Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Comparing Simulative and Formal Methods for the Analysis of Response Times in Networked Automation Systems

Jürgen Greifeneder, Liu Liu, and Georg Frey

Electrical and Computer Engineering Department
University of Kaiserslautern, Kaiserslautern, Germany
e-mail: {greifeneder|liuliu|frey}@eit.uni-kl.de).

Abstract: Networked Automation Systems (NAS) result from the increasing decentralization of automation systems using new network structures. Those structures are less expensive and more flexible than traditional ones. However, they introduce stochastic and coupled temporal behavior. Therefore, a detailed analysis is necessary accounting for the special characteristics of NAS. In this article, two approaches for the analysis of response times in NAS are presented. While simulation using Dymola/Modelica offers a user-friendly implementation of the system models, probabilistic model checking using PRISM gives more accurate and reproducible results in less time. The strengths and weaknesses of the two approaches are discussed based on a typical NAS scenario. The results are then validated by a large number of measured samples. It is demonstrated that quite accurate results are obtainable by both approaches.

1. INTRODUCTION

The trend towards an increasing decentralization in automation systems by means of new network structures leads to Networked Automation Systems (NAS, Fig.1). Due to those networked and decentralized architectures, a variety of delays with probabilistic duration are introduced into NAS. These aspects have direct influences on dependability, quality, safety, and reliability issues of automation processes.

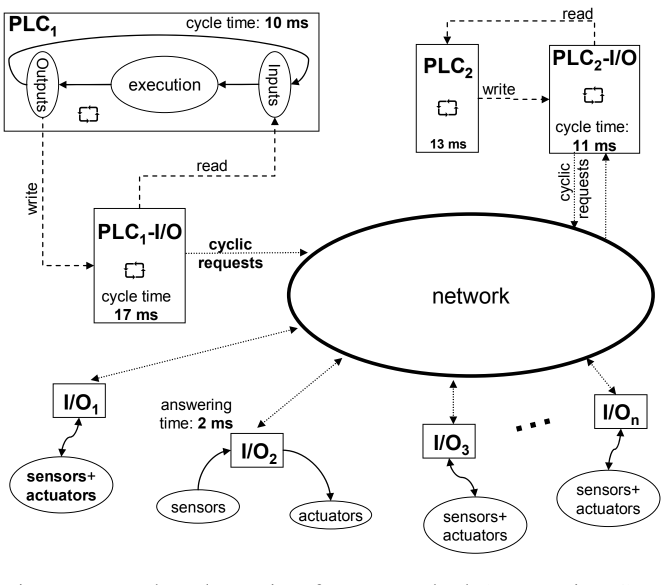

Fig. 1 Example schematic of a Networked Automation System (NAS).

The analysis of response times (i.e. delays) lies the basis for the quantitative evaluation of temporal system properties. However, only few methods are feasible for such an analysis (see section 2). This paper is arranged to cover such issues in detail, and is organized as follows: In the third section, a simulative approach using the simulation environment Dymola is introduced, followed by the presentation of a formal approach based on Probabilistic Model Checking (PMC) in section 4. Section 5 compares these two methods. Further, the two approaches are applied to a case study and the obtained results are compared with extensive laboratory measurements in section 6. Finally, some important points are summarized and an outlook is given.

2. REQUIREMENTS FOR ANALYSIS METHODS

For the analysis of response times in a NAS, it is necessary to take account of the process shown in Fig. 2. The process to be supervised covers the signal change at a sensor, as well as the associated signal processing and the resulting reaction at the actuator. Such a procedure begins with sending the request message from the PLC-I/O to the field-I/O. After being transmitted through the network, processed by the field-I/O and transmitted back, the replied message is processed by the PLC. In this course an associated actuator instruction (together with the next inquiry on sensor) is sent from the PLC-I/O to the field-I/O through the network. The process ends with the activation of the actuator.

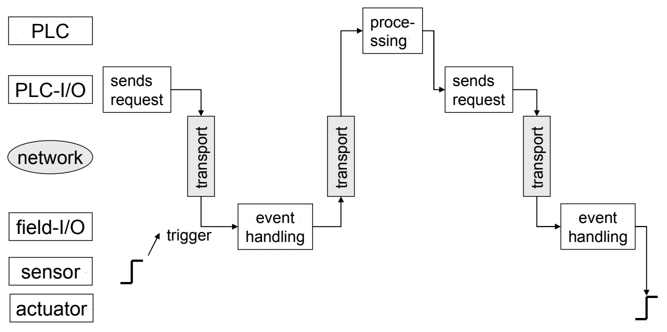

Fig. 2 Response time in NAS.

If failures, errors, and queuing times shall be considered, it is necessary to know the corresponding occurrence probability functions. Furthermore, it is important to consider the times, which are necessary to recognize and to handle the above mentioned situations. Additionally, the probability, that an information is completely lost, i.e. the probability that a necessary information does not arrive within a given time interval, needs to be taken into account.

From the above process, four fundamental questions in the analysis of response times are concluded:

1. Maximum- und minimum response times: Examples for this are: (1) the time which elapses from the activation of an emergency-stop button till the stop operation being executed, or (2) the earliest permissible opening of a clamp after assigning an instruction.

2. Distributions: For the analysis of performance and quality of control, it is essential to determine the (whole) probability distribution. The mean value and deviation can be computed from the distribution and properties such as error sensibility can be also estimated. Such a probability distribution can be determined directly using an analytic approach or by composing the relative portions determined for several time segments.

3 Interval probabilities: Therein are all the questions associated, which are based on the probabilistic boundaries. For example: with which probability is the system capable to react on a signal change within a given time interval? To answer questions like this, it is not necessary to calculate the whole distribution of the response times. Yet it is sufficient, as all the information is given within the distribution.

4. Differences, Distances: time difference between arrivals of two sequential data-packets or the probability whether the data packets arrive in the correct sequence.

For the analysis of (in account of response times) dependability and performance relevant questions of a NAS, it is required to use a methodology, which fulfils the following five properties:

1. Modeling capability of stochastic and timed eigenbehaviors of all the components. In particular, it has to be noted, that it is necessary to have the ability of implementing both, arbitrary time distribution as well as deterministic and stochastic decisions.

2. Definition capability of arbitrary stochastic and timed initial- and boundary conditions. This property is especially required because mostly the system and the process to be considered are not synchronized.

3. Verifiability of stochastic and timed properties. It is important to enable the possibility to determine probability distributions over time as well as to determine the probabilities for a system behavior within a given time frame. Pure worst-case analysis is not appropriate due to the stochastic eigenbehaviors of NAS.

4. Capability to implement the interrelations between components among each other and between components and input signals. This is essential for the description of various synchronization aspects and process dependencies in a NAS.

5. Statistical significance of the results. In a NAS, lots of effects occur with low probabilities. Therefore completeness and confidence of results are required. Completeness means that the analysis considers all the possible evolutions of the system. On the other hand, confidence of results indicates that the obtained result can be regarded as an acceptable approximation of the correct probability value (Error accuracy).

Besides these five key criteria, the scalability of the system model built by the analysis method is particularly important. This is the main requirement to analyze re-configurations and to do re-engineering. For example, with high scalability of models, it is possible to discuss the influence of connecting an additional component with reasonable expenditure and acceptable time costs. Moreover, other properties should also be considered, such as lower resource consumption concerning time and storage costs and complementary software tools.

Known approaches for the determination of delay times are measurement, static analysis, verification and simulation. The measurement procedures can also be used for validation and especially for parameterization of the models needed in other approaches, e.g. (Parrott et al, 2006), (Irey et al. 2004).

Static analysis methods furnish statements based on the system configuration. I.e. all system processes are quasi frozen and represented by characteristic values such as e.g. the minimum, maximum, or average value. While average values are usually used for questions of capacity interpretation, an analysis using minimum values is as much a safety aspect (“the earliest opening of a security clamp”) as using maximum values. The maximum value based analysis is also used for a more detailed interpretation of capacity questions. If a system interpretation does not consider the exact architecture of the entire system (such as e.g. the exact modeling of bottlenecks) this will result in very conservative limits. A multitude of approaches and tools are available for static analysis, such as e.g. (Stanczyk and Obuchowicz, 2003). The main advantage of static analysis methods lies in their low computational complexity and hence in the fact that even large systems can be analyzed with relatively little efforts and time. The disadvantage is, however, that only statements on behavior modes for extreme values are possible.

An interesting alternative is the stochastic network calculus, introduced by (Vojnovic and Boudec, 2002) as it is able to deliver distributions if the input variables are independent of one another. Same is true for the use of queuing theory as applied e.g. by (Song et al., 2002). However, in a NAS, the behavior of the components is not independent, as for example one and the same component might be passed twice. If that component then exhibits a cyclic behavior, its influence is not the same in the two cases.

Formal verification approaches like model checking (MC), have for their aim to cover all of the system’s practically possible evolutions. Further-more, in comparison to static approaches, MC holds the advantage of heeding to and analyzing both parallel and dynamic behavior equally. In the case of classical model checking, the main focus lies on the question of whether or not a property to be checked is fulfilled. Although there is a number of works dealing with MC, e.g. (Vogel-Heuser et al., 2006), and a number of very good tools, e.g. Uppaal, Kronos, are available, classical model checking still has two disadvantages unacceptable for NAS analysis:

firstly, stochastic behavior modes cannot (or hardly at all) be represented, and secondly, results are strictly binary, which has the consequence that examinations not result in probabilistic statements (respectively distributions of delay times). The situation changes with the use of Probabilistic Model Checking. PMC is based on models with probabilistic transitions. Furthermore, it allows to test probabilistic statements or even to derive distributions. Strictly speaking, the latter is not a function of the classical PMC, but supported by all common tools (e.g. PRISM, MRMC).

Simulating approaches mainly differ from static ones by their extended possible illustrations: dynamics and random decisions can be imitated in simulations. Yet, the used tools are either borrowed from the net analysis (e.g. OMNet) or else they use simulators of discrete event e.g. CPNtools (Marsal et al , 2006) or hybrid systems e.g. Matlab/Simulink (Cervin et al., 2003) or Dymola (Liu and Frey, 2007).

From the comparison above, it results that only simulation and probabilistic model checking are suitable for the analysis of NAS. However, both methods have their own specific characteristics and therefore their suitability for a certain investigation depends upon the given objective. Hence, prior to the comparison in section 5, both methods are introduced exemplarily in some detail in the following two sections. A Modelica Model is implemented for the simulative approach (section 3) and the Probabilistic Model Checker PRISM is utilized for the probabilistic formal method (section 4).

3. SIMULATION

The basic principle of a stochastic simulation is to reproduce a system's behavior randomly – by using varying initial conditions – on the basis of a (probabilistic) system model. By repeated executions, diverse images of the system's behaviors can be gained. The obtained results point out the individual evolution on the system as well as several different description properties such as mean value and distribution of global system behavior. However, the statistical significance is hereby an inevitable deficiency. That is to say, the result obtained by overlying individual system evolutions can only be regarded as an acceptable approximation of the correct system behavior under the restriction of a large amount of simulation runs. Due to the interactions of different components as well as the existence of failures and interferences in a NAS, events with low occurrence probabilities have to be considered. Thus it is necessary to determine the required amount of simulation procedures by means of statistic approaches of confidence interval estimation. For example, in order to guarantee for a result in the magnitude of $r=10^{-3}$ with a confidence of $\gamma=99\%$ that a simulated result deviates by a maximal factor of $B=0.1*r$ from the real value, the following formula can be used to determine the necessary amount of samples n:

$$n = 2{z_{1-\alpha/2}}^2 \frac{r\,(1-r)}{B^2} + \frac{{z_{1-\alpha/2}}^2}{B}\sqrt{1+\frac{4r^2(1-r)^2}{B^2}}$$

Here $z_{1-\alpha/2}$ is estimated from the standard distribution function tables in which γ is the integral of this function within the boundary of $\pm z_{1-\alpha/2}$. In the case of the given example, it is required to have 1.3 million samples in order to make a correct approximation. Fig. 3 illustrates the graph of necessary samples for different values of B. According to those graphs, it is clear that if only a rough estimation about system behaviors is required, fewer samples are sufficient (see the comparison in Fig. 11). If the necessary amount of samples is taken into consideration, the simulative approach becomes appropriate for the response time analysis of NAS. The great advantages of simulation are the various tool-supports and the integrated graphical interface, thus the modeling and calculation of complex systems in large scale is relative simple.

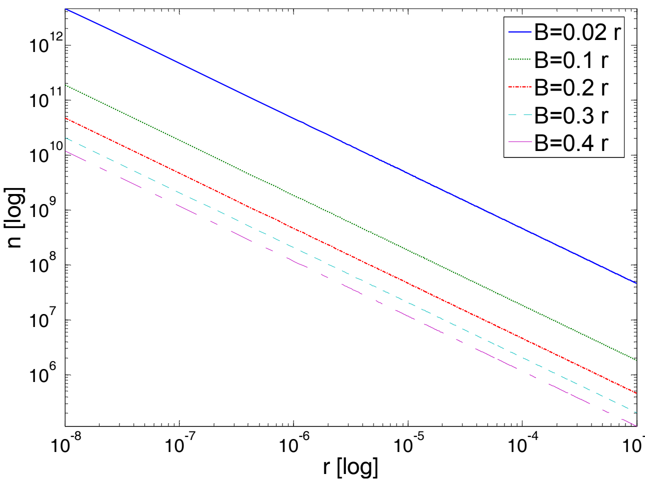

Fig. 3 Minimum required sample size to guarantee a resulting relative percentage r being within an interval [r-B, r+B] with a probability of 99%.

To choose the appropriate random algorithm for different system evolutions, the following requirement needs to be considered. That is, whether all the possible system evolutions are covered in accounting for periodic execution and initial values.

The problem is that it is not sufficient to consider the execution sequences in their local surrounding only, as lots of the distributed processes operate in a manner that are not as the same as they would, if they were separated. Furthermore, two questions arise:
(1) In which state will each of the distributed processes be found by the occurrence of an external signal? To cover different possibilities, the input signal has to be generated at random times. This is implemented in the same way as for the measurements, as discussed in some detail in the Fig. 9.
(2) How to handle unsynchronized interacting processes? In the presented comparison it is assumed that either the period lengths of interacting processes are prime or exhibit minimum deviations of system clocks. In this special case, all the distributed processes can be shifted arbitrarily to each other. In simulation, this can be generated based on the following assumption. If a random deviation of up to 0.01% is added to all cycle times, any arbitrary time shift is reachable in the long run.

In the presented approach (Liu and Frey, 2007) the models are built in Modelica and graphically presented and connected in Dymola. Modelica/Dymola has the advantage that its component-oriented graphical interface is easily understandable in the automation community. The library concept of Modelica dramatically reduces the time needed to model systems. In (Wagner et al, 2008), a device library with network components and process hardware was introduced (available at: http://www.eit.uni-kl.de/frey/). Compared to other tools such as Matlab/Simulink, Modelica emphasizes the advantages of object-oriented modeling. i.e., modeling is based on connection of components without consideration of calculation order. The resulting signal flows in the overall system need not to be defined globally; instead, they are constructed automatically from various locally defined behaviors.

Fig. 4 shows a Model of a small NAS as built using the previously mentioned library. It consists of a PLC, which is connected with two IO-modules through an Ethernet-Switch. At one of the IO-Modules a Sensor got connected, while at the other one an Actuator got attached.

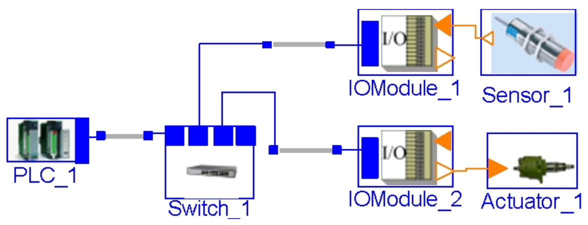

Fig. 4 Screenshot of the NAS-model created in Modelica/Dymola.

4. PROBABILISTIC MODEL CHECKING (PMC)

By using a formal logic, it is verified whether or not a predefined property will hold on a system's model. In contrary to simulation based approaches, a simple repetition of evolutions within the model is neither possible nor necessary for PMC. This is the most important advantage. Oppositely, the main disadvantage is that there is nearly no tool-support available for the relative young PMC method. In consequence, the modeling task has to be done on program-code-level (cf. Fig. 7). Furthermore, PMC does not support the use of continuous models. From here it follows that the accuracy of the results is dominated by the discretization step width. Unfortunately the step width can not be chosen arbitrarily small, as this would result in an exponential increase of the state space. Moreover, access conflicts (respectively the waiting queues which are necessary to deal with access conflicts) comprise a real challenge to the resource consumption. Finally, for the use of PMC it is necessary to reduce the system's complexity by dividing large systems into smaller units, which is not supported by any tools yet. However, the already discussed time transformation makes it possible to describe a large number of complex processes by their characteristic (stochastic) transfer behavior – and PMC only proves one property at the same time, i.e. adjusted models can be used.

If the processes of the system are independent, the determination of the initial state distribution is relatively simple: each process may proceed from each of his states. If the processes are not independent, the dependencies have to be implemented, which is not too complicated by using PMC. Doing so, it is not necessary to determine an optimal start scenario as in the case of simulation. This is because all possible initial states will be covered automatically.

It is neither necessary nor wise nor possible to let the system evolve longer than the process to be supervised. It is not necessary, as all evolutions will have been covered after one run of the process anyway. It is not wise, as the state space would be increased unnecessarily. It is not possible, as some situations occur more than once, due to the cyclic character contained in several processes. This would lead to tampered results as discussed in (Greifeneder and Frey, 2006).

In this work, software from the University of Birmingham is utilized: PRISM (Kwiatkowska et al., 2002). Several examples of NAS components have been already modeled using this software. As the modeling process of NAS should be done in continuous time, whereas the model-checking must be executed using Discrete Time Markov Chains (DTMC), a design-process got defined in (Greifeneder and Frey, 2007a). The first step in this process (cf. Fig. 5) is the modeling task, which is using a graphical description language (DesLaNAS), defined in (Greifeneder and Frey, 2007b). DesLaNAS also supports the definition of initial state and "to-be-checked"-properties. In the next step, the model is mapped into a probabilistic timed automaton (PTA). This PTA is defined for the special needs of NAS as discussed in (Greifeneder and Frey, 2007a,b). One of the most important differences to other PTA-definitions is the necessity of NAS that not only the initial state but also the initial clock values must be assigned stochastically. In the third step, this continuous automaton is transformed into a discrete automaton in dependence of the discretization step width on the one hand and the properties to be verified on the other hand. The latter is important as depending on the discretization accuracy only part of the original information may be of interest for the property to be checked. Finally the PRISM-code is generated.

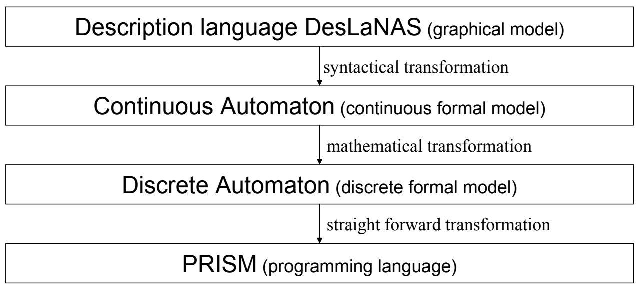
Description language DesLaNAS (graphical model)
syntactical transformation
Continuous Automaton (continuous formal model)
mathematical transformation
Discrete Automaton (discrete formal model)
straight forward transformation
PRISM (programming language)

Fig. 5 Design Process (Greifeneder and Frey, 2007a).

In Fig. 6 the DesLaNAS-model of a network transmission is shown. This automaton consists of two states: Initially, it waits in the state "idle" until a packet to be transmitted arrives. The corresponding event is named "send" which is written on the arc of the state transition. If such an event occurs, the automaton changes into the "deliver"-state. There, it stays for a stochastic time period $d_{net}(x)$ which abstracts the network transmission towards a stochastic process. This abstraction is feasible, as networks in NAS are quite fast and seldom crowded. As soon as this period has elapsed, the automaton returns into the "idle" state. Thereby the output event "*deliver*" is produced, which can be used by other automata in the system.

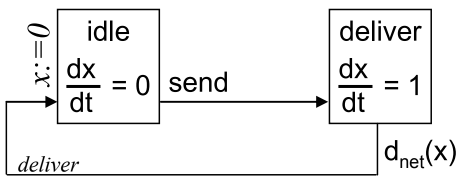

Fig. 6 DesLaNAS-Model of the network transmission.

In the time discretization step this becomes transformed to an automaton with n+1 states, where n is given by the maximum network delay (given by $d_{net}(x)$) divided by the discretisation step width (rounded towards the next integer). That is, the automaton has n states deliver, each of them associated with a specific time, and the idle state which does not have any time information and therefore is not influenced by the discretisation.

The corresponding PRISM-module-code is shown in Fig. 7. The state "idle" got coded as Net=0. If the (now discrete!) overall system's automaton is in a state, labeled with „send", an integer value between 1 and n is assigned to the variable Net using the probabilities pN1 for Net=1 and pN2 for Net=2 and so on. Net=n equals the maximum network delay, Net=1corresponds to the minimum. Obviously the probabilities pN1 to pNn must sum up to one. Finally, [t] and [p] represent synchronization variables which force this automaton to act synchronously to a clock wide synchronization impulse.

Note: The output event *deliver* must be set to true when the variable Net changes from 1 to 0. In this simple case however, this can be done by a logical representation that maps *deliver* to Net=1 without generating an additional variable.

```
module Network
  Net : [0..Net1Max] init 0;
  [t] Net=0 & !send -> Net'=0;
  [t] Net=0 &  send -> pN1:(Net'=1)+...+ pNn:(Net'=n);
  [t] Net>0            -> Net'=Net-1;
  [p] true             -> Net'=0;
endmodule
```

Fig. 7: PRISM-Code of the network transmission.

5. METHODS COMPARISON

Table 1 gives a comparison on the distinguishing aspects of both approaches: It is not possible to model continuous and closed-loop processes using PMC/PRISM at present. Furthermore, the modeling costs and complexity are relative high because of missing graphical tool support. The Modelica-based simulation approach in comparison has drawbacks in accuracy and resource consumption.

Table 1 Comparison of the methods

	Modelica	PMC
Closed-loop	+	-
Modeling costs	+	-
Complexity	+	-
Accuracy	-	+
Resource consumption	-	+

6. CASE STUDY

In order to validate the results from both proposed approaches, a NAS laboratory test-bed was designed (Fig. 8). One highlight of the test-bed is that the cycle times of PLC and PLC-I/O can be configured by software and thus various system configurations can be emulated on the same hardware. PLC and PLC-I/O are implemented on separate microcontroller boards (ATMEGA32). Moreover, the sensor, actuator and the evaluation module are also implemented individually on microcontrollers. The evaluation module takes charge of sensor signal generation and response time registration. The WLAN-based communication between these components is realized by WiPorts from Lantronix. WiPort is a compact network processor module which enables to build wireless communication over serial interfaces. The communication between PLC and PLC-I/O is realized by SPI-bus.

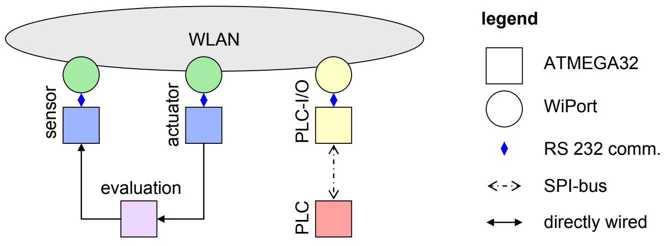

Fig. 8 Laboratory measuring setup.

The key point to guarantee stochastically independent measurements is to choose a proper random signal generation. The presented method is composed of two steps (Fig. 9).

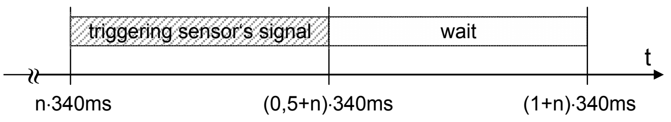

Fig. 9 Triggering the sensor's signal.

An event is triggered cyclically every 340 ms. Secondly, the sensor signal is generated after a random waiting time (0 to 170 ms). The evaluation of response time begins with the occurrence of the sensor signal and elapses until the activation of the actuator. After each measuring procedure, the obtained value is transmitted to a PC connected to the evaluation module through a RS 232 interface. The PC logs all the measured values in a text file for further evaluation. This way simplifies the logging of measured values and allows a long time measurement without considering storage limitations on the evaluation module. The chosen values (170 respectively 340 ms) are the multiples of the cycle times of 10 and 17 ms. Thereby it is ensured that all the possible shifts of cycles are taken into consideration.

For 1.3 millions samples, the experiment must run continuously for five days. The simulation takes 24 hours for 300.000 samples. In contrast, the PMC-analysis in PRISM takes 17 seconds. Fig. 10 shows the convergence of measurements and simulations against an increasing number of samples. The comparison of the results against those of PMC (Fig. 11) demonstrates that correct results are attainable by both approaches. However, it is noticeable that the (relatively large) PMC-step width of 1 ms leads to a less appropriate mapping of the two steps. However, decreasing the discretization step-width by a factor of 10 would not only require to rerun the PRISM-Code-generation shown in Fig. 5 but also lead to an increase in resource consumption (storage and time) that is definitely remarkable larger than a factor of 10.

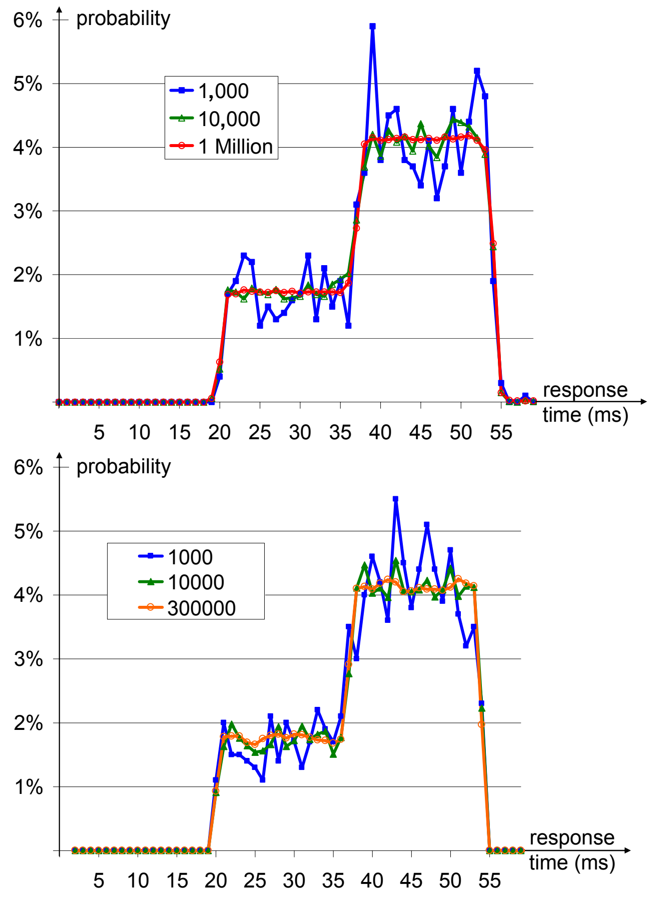

Fig. 10 Convergence of measurement (top) and simulation (bottom).

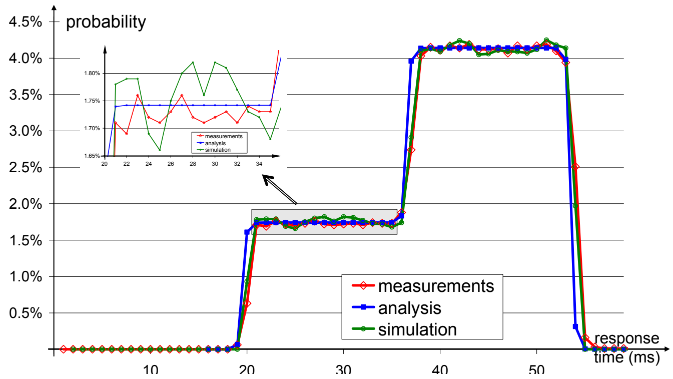

Fig. 11 Relative frequency distribution of 1.3 millions measured response times with 300000 simulated ones and the analytic ones. (Parameters: PLC: 10 ms and PLC-I/O: 17 ms).

7. SUMMARY AND OUTLOOK

In the presented work, two approaches for the analysis of response times in Networked Automation Systems are compared: the simulation by Dymola/Modelica and the formal analysis by Probabilistic Model Checking (PMC). It is demonstrated that both methods have their own specific characteristics and are therefore differently well-suitable depending upon the aim of the analysis. The Modelica-model is notably simple to handle, while the PMC is much faster for small systems as long as discretization is not too fine. As for the drawbacks: simulation results are only acceptable based on a large number of samples, while for PMC the necessary discretization and the difficult implementation of queues are to be noticed. Validation by measurements shows that good results are retainable by both methods.

8. REFERENCES

Cervin, A., D. Henriksson, B. Lincoln, J. Eker, K-E. Årzén (2003). How Does Control Timing Affect Performance? In: IEEE Control Systems Magazine, 23:3, pp. 16-30.

Greifeneder, J. and G. Frey (2006). Determination of Delay Times in Failure Afflicted Networked Automation Systems using Probabilistic Model Checking. In: Proc. 6th IEEE WFCS, Torino, pp. 263-272.

Greifeneder, J. and G. Frey (2007a). Probabilistic Timed Automata for Modeling Networked Automation Systems. In: Proc. 1st IFAC DCDS, Cachan, pp. 143-148.

Greifeneder, J. and G. Frey (2007b). DesLaNAS - a language for describing Networked Automation Systems. In: Proc. 12th IEEE ETFA, Patras, pp. 1053-1060.

Irey, P., B.L. Chappell, R.W. Hott, D.T. Marlow, K. O'Donoghue, T.R. Plunkett (2000). Metrics, Methodologies, and Tools for Analyzing Network Fault Recovery Performance in Real-Time Distributed Systems, IPDPS, Mexico, Springer LNCS 1800, pp. 1248-1257.

Kwiatkowska, M., G. Norman and D. Parker (2002) PRISM: Probabilistic symbolic model checker. In: Proc. TOOLS'02, Springer LNCS 2324, pp. 200-204.

Liu, L. and G. Frey (2007). Simulation Approach for Evaluating Response Times in Networked Automation Systems. In: Proc. 12th IEEE ETFA, Patras, pp. 1061-1068.

Marsal, G., B. Denis, J-M. Faure and G. Frey (2006). Evaluation of Response Time in Ethernet-based Automation Systems. In Proc. 11th IEEE ETFA, Prague, pp. 380-387.

Parrott, J., J. Moyne and D. Tilbury (2006). Experimental Determination of Network Quality of Service in Ethernet: UDP, OPC, and VPN, Proc. ACC, Minneapolis

Song. Y., A. Koubfia and F. Simonot (2002). Switched Ethernet for real-time industrial communication: Modelling and message Buffering delay evaluation. In: Proc. 4th IEEE Int. WFCS, Vasteras, pp. 27-35.

Stanczyk, J. and A. Obuchowicz (2003). The max-plus algebra approach to the prototyping of concurrent processes. In: Proc. 9th IEEE MMAR, Miedzyzdroje, Vol. 2, pp. 857-862.

Vogel-Heuser, B., D. Witsch, J.-M. Faure, G. Marsal (2006). Performance Analysis of industrial Ethernet networks by means of timed model-checking. In: Proc. 12th INCOM, Saint-Etienne, pp. 101-106.

Vojnovic, M. and J.-Y. Boudec (2002). Stochastic Analysis of Some Expedited Forwarding Networks. In: Proc. 21st IEEE INFOCOM, New York.

Wagner, F.; L. Liu and G. Frey (2008). Simulation of Distributed Automation Systems in Modelica. In: Proc. 6th Int. Modelica Conference, Bielefeld, Vol. 1, pp. 113-122.