Time Performance Formal Evaluation of Complex Systems

Valdivino Alexandre de Santiago Júnior[1] and Sofiène Tahar[2]

[1] Instituto Nacional de Pesquisas Espaciais (INPE)
Av. dos Astronautas, 1758, São José dos Campos, São Paulo, SP, Brazil.
valdivino.santiago@inpe.br
[2] Concordia University
1455 De Maisonneuve Blvd. West, Montreal, QC, Canada.
tahar@ece.concordia.ca

Abstract. Formal verification methods, such as Model Checking, have been used for addressing performance/dependability analysis of systems. Such formal methods have several advantages over traditional techniques aiming at performance/dependability analysis such as the use of a single computational technique for evaluation of any measure and all complex numerical computation steps are hidden to the user. This paper reports on the use of Probabilistic Model Checking for time performance evaluation of complex systems. We propose an approach, TPerP, that allows a professional to clearly address time performance analysis based on Continuous-Time Markov Chain (CTMC). Our approach takes into consideration several types of delay analyzes. We applied it to a balloon-borne high energy astrophysics scientific experiment where we dealt with CTMCs that had around 30 million reachable states and 75 million transitions, and we concluded that the current definition used in the balloon system is inadequate in terms of performance.

1 Introduction

Studies about performance evaluation of systems date back to the early 1900s where single queues, Markov Chains, networks of queues and Stochastic Petri Nets have been used for this purpose. Particularly, Markov Chains have been applied to performance assessment since around 1950 [1].

Performance evaluation is thus a mature field. However, formal verification methods, such as Model Checking and Theorem Proving, have also been used for addressing performance/dependability analysis of systems. Such formal methods have several advantages over traditional techniques (e.g. simulation) aiming at performance/dependability analysis. For instance, temporal logic offers a high degree of expressiveness and flexibility where most standard performance measures (e.g. transient probabilities, long-run likelihoods) can be easily expressed. Moreover, it is possible to specify complex measures in a succinct way by nesting temporal logic formulas [2].

Model Checking is a fully algorithmic approach towards performance evaluation where a single computational technique is used for assessment of any

possible measure, and its time and space complexity is attractive. In the worst case scenario, the time complexity is linear in the size of the measure specification (logic formula), and polynomial (order 2 or 3, at most) in the number of states of the stochastic process. Regarding space complexity, in the worst case, it is quadratic considering the number of states of the stochastic process [2, 3]. Not less important, especially for practitioners, using Model Checking for performance evaluation is interesting because all algorithmic/implementation details, all detailed and complex numerical computation steps are hidden to the user.

In this paper, we report on the use of Probabilistic Model Checking for time performance evaluation of complex systems. We organized the activities accomplished in this work on an approach, **Time Performance Evaluation via Probabilistic Model Checking** (TPerP), that allows a professional to clearly address time performance analysis based on Continuous-Time Markov Chain (CTMC) and Probabilistic Model Checking [3–6]. Even though TPerP is based on standard steps defined for Model Checking, it takes into consideration several types of delay analyzes and provides directives so that industry professionals may use it for the development of real and complex systems/software. We applied it to a balloon-borne high energy astrophysics experiment under development at the *Instituto Nacional de Pesquisas Espaciais* (INPE) in Brazil. We dealt with CTMCs that had around 30 million reachable states and 75 million transitions and thus Probabilistic Model Checking confirmed to be a suitable solution for the performance analysis of complex systems. We concluded that the current definition used in the balloon system is inadequate in terms of performance.

This paper is structured as follows. Section 2 presents our approach, TPerP. Section 3 shows the characterization of the problem providing details about the case study. Results and considerations by applying TPerP to the space system are in Section 4. Section 5 presents related work. In Section 6, conclusions and future directions are mentioned.

2 The TPerP Approach

TPerP takes advantage of Probabilistic Model Checking and CTMC to assist in time performance analysis of complex system/software projects. The TPerP approach is shown in Figure 1.

The first step that should be accomplished is to define the parameters to be addressed. These parameters are variables that affect the time performance of the system. TPerP aims at finding the most suitable/optimal values of such parameters. Some examples of these variables are the amount of packages a buffer of a hardware device must store, the size of a packet to be sent from one equipment to another, and the number of sensor measurements to be sent to a central management computer. All these parameters are considered taking into account the time performance perspective.

In computer networking or telecommunications, there are various types of delays (d_i). TPerP considers the following ones: (i) Propagation Delay: $PD = l/s$. It is the ratio between the link length (l) and the propagation speed (s)

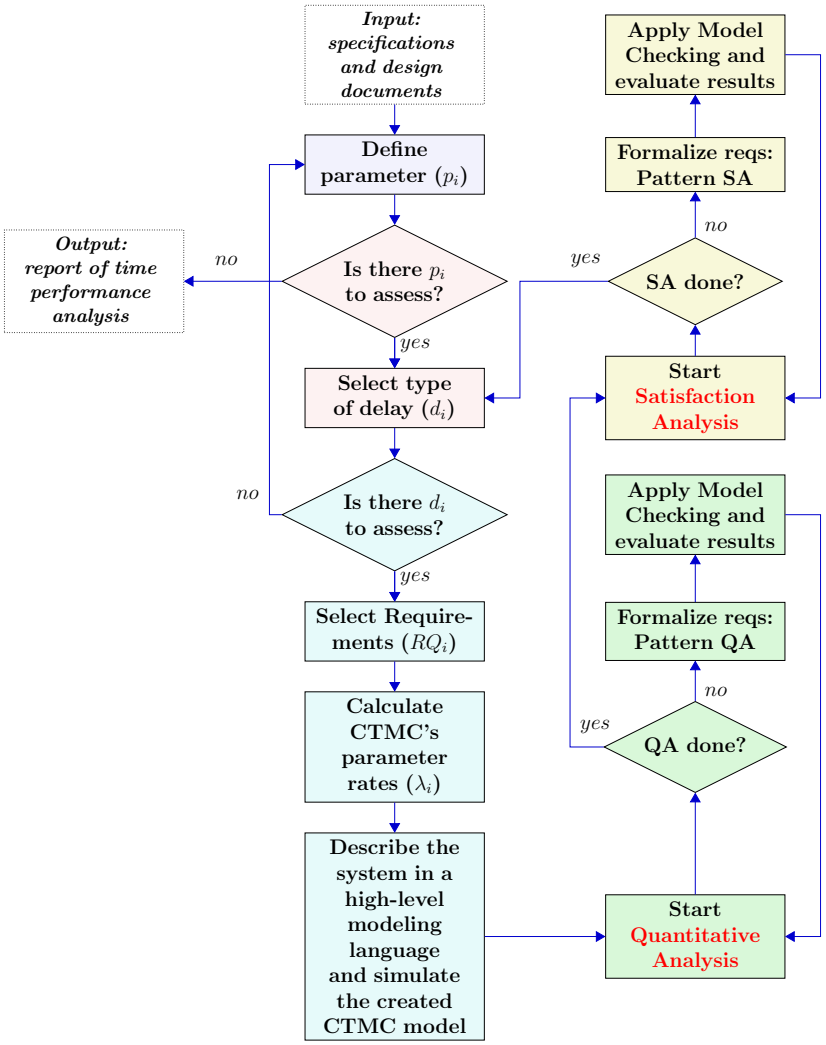

Fig. 1. The TPerP approach. Caption: QA = Quantitative Analysis; SA = Satisfaction Analysis

over the specific medium. In wireless communication, the speed is the light's propagation speed; (ii) Transmission Delay: $TxD = PDU\_size/bit\_rate$. It is the amount of time from the beginning until the end of a message transmission. It depends on the size of the Protocol Data Unit (PDU) and the speed set for the physical transmission medium; and (iii) Time for PDU Generation: TPG. This is the time needed to create/mount a PDU such as as packet or a frame. It basically depends on the requirements for generating system's data.

It is possible that one d_i can be indeed a combination of two or all of the delays defined above. In addition to these, other delays such as queue processing delays can be addressed. Once a type of delay is chosen, TPerP suggests to select the quantitative requirements based on the system/software specifications.

The next step is to calculate CTMC's parameter rates, λ_i. There should be defined as many λ_i as the amount of relevant flows of data transmission between communicating units. Besides, the calculation of λ_i should be done considering the piece of information that is transmitted per unit of time resolution. Although other ways to calculate the parameter rates are possible, we would like to emphasize two of them:

a) Local PDU influence.

$$\lambda_i = \frac{bit\_rate}{PDU\_size \times time\_res}. \tag{1}$$

In this case, λ_i is calculated simply due to the size of the PDU of the local flow of data transmission;

b) Diverse processing delays and encapsulation influence.

$$\lambda_i = \frac{bit\_rate}{\left(PDU\_size + \left(\sum_j pd_j + \sum_k en_k\right) \times bit\_rate\right) \times time\_res}. \tag{2}$$

Here, $\sum_j pd_j$ and $\sum_k en_k$ represent, respectively, the influence of delays such as queue processing and the impact of other sizes mainly related to the encapsulation features of network architecture models such as Open Systems Interconnection (OSI) and The Consultative Committee for Space Data Systems (CCSDS). Note that in order to use Equation 2, it is necessary to consider at least one influence (processing delays or encapsulation) or both.

Let us assume that a certain flow of data transmission between two computing devices has the following characteristics: PDU size = 65,535 Bytes, bit rate = 1 Gbps, queuing and end system delays (encoding, decoding) = 50 ms. Let us also assume that a encoding system is used where each Byte of data is assigned a 10-bit code like the 8b/10b encoding used in Gigabit Ethernet. In addition, let us consider that a certain requirement, RQ_i, demands that the system must meet a time bound in the range of milliseconds. Thus, the parameter rate is in accordance with Equation 2:

$$\lambda = \frac{10^9}{(65535 \times 10 + 0.05 \times 10^9) \times 1000} = 0.019741. \tag{3}$$

In Equation 3, λ means that, for each millisecond, 0.019741 of a PDU is received by the destination device.

Describing the system in a high-level language supported by a Probabilistic Model Checker such as PRISM [4] is the next step. TPerP suggests the traditional procedure where CTMCs are not obtained in a straightforward way, but rather generated automatically from higher-level descriptions in languages such as stochastic Petri nets, stochastic process algebras [5], or state-based languages [4]. This description is read into a Stochastic/Probabilistic Model Checker which then automatically creates a CTMC based on it. Besides, it is very important to simulate the CTMC model thoroughly in order to avoid an excessive number of false positive counterexamples. The simulation should be carried out by means of the features (outputs) provided by Model Checkers, for example, checking if there are transitions between states of the CTMC model consistent with the expected behavior of the system, and according to the defined parameter rates.

TPerP provides some guidelines to develop the description of the system. In Model Checkers that allow synchronization, our approach suggests using it as much as possible so that modules can make transitions simultaneously. Using auxiliary variables, such as boolean ones, to indicate the end of certain processing is not advisable. Such variables increase the state space and, for the purpose of time performance analysis modeling, they may be replaced by synchronization. For instance, in PRISM, some parts of the modules may be as shown below.

```
module device1
...
[action1] var1 = max_var1 -> rate1 : (var1' = 0);
...

module device2
...
[action1] var2 < max_var2 -> 1 : (var2' = var2 + 1);
...
```

We can see that *action1* synchronizes *device1* and *device2* avoiding the use of a boolean variable to state that a certain PDU is ready to be transmitted from one device to another.

2.1 Quantitative Analysis

Probabilistic Property Specification Templates (ProProST) is a specification patterns system for probabilistic properties as they are used for quality requirements [7]. ProProST complements and extends existing patterns systems [8, 9], in that it allows to specify probabilistic properties as they are required to formulate quality properties. ProProST provides a solution in the form of a formal specification template for Continuous Stochastic Logic (CSL) [3].

Specification patterns systems are very important for Model Checking real world applications since they provide templates/guidelines so that professionals can use them in order to formalize their requirements/properties. Based on this fact, TPerP directs the practitioner to use the *Probabilistic Until* pattern of ProPoST [7] for CSL:

$$\mathcal{P}_{\bowtie p}[\Phi_1 \ \mathcal{U}^{[t_1,t_2]} \ \Phi_2]. \tag{4}$$

In TPerP, we call this Pattern for Quantitative Analysis (*Pattern QA* in Figure 1) and it is presented in Equation 5:

$$\overline{(\alpha, \mathcal{P}_{=?}[pdu\_notsent \ \mathcal{U}^{\leq T} \ pdu\_received])}. \tag{5}$$

However, there are three remarks if we compare ProProST's formula (Equation 4) with TPerP's formula (Equation 5). First, some Model Checkers such as PRISM allow to specify properties in a quantitative way. In other words, rather than just verifying and answering true or false whether or not a probability is above or below a certain bound, such tools allow to return the actual probability of some behavior of the model under consideration. For time performance evaluation this is very suitable because in many situations we are interested in knowing what is the optimal value of a parameter p_i based on the time constraints defined in the requirements. If we know the exact value of a probability, we are able to respond more adequately this question. TPerP suggests using the $\mathcal{P}$ operator in this way: $\mathcal{P}_{=?}$. As a result, we will have the value of the probability for the paths starting in the initial state ι.

Naturally, several PDUs may be transmitted from one device to another. Therefore, it is interesting when accomplishing a time performance evaluation to consider a fine-grained analysis, i.e. to take into account α states from where to start the analysis instead of doing it from the initial state ι. Thus, α in TPerP's formula (Equation 5) means precisely to start the analysis from the states that indicate that a PDU_k is ready to be sent from the source device but which has not yet been transmitted to the destination device. Thus, TPerP takes the average values of probabilities for the paths that satisfy the path formula $[pdu\_notsent \ \mathcal{U}^{\leq T} \ pdu\_received]$, such paths start in α states.

The third remark is about the time interval $[t_1, t_2]$. By default, TPerP instantiates this time interval as $[0, T]$. But it is perfectly possible to change the lower bound from 0 to other real number just assuring that $t_1 \leq t_2$.

Let us consider the following requirement from the automotive industry [10]: *If the system's diagnostic request IRTest is set, then the infrared lamps are turned on after at most 10 seconds.* Thus, a possible physical architecture has an Electronic Control Unit (ECU_{diag}) responsible for ordering the diagnostic request and other unit (ECU_{lamp}) which indeed turns the infrared lamps on. So, communication could take place via one of the protocols used in the automotive industry such as the Controller Area Network (CAN). Hence, the formalization of the requirement is as follows:

$$\overline{(\alpha, \mathcal{P}_{=?}[request\_notsent\_k \ \mathcal{U}^{\leq 10} \ request\_received - lampson\_k])}. \tag{6}$$

In Equation 6, the state formula $request\_notsent\_k$ means that the kth request (e.g. a CAN frame or a PDU of a CAN-based higher-layer protocol) is ready, in the ECU_{diag}, to be sent to the ECU_{lamp} but has not yet been transmitted. The state formula $request\_received - lampson\_k$ means that not only this kth request has been transmitted and received by ECU_{lamp} but also that the ECU_{lamp} turned the lamps on. The state formulas are usually characterized

by means of model variables that indicate the states in which the PDU has not been sent (*notsent*) and that the same PDU has been received (*received*).

After applying Model Checking and evaluating the results, it should be decided whether the quantitative analysis must be ended or not. It is very beneficial that "new" requirements are created by modifying (usually to a larger value) the time constraints defined in the original specifications' requirements. This activity is very useful to find out how inadequate is a particular solution of a system in terms of performance. Therefore, the quantitative analysis should be repeated for each new defined requirement (new time value).

2.2 Satisfaction Analysis

The satisfaction analysis proposed by TPerP is in line with the traditional verification way of properties assessment in Model Checking. This step is particularly suited to realize whether an existing system/software solution is in accordance with system/software time requirements. If a solution does not satisfy a time requirement, thus we can determine which are the necessary modifications in the solution to achieve this goal.

Again the *Probabilistic Until* pattern of ProPoST [7] for CSL is used, starting the analysis from α states as explained in the previous section. In TPerP, we denote this Pattern for Satisfaction Analysis (*Pattern SA* in Figure 1) and it is given by:

$$\exists(\alpha, \mathcal{P}_{\geq bound}[pdu\_notsent \ \ \mathcal{U}^{\leq T} \ \ pdu\_received]). \tag{7}$$

A note should be made on the $\exists$ quantifier in this TPerP's formula (Equation 7). We took advantage of Model Checkers such as PRISM that allow to reason whether there are some paths that satisfy the path formula $[pdu\_notsent \ \ \mathcal{U}^{\leq T} \ \ pdu\_received]$, such paths start in α states, with a probability greater than or equal to a probability *bound*.

It is important to stress that, by using $\exists$, we are not going into the formal details of the qualitative fragment of Probabilistic Computation Tree Logic (PCTL) or CSL. Path quantifiers, $\forall$ and $\exists$, present in the syntax of CTL were replaced by the probabilistic operator $\mathcal{P}_{bound}$ in PCTL and consequently in CSL. We just relied on the benefits of some tools that enable us to reason about a fine-grained time performance analysis.

Let us consider again the real-time requirement from the automotive industry [10] presented in Section 2.1. Its formalization in accordance with *Pattern SA* can be:

$$\exists(\alpha, \mathcal{P}_{\geq 0.98}[request\_notsent\_k \ \ \mathcal{U}^{\leq 10} \ \ request\_received - lampson\_k]). \tag{8}$$

Similarly to the quantitative analysis, it should be decided whether the satisfaction analysis continues or not, in accordance with the same reasoning of creating new requirements by varying time. Likewise, the satisfaction analysis should be repeated for each new defined requirement.

3 Case Study: Characterization of the Problem

protoMIRAX is a hard X-ray imaging telescope [11] which has been developed at the *Instituto Nacional de Pesquisas Espaciais* (INPE) in Brazil. This scientific experiment will be launched by a balloon and it will operate between 40 to 42 km of altitude. The main objective of protoMIRAX is to carry out imaging spectroscopy of selected bright X-ray sources to demonstrate the performance of a prototype of the MIRAX satellite's instrument. A very simplified view of the protoMIRAX's physical architecture is shown in Figure 2.

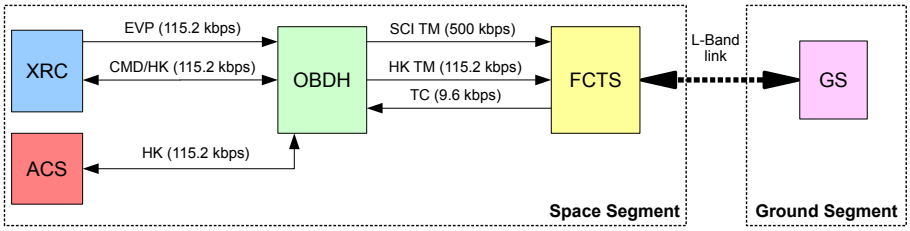

Fig. 2. Simplified physical architecture of the protoMIRAX balloon experiment

The On-Board Data Handling Subsystem (OBDH) is responsible for acquiring, formatting and transmitting all TeleMetry (TM) data that come from several subsystems (X-ray Camera (XRC), Attitude Control Subsystem (ACS)) of protoMIRAX's space segment to the Ground Station (GS). The OBDH is also responsible for receiving and retransmitting, when necessary, various types of TeleCommands (TCs) sent by the GS to the space segment. For each X-ray photon detected by the XRC, a 6-Byte packet is created encasing the time stamp, $x-y$ position, and energy (pulse height) of the event (X-ray photon). This 6-Byte unit of information is called an *event packet* (EVP) and such event packets are sent every 1 second to the OBDH by means of a 115.2 kbps RS-422 unidirectional serial interface. XRC generates 40 event packets/s.

In order to transmit scientific data TM (SCI TM) to ground, a 500 kbps synchronous channel connects the OBDH with the Flight Control and Telecommunications Subsystem (FCTS). Housekeeping data TM (HK TM) are sent to the GS via a serial RS-232 channel operating at 115.2 kbps.

The total size of TC and TM (focus of this research) packets are variable and defined according to each space mission. The problem that needs to be solved by this research is as follows. As the OBDH continuously stores on-board and sends to the GS all TM data (scientific, housekeeping, etc.) it obtains, what is the most suitable (optimal) size of the scientific data TM packet, with event packets created by the XRC and sent to the OBDH, so that we have the minimum delay of such scientific data stored on-board and the same data that are visualized on the GS in real-time? In previous INPE's balloon scientific instrument, this delay was in the range of hour. A suitable performance analysis was not accomplished for this previous project.

4 Application of TPerP

We applied TPerP to the protoMIRAX instrument. We defined one parameter, p_1, which is the size of the scientific data TM packet. Thus, we would like to know what is the most suitable (optimal) value of p_1. We will denote this optimal value as OP_EVP. XRC generates 40 event packets/s and transmits to the OBDH which formats and generates a scientific data TM packet to be sent to the GS. OP_EVP is a value which ranges from 1 to 108. In other words: (i) Minimum size of TM packet ($OP\_EVP = 1$; Size in Bytes = 274). This implies 1×40 event packets. Once the OBDH receives the 40 event packets in each second, it formats, stores the data in its memory, and transmits them straight to the GS; and (ii) Maximum size of TM packet ($OP\_EVP = 108$; Size in Bytes = 26,596). This implies 108×40 event packets. In other words, the OBDH waits for the arrival of 108 units of 40 event packets and, after that, the OBDH sends such data to the GS.

4.1 Transmission Delay

We chose two types of delays where the Transmission Delay (TxD) was the first one. Thus, one performance requirement was taken into account: RQ1 - *The delay between the scientific data TM packet stored in the OBDH's computer memory and the same data received by the GS must be, at maximum, 500 milliseconds.*

Four parameter rates were considered according to 4 flows of data transmission, and all such rates were calculated taken into account a resolution in ms: (i) $\lambda_{evp} = EVP\_rate/1000 = 0.04$. This is the XRC's event packets generation rate; (ii) $\lambda_{xrc\_sci} = (bit\_rate_{x-o})/(PDU\_size_x \times 1000) = 0.045714$. This rate relates to the transmission of a unit (40) of event packets from the XRC to the OBDH. Note that it is a local PDU influence rate (Equation 1); (iii) $\lambda_{obdh\_sci} = (bit\_rate_{o-g})/((PDU\_size_o + (qd + od) \times bit\_rate_{o-g}) \times 1000)$. This rate relates to the transmission of scientific data (event packets), after being completely formatted, from the OBDH to the GS. It is a diverse processing delay rate (Equation 2) where qd it is the queue processing delay within the software embedded into the OBDH's computer, and od refers to other delays due to further required processing. Note that the size of the OBDH's PDU (PDU_size_o) is directly proportional to OP_EVP. Hence, for $OP\_EVP = 1$, $\lambda_{obdh\_sci} = 0.003285$, and for $OP\_EVP = 108$, $\lambda_{obdh\_sci} = 0.001378$; and (iv) $\lambda_{all\_hk} = (bit\_rate_{hk-o-g})/((PDU\_size_h + (qd + od) \times bit\_rate_{hk-o-g}) \times 1000) = 0.001525$. This rate relates to the transmission of housekeeping data from several subsystems (XRC, OBDH, ACS) to the OBDH which then sends housekeeping information to the GS. It is also a diverse processing delay rate.

We used PRISM and hence we described our system using its language, and simulated the behavior of the CTMC model. Our models range from 546,530 reachable states and 1,647,070 transitions ($OP\_EVP = 1$) to **29,785,885 reachable states and 75,502,215 transitions** ($OP\_EVP = 108$). After realizing that the CTMC model truly reflects our system, we formalized RQ1 as

proposed by TPerP:

$$\overline{(\alpha, \mathcal{P}_{=?}[onboard\_TM\_k \ \ \mathcal{U}^{\leq 500} \ \ ground\_TM\_k])}. \tag{9}$$

The state formula $onboard\_TM\_k$ means that the kth scientific data TM packet is formatted and ready, in the OBDH, but has not yet been transmitted to the GS while the state formula $ground\_TM\_k$ means that such kth packet has been transmitted and received by the GS. As explained in Section 2.1, α are the states from where to start the analysis because they represent the situations where PDU_k is ready to be sent from the OBDH but such PDU has not yet been sent and received by the GS.

We did several experiments and created several graphics varying the time in accordance with $0 \leq t \leq$ T, where T = 500 ms. Due to space restrictions we will not show them. The current solution defined for the protoMIRAX system is $OP\_EVP = 1$ (minimum). Analyzing the results of the Model Checking, we noticed that such a solution is not a good option because the average value of the probability is too low (0.3316) when T = 500 ms.

As OP_EVP increases, we could see a significant improvement on the average value of the probability when T = 500 ms. However, using a large value of OP_EVP is not the best solution. When $OP\_EVP = 108$ (maximum), the average value of the probability for T = 500 ms is only 0.4924. Figure 3 shows the mean values of probabilities for T = 500 ms for all possible values of OP_EVP. We perceive that there is a set of optimal values: $12 \leq OP\_EVP \leq 19$. The highest mean probability is due to $OP\_EVP = 15$ (0.6954).

We continued the quantitative analysis in order to find out how unsuitable was the current solution ($OP\_EVP = 1$). Thus, we changed RQ1 and created a new requirement where the time is now 1 hour. We noticed a small improvement but the average value of the probability reaches a limit still too low (0.3581). Importantly, the result of this analysis does not claim that the scientific data will last one hour, or even more, to reach ground (GS). The maximum value of the probability for $OP\_EVP = 1$ is, in fact, 0.8065. However, a low mean value of probability means that, on average, significant delays may occur with greater probability when the operation of the protoMIRAX system.

Regarding the satisfaction analysis, we accomplished it in order to answer this question: given the current characteristics of the protoMIRAX system (packet sizes, communication rates, etc.) is RQ1 satisfied? The previous quantitative analysis has provided an indication of what value, or interval, of OP_EVP would be the most appropriate. Such analysis also suggests that the current solution, $OP\_EVP = 1$, is inappropriate. But nothing was said concerned the satisfaction of RQ1. We formalized RQ1 in accordance with TPerP:

$$\exists(\alpha, \mathcal{P}_{\geq 0.98}[onboard\_TM\_k \ \ \mathcal{U}^{\leq 500} \ \ ground\_TM\_k]). \tag{10}$$

In Table 1, we see that RQ1 is NOT satisfied in accordance with the current characteristics of the protoMIRAX experiment. No value of OP_EVP was such that the CTMC model satisfied RQ1 (T = 0.5 s). Note that the current characteristics of the protoMIRAX system (packet sizes, communication rates) only

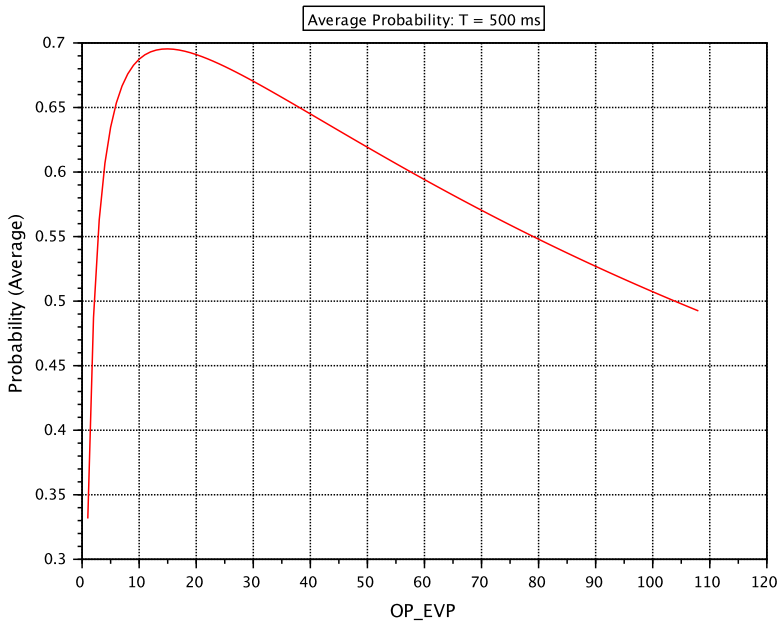

Fig. 3. TxD: Average values of probabilities considering T = 500 ms (all OP_EVP)

begins to fulfill the time performance requirements from T $\geq$ 2 s but, even so, for some values of OP_EVP (10, 13, 15, 17, and 30). It is important to stress that neither for the minimum (1) nor for the maximum (108) value of OP_EVP the requirement is satisfied when T = 2 s. In addition, the minimum value (1) and current solution does not satisfy the requirements even if we consider T = 1 day (86,400 s). This is another result that corroborates the previous conclusion: current solution ($OP\_EVP = 1$) is inadequate.

Table 1. TxD: Satisfaction of time performance requirements

OP_EVP	**Time (seconds)**					
	0.5	**1**	**2**	**5**	**10**	**86,400**
1	✗	✗	✗	✗	✗	✗
10	✗	✗	✓	✓	✓	✓
13	✗	✗	✓	✓	✓	✓
15	✗	✗	✓	✓	✓	✓
17	✗	✗	✓	✓	✓	✓
30	✗	✗	✓	✓	✓	✓
108	✗	✗	✗	✓	✓	✓

4.2 Total Time

For protoMIRAX, the Propagation Delay is neglected due to the small distance of operation of the balloon compared to the light's propagation speed. However, it is interesting to consider the Time for PDU Generation (TPG). For this system, such a time is basically defined as a function of OP_EVP. That is, if $OP\_EVP = 10$ thus 1 scientific data TM packet will be ready every 10 seconds (approximately) to be sent to ground. We considered the Total Time which is based on TPG and TxD: $Total\_Time = TPG + TxD$.

We repeated the process suggested by TPerP for this new delay. The requirement is basically the same as previously proposed but with T = 30 s. Since TPG is at least one second, there is no sense in demanding the system to meet a requirement in the range of milliseconds. The semantics to create the CTMC did not change but the parameter rates did (λ_i calculated with resolution in s): (i) $\lambda_{evp} = EVP\_rate = 40$; (ii) $\lambda_{xrc\_sci} = (bit\_rate_{x-o})/(PDU\_size_x + (OP\_EVP \times bit\_rate_{x-o}))$. Note the influence of the encapsulation feature (Equation 2) presented in the calculation of this rate due to its dependency on OP_EVP; (iii) $\lambda_{obdh\_sci} = (bit\_rate_{o-g})/(PDU\_size_o + (qd + od + OP\_EVP) \times bit\_rate_{o-g})$. In this case, we have both dependencies: processing delays and encapsulation; and (iv) $\lambda_{all\_hk} = (bit\_rate_{hk-o-g})/(PDU\_size_h + (qd + od + OP\_EVP) \times bit\_rate_{hk-o-g})$. Again, diverse processing delays and encapsulation influence were used.

Analyzing the results of the Model Checking where $0 \leq t \leq$ T, T = 30s, we realized that, initially, $OP\_EVP = 1$ had a better performance compared with the other values. This is explained by the lower TPG when $OP\_EVP = 1$. However, it was evident that the average value of the probability when $OP\_EVP = 1$ reaches again a low limit (0.6057). The interval $13 \leq OP\_EVP \leq 17$ is a good option, although $OP\_EVP = 10$ was the value which had the highest average value of probability (0.8103). In Figure 4, we see more clearly the mean values of probabilities considering all values of OP_EVP.

Concerning the satisfaction analysis, the protoMIRAX system also did NOT satisfy the requirement initially proposed (T = 30 s). No value of OP_EVP given in Table 1 was such that the requirement could be satisfied. Increasing T to 60 s, then $OP\_EVP = 10$ and $OP\_EVP = 13$ were the only ones to meet the requirement and thus none of the other values, including the minimum ($OP\_EVP = 1$), satisfied the property for T = 60 s.

4.3 Considerations about the Evaluation conducted via TPerP

Based on the time performance analysis accomplished via TPerP, we can conclude: (i) the current solution adopted in the protoMIRAX system, $OP\_EVP = 1$, is inadequate. This conclusion is valid not only if we consider the Transmission Delay but also the Total Time; (ii) the interval $13 \leq OP\_EVP \leq 17$ is a good alternative to solve this performance issue. As a first option, we suggest $OP\_EVP = 15$ (average value of the interval) to be used in the protoMIRAX system. It means a TM packet with 3,718 Bytes. This value is particularly suited

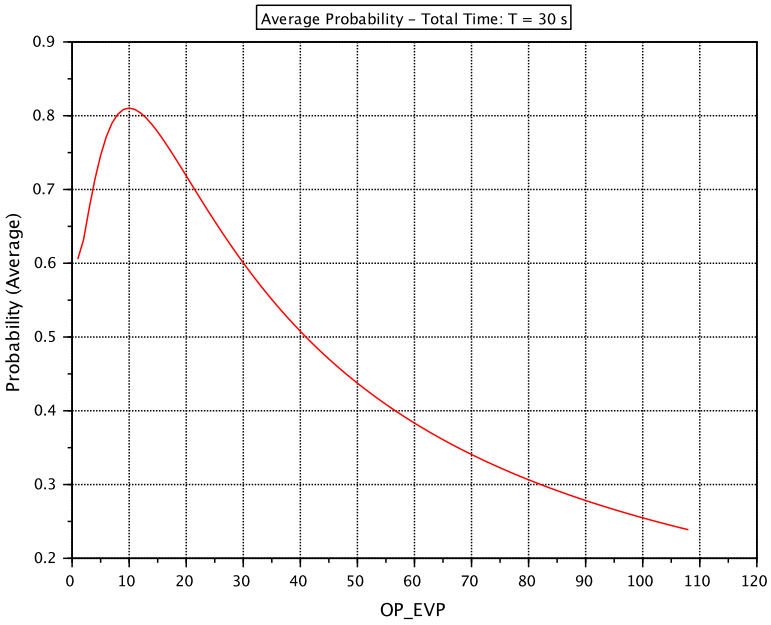

Fig. 4. Total Time: Average values of probabilities considering T = 30 s (all OP_EVP)

if more emphasis is given to a shorter Transmission Delay; and (iii) if we desire a lower Total Time, a second alternative is $OP\_EVP = 10$ (packet with 2,488 Bytes). This value also performed well in the TxD analysis, although it was not as good as $OP\_EVP = 15$. The advantage in using $OP\_EVP = 10$ rather than $OP\_EVP = 15$ would be the fact of having more frequent updating of scientific data visualized on the GS's computers.

We can highlight some points given the results presented by using TPerP for this space system, aiming to apply our approach to other types of systems. For applications that need to store the acquired data and transmit them to another remote system, using a minimum value of a parameter (size of memory buffer, size of a packet) may be, at first, advantageous because, usually, this implies less processing demands (for example, less complex memory management). However, not always this minimum value may be the most appropriate (as we showed in this study), especially considering real-time systems where performance requirements may have greater relevance.

Different types of delays may require different solutions for the optimal values of the parameters that are being evaluated. The decision to choose the most suitable value of the parameters will depend on a priority of the considered delays.

5 Related Work

In this section, we will focus on some relevant studies that use formal verification methods aiming at performance/dependability evaluation of systems.

Probabilistic Model Checking was applied to address usability concerns that involve quantitative aspects of a user interface for a groupware system [5]. There are usability issues that are influenced by the performance of the groupware system rather than by its functional behavior. Our approach allows a more refined analysis considering the size of PDUs to estimate the rates compared with their work. In addition, the CTMC model they developed is very simple (19 states) which raises doubts whether the same results would be achieved for more complex models.

Probabilistic Model Checking was also used to analyze dependability properties of a embedded controller-based system [4]. Properties like the probability of the system shutting itself down, the expected amount of time spent in different classes of states of the model (using reward-based properties), expected number of reboots, were taken into account. The basic difference between TPerP and this work is that we aimed at evaluating the system time performance considering its normal operational behavior, and in [4] the authors aimed to assess dependability-related issues.

In [6], the authors showed how CSL can be used to specify state- and path-based dependability properties of systems being modeled as CTMC. Although dependability was the focus and properties in CSL to reason about probabilities of Quality of Service (QoS) were considered, some time-performance requirements were assessed. This paper is more a proof of concept to show the potential of the recently, at that time, introduced CSL for dependability/performance analysis.

A report on the use of the COrrectness, Modeling and Performance of Aerospace SyStems (COMPASS) toolset for correctness, dependability, and performance analysis of a satellite system-level design was presented in [12]. The greatest motivation behind their research is having a single, integrated, system model that covers several aspects (discrete, real-time, hybrid, probabilistic) rather than using various (tailored) models covering different aspects. The case study is interesting and complex (50 million states) and they accomplished several analyzes. However, the performability evaluation analysis ran out of memory after 9 hours. Moreover, the analysis was conducted when the satellite project was in its Phase B where requirements were not fully detailed. Thus, it is not very clear if they were able to use detailed and defined requirements as we did in our case study and also whether the performance analysis accomplished considers the same type of time perspective and granularity that we carried out.

In a follow-up paper, authors of [12] published another work where they presented the application of the COMPASS toolset to the same project but addressing Phase C of the satellite's system engineering lifecycle [13]. Thus, there were many more design details than in the previous attempt. However, they focused on diagnosability, not performability, analysis which was intractable in the previous work as it needed more computing resources than they had available.

Performance modeling of concurrent live migration operations in cloud computing systems was presented in [14]. The authors used CTMC and Probabilistic Model Checking as we did. They made some assumptions in their model so that they could accomplish the analysis in a shorter time. For instance, migration requests for sender servers are distributed in a uniform way, and thus they could simplify the model for a large number of servers with the ratio of the number of sender and receiver servers. This is clearly an attempt to deal with the state space explosion problem. It is not evident if this uniform assumption is consistent with the real characteristics of such systems.

We can point out the following differences and in some cases advantages of our approach compared with these previous studies: (i) TPerP has well-defined activities for the application of Probabilistic Model Checking for evaluating a specific type of performance measure (delay). It is vital that systematic procedures are proposed so that formal methods can have a wide acceptance in the industrial practice; (ii) we clearly define equations to calculate the parameter rates of the CTMC model considering local PDU influence, queue processing delays, encapsulation influence due to network architecture models, and time resolution; and (iii) with the exception of the studies [12, 13], all remaining papers that we mentioned used very small case studies. We dealt with complex models and so we believe that our approach is suitable for large scale applications.

6 Conclusions

In this paper we report on the use of Probabilistic Model Checking to evaluate time performance of complex systems. We organized the activities that we carried out in TPerP, an approach that analyzes several types of delay and goes towards a wide acceptance of formal methods in practice. Our approach defines clear steps to be followed by professionals by providing guidelines to calculate parameter rates, and suggesting the use of a specification patterns system.

We applied TPerP to a complex space application under development at INPE aiming at finding the optimal/most suitable size of the scientific data TM packet, so that there is a minimum delay of such scientific data stored onboard and the same data that are visualized on the ground. We found that the current definition of the balloon-borne experiment is inadequate and we suggest different sizes for the TM packet: $OP\_EVP = 15$ if we consider a shorter Transmission Delay; or $OP\_EVP = 10$ if the Total Time is the driving factor. CTMC models up to 30 million reachable states and 75 million transitions were analyzed showing the usefulness of our approach.

Future directions include to propose a method for the automatic translation of the system/software behavior into the high-level modeling language of a Probabilistic Model Checker such as PRISM. This step is quite interesting because it can prevent errors in (manual) translating the system solution into the language of the Model Checker. It is also interesting to investigate the use of other ProPoST's patterns and find out the impact on the time performance analysis. Finally, application to other case studies (aerospace domain, automo-

tive industry, etc.) should be addressed to consider the generalization of our approach.

Acknowledgments

This work was supported by a CNPq BSP grant, Institutional Process Number 455097/2013-5, Individual Process Number 170143/2014-7.

References

1. G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. Jonh Wiley & Sons, New York, NY, USA, 1998.
2. C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Performance evaluation and model checking join forces. *Commun. ACM*, 53(9):76–85, 2010.
3. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Engineering*, 29(6):524–541, 2003.
4. M. Kwiatkowska, G. Norman, and D. Parker. Controller dependability analysis by probabilistic model checking. *Control Engineering Practice*, 15(11):1427–1434, 2006.
5. M. H. ter Beek, M. Massink, and D. Latella. Towards model checking stochastic aspects of the thinkteam user interface. In *Interactive Systems. Design, Specification, and Verification*, volume 3941 of *LNCS*, pages 39–50. Springer, 2006.
6. B. R. Haverkort, H. Hermanns, and J.-P. Katoen. On the use of model checking techniques for dependability evaluation. In *Proc. IEEE Symp. Reliable Distributed Systems*, pages 228–237. IEEE, 2000.
7. L. Grunske. Specification patterns for probabilistic quality properties. In *Proc. Int. Conf. Software Engineering*, pages 31–40. ACM, 2008.
8. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proc. Int. Conf. Software Engineering*, pages 411–420. ACM, 1999.
9. S. Konrad and B. H. C. Cheng. Real-time specification patterns. In *Proc. Int. Conf. Software Engineering*, pages 372–381. ACM, 2005.
10. A. Post and J. Hoenicke. Formalization and analysis of real-time requirements: A feasibility study at BOSCH. In *Verified Software: Theories, Tools, Experiments*, volume 7152 of *LNCS*, pages 225–240. Springer, 2012.
11. J. Braga, F. D'Amico, M. A. C. Avila, A. V. Penacchioni, J. R. Sacahui, V. A. Santiago Júnior, F. Mattiello-Francisco, C. Strauss, and M. A. A. Fialho. The protoMIRAX Hard X-ray Imaging Balloon Experiment. *Astronomy & Astrophysics*, 580:A108, 2015.
12. M.-A. Esteve, J.-P. Katoen, V. Y. Nguyen, B. Postma, and Y. Yushtein. Formal correctness, safety, dependability, and performance analysis of a satellite. In *Proc. Int. Conf. Software Engineering*, pages 1022–1031. IEEE Press, 2012.
13. M. Bozzano, A. Cimatti, J.-P. Katoen, P. Katsaros, K. Mokos, V. Y. Nguyen, T. Noll, B. Postma, and M. Roveri. Spacecraft early design validation using formal methods. *Reliability Engineering & System Safety*, 132:20 – 35, 2014.
14. S. Kikuchi and Y. Matsumoto. Performance modeling of concurrent live migration operations in cloud computing systems using PRISM Probabilistic Model Checker. In *Proc. IEEE Int. Conf. Cloud Computing*, pages 49–56. IEEE, 2011.