluate. The paper has applied the approach to the evaluation of two different QoS properties: availability and packet loss. The availability was interpreted as "readiness for correct service" [3]. The case of packet loss assumed that packets were discarded depending on the saturation of the system. For other possible causes of packet loss, such as network problems, an appropriate system model representing the behaviour of interest should be created. Previously, in [31], we evaluated another QoS property, the system response time. Summarising, our approach is not restricted to a given set of QoS properties, although the engineer needs to build the system model that allows the evaluation of the QoS properties of interest.

On the changing traffic. Our approach accommodates in a single model several different workload intensities and changes between them. When the workload changes, the model does not become imprecise, unless a change in the more general workload pattern happens. Therefore, it is expected that the model remains valid for "long periods" of time. We say "long period" with respect to the time needed to create the model. Since guessing the moment when a complex workload pattern has changed is a difficult task, we suggest that rearranging the model may be a task that could be scheduled periodically (e.g., daily, weekly or monthly).

On real-time models. The creation of the workload models is not immediate. The time needed to create them may be negligible when the rearrangement process is carried out in a daily or weekly basis, as a background task. However, it is too time consuming to execute the full approach during the frequent and real-time process of taking an adaptation decision. As a single workload model is expected to be valid for a long period of time and for several adaptation decision

moments, we position the execution of the generation of the accurate workload model in the typical slow deliberation stage of autonomous systems [17]. Such deliberations produce part of the *Knowledge* that is later used during the execution of the traditional MAPE-K control loop [24].

On the efficiency of the approach. Differently from event driven simulation, here the time required to compute the parameters does not depend on the arrival rate but on the number of time intervals in the trace. In particular,

1. The time to compute the initial MMPP mainly depends on the quantity of time intervals in the trace. However, the size, in terms of number of states, of the MMPP depends on the maximum and minimum values in the trace.
2. For any *width parameter* not lower than 1, as an upper bound, the number of states is lower than the square root of the maximum value. For instance, for a *width parameter* of 2, 1000 states can describe an arrival rate from 1 to 4 million of events per time interval. Therefore, the representation of the MMPP does not create a limiting constraint.
3. The process of reducing the number of states also depends on the number of intervals in the trace.
4. It is necessary a QoS evaluation for each of the states in the MRM created in Section 6.3 -to calculate the reward values of each state- and an additional model evaluation of the MRM itself.

On the effectiveness of the approach. It has been confirmed by the accuracy of the results of the experiments in Section 7. However, factors that may reduce such effectiveness could be the following:

1. The accuracy of the results provided by the model-based analysis/simulation engine, GreatSPN [14] in our case. For states with very low residence time, as one per thousand of time, a rounding error in the third decimal may mean to double the expected residence time.
2. We have found that the MAWI trace contains some entries that represent very abrupt increments that last for extremely little time. Although this property of the trace violates some of the assumptions under which our approach was developed, the results obtained were pretty accurate. For instance, around index 5350, in the trace of Figure 13, we can found that the arrival rate suddenly increments from around 240000 packets every ten

seconds to more than 520000, and then it immediately decreases again to around 240000. This could probably cause that a reactive adaptation engine will not be able to follow the speed of changes in these concrete cases. We believe that providing our model also with an explicit representation of these cases of abrupt variation can further improve the accuracy of the results. However, we leave this study as future work because such possible extensions will entail an increment of the complexity of the model, hence reducing its practical application.

On an upper bound to calculate the workload model. We see at present two upper bounds to calculate the workload model. The first refers to the level of automation.There are still some manual steps, and these are a limitation for the application of the approach. At present, all the steps from the beginning of the execution to the generation of parameters for the models and the states and transitions of the MRM are automated. These steps entail: the creation of the initial MMPP, the creation of the state reduced MMPP, finding the transient periods and the computation of the parameters for the transient models and stable models (e.g., $\lambda_n, \lambda_{n'}, \lambda_{inc}, \lambda_{dec}$, etc.). However, the computation of the reward values that are associated to states in the MRM still needs some manual work. Therefore, the more states in the generated MRM, the more manual work required, being an upper bound on the the amount of manual work that can be afforded. The second upper bound refers to rounding errors in numerical calculations. In transient models where arrival rate values are very high and the transient period relatively slow, the internal numerical analysis of the model performs arithmetical operations with real values whose operands differ in several orders of magnitude. This may cause inaccuracies in the results due to rounding errors during these operations.

9. Related work

Workload modelling and analysis has been widely recognised as a critical part for the design and development of dependable software systems, see, for example, the work on capacity planning by Menasce et al. [28], or the works of Serazzi et al. [6, 26] considering different types of applications, just to cite a few. Besides, it has been observed that the workload, for some kind of systems, is far from being stable but it presents high variability and shows burstiness [30, 2]. Therefore, if the workload model does not account for the existing burstiness, then the model analysis can lead to optimistic results. Research on workload and network

traffic considering the burstiness in the arrival rate used MAPs and MMPPs [16] and their results show an accurate modeling of the workload variability, see for example [21, 25]. In particular, work on fitting MMPP and MAP parameters from workload traces with burstiness is very useful for the analysis of QoS properties of a wide range of systems. The parameter fitting of Markovian models, such as MMPPs, starting from traffic traces is a promising research field where works [20, 22, 29, 34, 10, 11] propose techniques that can be of interest for the adaptive systems field. Some of these fitting works also deal with the modeling of burstiness characteristic. Recent work [9] proposes a generalization of MMPP that also allows representing the arrival process of requests for systems that receive different classes of requests. However, to the best of our knowledge, this aspect has been neglected in the modelling and analysis of adaptive systems. We think that this topic is of great importance for systems deployed on the Internet since it will help to improve their behaviour and overall performance and quality.

Starting from the results in [31], we strove for an extension to deal with more than two stable states, i.e., to advance towards an MMPP(N) modeling. In fact, we knew that if we were able to get such fine grain modeling then the accuracy of the QoS results would be of great quality, which was the main critical issue in our initial work. To this end we exploited the results obtained in [18, 7] to choose the estimators of the workload trace and the algorithm proposed in [21, 25] to fit the MMPP(N). Then, we had to deal with the QoS analysis problems we have described in the paper, which completely reshaped the proposal in [31]. The proposed QoS analysis is able to guarantee that the system meets the requirements in any state -stable or transient-. This is a restriction harder than to meet the requirements considering only a *fully aggregated* model as we have initially done in [31].

10. Conclusion

Current generation of systems deployed on the Internet needs substantial improvement to adequately manage workload issues. Unfortunately, it is very common to hear everyday news about degradations, disruptions or even complete fall down of systems. The situation will aggravate when deployments in the cloud become first class citizens, which will happen soon. Most of these QoS problems arise due to an inappropriate system management of the workload, which in the Internet is bursty and highly variable. Self-adaptive techniques offer a solution for Internet-deployed systems to adequately manage workload issues. However, the very burstiness and variability make the adaptation processes challenging.

In this work we have proposed a model-based evaluation of the QoS of adaptive systems. The characteristics of our approach make it specially suited for Internet deployed systems. Our solution addresses the problems and challenges previously described. Then, we proposed an accurate modeling of the burstiness and variability phenomena that allows identifying the adequate moments for system adaptation. Moreover, we leveraged stochastic models to attain an efficient QoS analysis of the system, which implies the use of Markov Modulated Poisson Processes and Markov reward models.

There are several interesting directions stemming from this work to be investigated. Currently, the transient periods are modeled with linear increments or decrements in the arrival rates. Different behaviors, such as logarithmic or exponential, could be analyzed and compared with the linear ones. Another possible line of research include the consideration of different analysis techniques and their comparison with the Petri nets adopted in this work. We are also working on the implementation of our approach on a real testbed, to assess its effectiveness through a more comprehensive set of real experiments.

Acknowledgments

This work has been partially supported by the European Commission under the H2020 Research and Innovation Action [DICE, Grant Agreement No. 644869], the Spanish Ministry of Economy and Competitiveness [ref. CyCriSec-TIN2014-58457-R], and the Aragonese Government [ref. T94, DIStributed COmputation (DISCO)]"

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley Series in Parallel Computing - Chichester, 1995.

[2] A. Ali-Eldin, O. Seleznjev, S. S. de Luna, J. Tordsson, and E. Elmroth. Measuring cloud workload burstiness. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 566–572, Dec 2014.

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

[4] R. Calinescu, C. Ghezzi, M. Z. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Commun. ACM*, 55(9):69–77, 2012.

[5] R. Calinescu, L. Grunske, M. Z. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic qos management and optimization in service-based systems. *IEEE Trans. Software Eng.*, 37(3):387–409, 2011.

[6] M. Calzarossa and G. Serazzi. Construction and use of multiclass workload models. *Perform. Eval.*, 19(4):341–352, 1994.

[7] G. Casale, N. Mi, L. Cherkasova, and E. Smirni. Dealing with burstiness in multi-tier applications: Models and their parameterization. *IEEE Transactions on Software Engineering*, 38(5):1040–1053, 2012.

[8] G. Casale, N. Mi, and E. Smirni. Model-driven system capacity planning under workload burstiness. *IEEE Transactions on Computers*, 59(1):66–80, 2010.

[9] G. Casale, A. Sansottera, and P. Cremonesi. Compact markov-modulated models for multiclass trace fitting. *European Journal of Operational Research*, pages –, 2016.

[10] G. Casale, E. Z. Zhang, and E. Smirni. Kpc-toolbox: Best recipes for automatic trace fitting using markovian arrival processes. *Perform. Eval.*, 67:873–896, September 2010.

[11] G. Casale, E. Z. Zhang, and E. Smirni. Trace data characterization and fitting for markov modeling. *Perform. Eval.*, 67(2):61–79, Feb. 2010.

[12] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, editors. *Software Engineering for Self-Adaptive Systems [outcome of a Dagstuhl Seminar]*, volume 5525 of *Lecture Notes in Computer Science*. Springer, 2009.

[13] K. Cho, K. Mitsuya, and A. Kato. Traffic data repository at the wide project. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '00, pages 51–51, Berkeley, CA, USA, 2000. USENIX Association.

[14] Dipartamento di informatica, Universita di Torino. GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets, Dec., 2015. URL: www.di.unito.it/~greatspn/index.html.

[15] S. Donatelli and G. Franceschinis. The PSR methodology: Integrating hardware and software models. In J. Billington and W. Reisig, editors, *Application and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 1996.

[16] W. Fischer and K. Meier-Hellstern. The Markov-modulated Poisson process (MMPP) cookbook. *Perform. Eval.*, 18:149–171, September 1993.

[17] E. Gat. Artificial intelligence and mobile robots. chapter Three-layer Architectures, pages 195–210. MIT Press, Cambridge, MA, USA, 1998.

[18] R. Gusella. Characterizing the variability of arrival processes with indexes of dispersion. *Selected Areas in Communications, IEEE Journal on*, 9(2):203 –211, feb 1991.

[19] B. Haverkort and K. Trivedi. Specification techniques for markov reward models. *Discrete Event Dynamic Systems*, 3(2-3):219–247, 1993.

[20] H. Heffes and D. Lucantoni. A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *Selected Areas in Communications, IEEE Journal on*, 4(6):856 – 868, sep 1986.

[21] D. P. Heyman and D. Lucantoni. Modeling multiple IP traffic streams with rate limits. *IEEE/ACM Trans. Netw.*, 11(6):948–958, Dec. 2003.

[22] A. Horváth and M. Telek. Markovian modeling of real data traffic: Heuristic phase type and map fitting of heavy tailed and fractal like samples. In M. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, volume 2459 of *Lecture Notes in Computer Science*, pages 267–282. Springer Berlin / Heidelberg, 2002. 10.1007/3-540-45798-4_17.

[23] R. A. Howard. Dynamic probabilistic systems. vol. 2. semi-markov and decision processes, 1971.

[24] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[25] F. Mallor, P. Mateo, and J. Moler. A comparison between several adjustment models to simulated teletraffic data. *Journal of Statistical Planning and Inference*, 137(12):3939 – 3953, 2007. 5th St. Petersburg Workshop on Simulation, Part II.

[26] P. Manzoni, P. Cremonesi, and G. Serazzi. Workload models of vbr video traffic and their use in resource allocation policies. *IEEE/ACM Trans. Netw.*, 7(3):387–397, 1999.

[27] MAWI Working Group Traffic Archive: Packet traces from WIDE backbone. `http://mawi.wide.ad.jp/mawi/`. 2017.

[28] D. A. Menasce and A. F. A. Virgilio. *Scaling for E Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[29] H. Okamura and T. Dohi. Faster maximum likelihood estimation algorithms for markovian arrival processes. In *Quantitative Evaluation of Systems, 2009. QEST '09. Sixth International Conference on the*, pages 73 –82, sept. 2009.

[30] V. Paxson and S. Floyd. Wide area traffic: The failure of poisson modeling. *IEEE/ACM Trans. Netw.*, 3(3):226–244, June 1995.

[31] D. Perez-Palacin, J. Merseguer, and R. Mirandola. Analysis of bursty workload-aware self-adaptive systems. In *Third joint WOSP/SIPEW international conference on Performance Engineering - ICPE*, pages 75–84, Boston, USA, 05/2012 2012. ACM, ACM.

[32] D. Perez-Palacin, R. Mirandola, and R. Calinescu. Synthesis of adaptation plans for cloud infrastructure with hybrid cost models. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 443–450, Aug 2014.

[33] D. Perez-Palacin, R. Mirandola, and J. Merseguer. Qos and energy management with petri nets: A self-adaptive framework. *Journal of Systems and Software*, 85(12):2796–2811, 2012.

[34] T. Rydén. An em algorithm for estimation in markov-modulated poisson processes. *Comput. Stat. Data Anal.*, 21:431–447, April 1996.

[35] K. S. Trivedi, J. K. Muppala, S. P. Woolet, and B. R. Haverkort. Composite performance and dependability analysis. *Performance Evaluation*, 14(34):197 – 215, 1992.

[36] World Cup 1998 Access logs. `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`. 1998.

Appendix A.

We present here the main steps of the algorithm presented in [21]. It receives as input a trace of counts of requests received in consecutive intervals of time. Let us denote by: $\mathcal{C}$ the trace of counts, as the trace that originated Figure 1; $|\mathcal{C}|$ the number of entries in $\mathcal{C}$, and c_i the count of requests the i-th entry in $\mathcal{C}$ where $i \in [1...|\mathcal{C}|]$. The algorithm first creates the vector of arrival rates Λ and then fits the transition rate values in Q. These two processes are briefly described in the following:

Create a vector of arrival rates: To create the arrival rate in each state and, at the same time, to decide the value for the amount of states N, the algorithm uses the maximum and minimum values in $\mathcal{C}$, called $max(c_i)$ and $min(c_i)$ respectively.

- First, it assigns $\lambda_1 = (\sqrt{1 + max(c_i)} - 1)^2$.
- Then, to create every λ_n, $n > 1$, it iteratively applies the formula $\lambda_n = (\sqrt{\lambda_{n-1}} - a)^2$.
- The algorithm stops right after finding the first λ_n that satisfies $\lambda_n - a\sqrt{\lambda_n} \leq min(c_i)$.

Parameter a has an arbitrary value, e.g., a = 2 in [21]. It represents the width in the number of standard deviations, of the Poisson distribution, for the range of observations that will be associated with each arrival rate. For example, $a = 2$ means that there will be associated to each arrival rate all the observations between its mean plus/minus two standard deviations.

The value of n in the last iteration is assigned to N (i.e., the dimension of the MMPP) and the vector Λ is filled with the calculated λ_n values $1 \leq n \leq N$.

Fit the transition rate values: The fitting procedure in [21] of values in Q assumes that each c_i in the trace can only be produced by one state. Therefore, it exists the function $state(c_i)$ that, given a count of requests c_i ($i \in [1, |\mathcal{C}|]$), returns the state s_n that generated it ($1 \leq n \leq N$).

The behavior of function $state(c_i)$ is the following: $state(c_i) = s_n$ if and only if $((\lambda_n - a\sqrt{\lambda_n}) < c_i) \wedge (c_i \leq (\lambda_n + a\sqrt{\lambda_n}))$.

The univocal assignment of a state to a count of requests under this rule is possible because the algorithm followed to create λ_n values of the MMPP ensured that:

$$\forall_{n \in [1..N]}, \nexists\, n' \neq n \;\mid\; (\sqrt{\lambda_n} - a)^2 < \lambda_{n'} < (\sqrt{\lambda_n} + a)^2$$

Using this $state(c_i)$ function, values $q_{nn',\ n \neq n'}$ in Q are calculated as the probability of being the count of requests in position $i+1$ produced by state $s_{n'}$ given that the count of requests in position i has been produced by state s_n (i.e., formally $P(state(c_{i+1}) = s_{n'} \mid state(c_i) = s_n)$).

Finally, values q_{nn} are calculated as $q_{nn} = \sum_{n':n' \neq n} q_{nn'}$.

Appendix B.

We illustrate here the algorithm that calculates the mean time for the workload to change from a stable state s_n to another stable state $s_{n'}$ (case in which $\lambda_n < \lambda_{n'}$).

- **First step:** Data int workload trace with counts of requests $\mathcal{C}$ is traversed to find all intervals of increment from λ_n to $\lambda_{n'}$. Each interval is defined by its bounding positions in $\mathcal{C}$: $[init, end]$. Bound values *init* and *end* must satisfy that:

$$(c_{init-1} < \lambda_n) \;\wedge\; (c_{end+1} > \lambda_{n'})$$
$$\wedge$$
$$\nexists\, k \in [init, end] \;\mid\; c_k < \lambda_n \;\vee\; c_k > \lambda_{n'}$$

- **Second step:** In order to calculate a meaningful value that represents the mean time for the workload to change between values λ_n and $\lambda_{n'}$, we need to identify among the intervals found in the previous step in $\mathcal{C}$ those ones that are "real increments" in the workload trace. As "real increment" we

mean the intervals that show continuous increment in the long-term view. The reason to identify the "real increments" is that the workload trace $\mathcal{C}$ may contain different changing behaviors between λ_n and $\lambda_{n'}$, two examples of changing behaviors between values λ_n and $\lambda_{n'}$ are shown in Figure B.14. We want to ignore the intervals whose changing behavior is like the one depicted in Figure B.14(a) because they do not really represent a direct change between values λ_n and $\lambda_{n'}$ and could jeopardize the study, and keep the intervals whose changing behavior is like the one in Figure B.14(b). Hence, we decide to keep the intervals that show continuous increment in the workload and ignore those that contain some arrival rate decrements. However, this distinction is not straightforward to implement due to the presence of short-term variability in all the workload trace $\mathcal{C}$, which makes pinpointing continuous increments more difficult. Therefore, to address this step, the short-term variability should be temporarily filtered out in order to decide whether an interval shows continuous increment. The algorithm below in this Appendix addresses this issue.

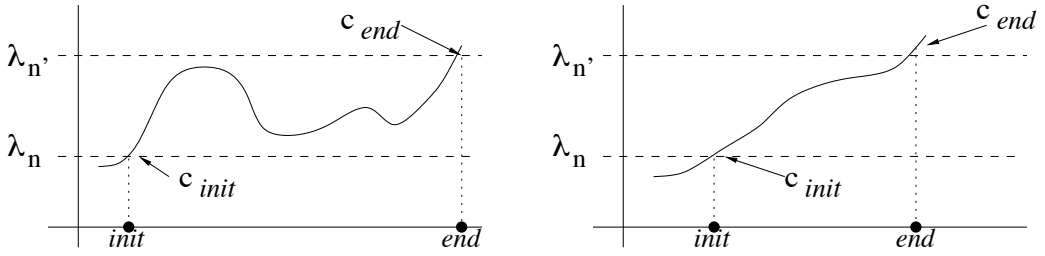

Figure B.14: Examples of workload increment

- **Third step.** For each interval that was kept in the previous step, calculate its length as $end - init$. Then, calculate the mean of these lengths and assign such value to $mt_{nn'}$.

Algorithm for deciding "real increments" in the workload

Given an interval $[init, end]$, the algorithm decides whether it represents a "real increment" of the workload. The basis of the algorithm is to check whether the workload is continuously increasing in the long-term within the interval.

- *First step.* To filter out short-term variability, we add the number of requests received during L consecutive periods into a single value. So, the i-th aggregated value, where $i \in \{0, ..., \lfloor \frac{end-init}{L} \rfloor\}$, will be $\sum_{l=0}^{L-1} c_{init+iL+l}$.

  L is a user's choice and represents how much we filter brief variations: a too low L would not filter short-term variability, while a too high L may recognize as continuous increments intervals that should be ignored.

- *Second step.* To check whether the workload is continuously increasing, we verify that

$$\forall_{i \in \{0,...,\lfloor \frac{end-init}{L} \rfloor - 1\}} \qquad \sum_{l=0}^{L-1} c_{init+iL+l} < \sum_{l=0}^{L-1} c_{init+(i+1)L+l}$$

Appendix C.

Figure C.15 depicts examples of stochastic models using GSPN. Part (a) models a workload increment, M_{wk}, according to Section 5.1. Requests are represented by tokens in places $pSystemInput$ and $pSystemInput'$, the first used by $M_{sys}(r_n)$ and the latter by M_{adapt}. Part (b) models the system in Section 2 when using r_n resources. Part (c) models a simple rule-based adaptation logic. The logic adds resources as a function of the arrival rate during the last time interval of length $t^{-1}_{Interval}$. Concretely, if the number of requests is higher than x_1, then it adds up to res_1 resources; if it is higher than x_2, then up to res_2; and if it is higher than x_3, then up to res_3. Methods to calculate suitable res_i and x_i values in this case have already been proposed [32, 33]. Finally, M_{adapt} also models, through transition t_{setup}, the time required to boot resources -servers and application- before they are ready to serve requests.

Figure C.16 depicts the aggregation of the three models in Figure C.15, composing them by places `pSystemInput`, `pSystemInput'` and `pResources`. This Petri net depicts a cycle of workload increment and adaptation. We decided to obtain the QoS results by using a steady-state analysis of the Petri net. So, we tuned the model to behave as a regenerative process of the cycle. We then added an immediate transition that fires when the cycle is finished for creating the initial marking again.

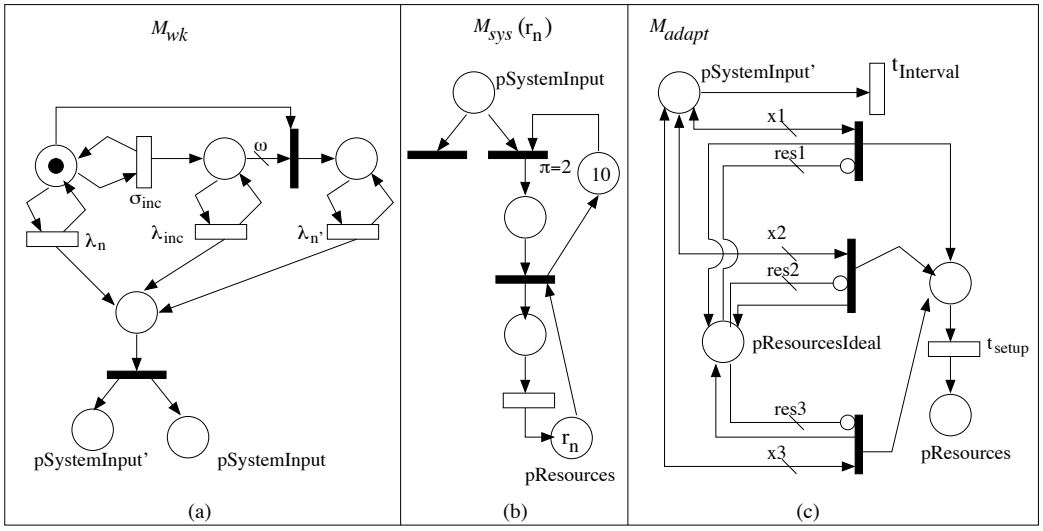

Figure C.15: Examples of GSPN models: (a) M_{wk}, (b) $M_{sys}(r_n)$ and (c) M_{adapt}

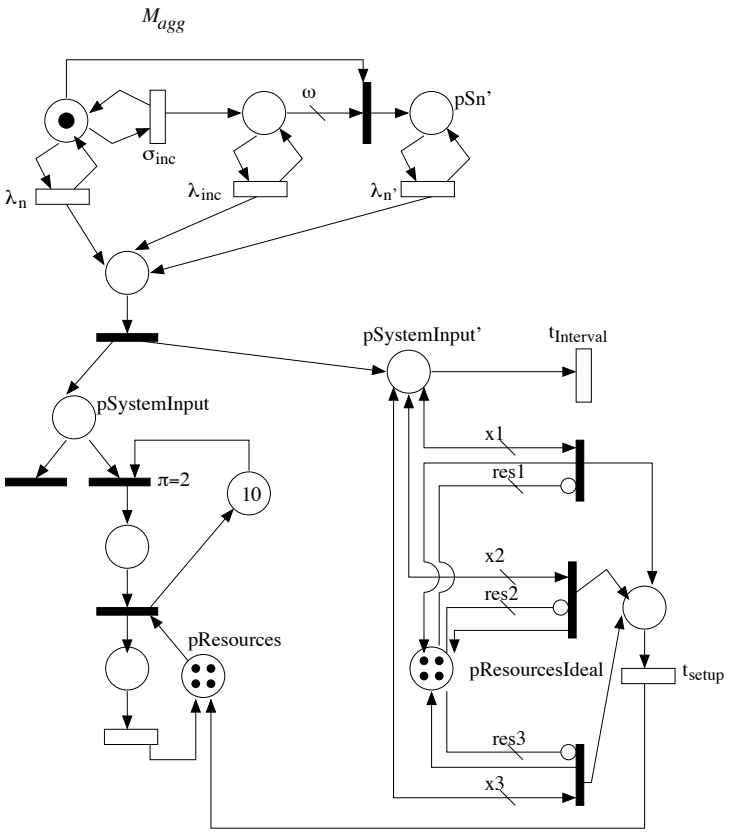

Figure C.16: GSPN aggregated model, where $r_n = 4$