Finding the bounds of response time of networked automation systems by iterative proofs

Silvain Ruel, Olivier de Smet, Jean-Marc Faure

Finding the bounds of response time of networked automation systems by iterative proofs

S. Ruel * **O. de Smet** * **J.-M. Faure** *

* *LURPA, ENS de Cachan, Cachan, France, (e-mail: {silvain.ruel, olivier.de_smet, jean-marc.faure}@lurpa.ens-cachan.fr)*

Abstract: Response time of modern automation systems is not constant but is featured by a distribution of values; finding the upper and lower bounds of this distribution is a crucial issue when designing critical systems. This paper shows how to obtain these bounds by proving timed properties on a formal model of the system, in the form of communicating timed automata. In this approach, bounds are obtained by iterative proofs of properties which are expressed by means of a parametric observer. Comparison of analysis results of formal models to measures on real automation systems shows the accuracy and interest of this approach.

Keywords: Time performances evaluation, Formal verification, Formal methods scalability, Uppaal

1. INTRODUCTION

Response time to an input event is a very significant time performance of any automation system. For a networked automation system (NAS), this feature does not own a constant value but is characterized by a distribution of values, however. The significance of this time performance explains why several methods have been developed to obtain, from a model of the system, either its distribution, like simulation methods (Pereira et al. (2004)), or the upper and lower bounds of this distribution, or an over-(under-) estimation of these bounds, e.g. analytic methods based on worst-case (best-case) performance estimation (Hung et al. (2004)) or on network calculus (Georges et al. (2002)). For NAS that are used in critical systems, focus is mainly, or even exclusively, put on the bounds of the distribution; in the rest of this paper, only this kind of application and then bounds assessment will be considered.

Timed model-checking is a promising technique to analyse critical systems, because it performs an exhaustive analysis of formal models. However, a timed model-checker yields only Boolean proof results (some properties hold or not) and is not able, in itself, to provide numerical values. The aim of this paper is to propose a response time bounds assessment method that benefits from the exhaustive analysis possibilities of model-checking. The key idea of this novel method is to prove iteratively formal properties that are expressed by using a parametric observer automaton.

The outline of the paper is the following. The class of NAS which is considered in this work as well as an experimental facility to measure time performances are presented in section 2; comparison of measures on real systems to formal analysis results will allow to validate the proposed approach indeed. The method to obtain time response bounds by proving iteratively formal properties is detailed in section 3. Section 4 addresses the issue of NAS formal models construction; a strategy to build models which are tractable by existing timed model-checker while leading to trustworthy proof results is explained. Last, experimental results that have been obtained both by measurement and by the proposed method are compared and discussed.

2. RESPONSE TIME OF AUTOMATION SYSTEMS

2.1 Class of NAS considered

Several industrial Ethernet solutions (Ethernet/IP, Modbus-TCP, Ethernet Powerlink, Profinet, ...) can be selected to implement a NAS. This paper considers only the NAS which rely on the Modbus-TCP protocol, while the methodology which is presented can be applied to other kinds of networks as well. More precisely, focus is put on the automation systems in which logical controllers and remote input-output modules (RIOMs) communicate to carry out automation functions; with the protocol selected, controllers are clients and RIOMs are data servers. Fig. 1 shows an example of such a system that will be used for defining a case study.

The main features of the physical components of these systems are:

- Controllers (Programmable Logical Controllers – PLCs – or industrial computers) are modular. Within each controller, a calculus processor runs a program cyclically, while a communication processor performs a cyclic scanning of some RIOMs, termed IOscanning. It matters to underline that the cycles of these two processors are asynchronous, data exchanges being made by means of a shared memory.
- The network includes Ethernet switches and Ethernet links and is dedicated only to communications between the controllers and the RIOMs; there is no other additional traffic.

- The inputs and outputs from/to the plant are gathered in RIOMs which are directly connected to the network. One RIOM may be shared by several controllers.

Moreover, it will be assumed that there is no frame loss, which is a quite reasonable assumption for this kind of switched industrial Ethernet solution in the concerned operational conditions.

2.2 Case study

To illustrate the response time bounds assessment method, the response time to a particular input event of the system depicted in Fig. 1 will be considered. This NAS includes 2 modular PLCs, 9 RIOMs and 3 Ethernet 8-port switches; only 5 ports are used for each switch in this system. PLC1 scans the inputs and outputs from remote modules R1, R2, R3, R4, R5 and R6 whereas PLC2 communicates with remote modules R4, R5, R6, R7, R8 and R9. Thus both PLCs share 3 modules (R4, R5 and R6) and one switch (SW2).

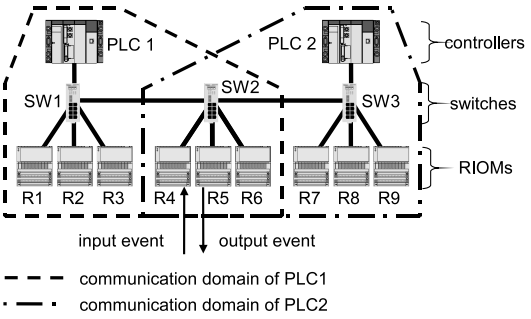

Fig. 1. Case study

Only the response time to an input event on remote module R4 will be focused on, in this case study. More precisely, focus is put on the delay between the occurrence of a given input event on remote module R4 and the corresponding occurrence of an output event on remote module R5 that is a consequence of the input event. In this study, the value of this output event is computed by PLC1 from the value of the selected input (Fig. 2). PLC2 only reads the values of R4 and R5 inputs.

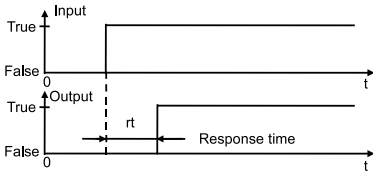

Fig. 2. Response time definition

The experimental results that will be presented in what follows - measures on a real NAS in the next sub-section, as well as results of analysis of a formal model of this NAS, in section 5 - have been obtained by instantiating the components of this system with industrial COTS (components off the shelf) from the Schneider Electric company. In particular, for each PLC, the duration of the cycle of the calculus processor spans from 2 to 3 ms; these values were directly obtained with the programming software tool PL7pro from the company. The IOscanning duration ranges from 9.24 to 10.74 ms; these latter two limits were obtained by using the Wireshark network protocol analyzer (Combs (2007)).

2.3 Response time measurement

A facility to measure time performances of systems that can be modeled as DES (Discrete Event Systems) has been developed at LURPA some years ago. This facility is named PRISME [1] and is mainly composed of:

- a square wave generator which delivers variable period signals whose rising and falling edges may be seen as events;
- a logical analyzer to collect timing diagrams that contain both events occurrences order and durations between occurrences;
- a real-time computer to coordinate events generation and timing diagrams collection.

To measure the response time of a NAS (Fig. 3), the wave generator is connected to the considered input and the logical analyzer collects the timing diagrams of the input and output from which the response time is defined.

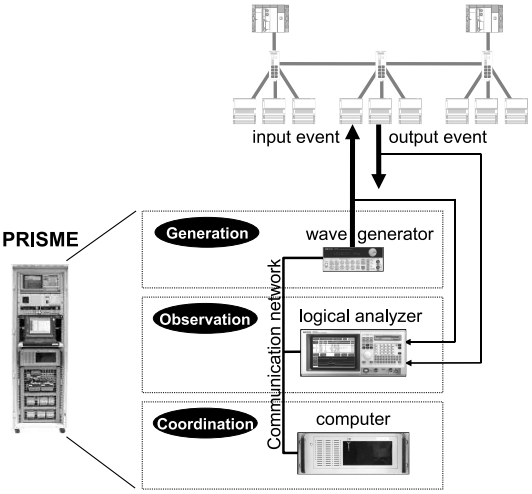

Fig. 3. Response time measurement

As the response time is characterized by a distribution of values, lots of measures are necessary; this is not an issue with the PRISME facility because measurement is automated in that case. Nevertheless, it matters to avoid synchronization between the input signal and the IOscanning cycle to prevent from meaningless measures. Then the frequency of the input signal must be set as explained in Denis et al. (2007).

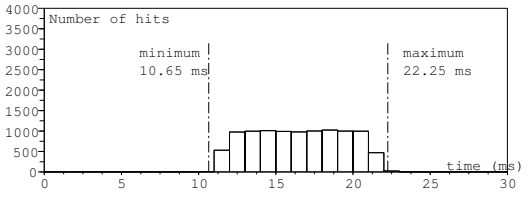

Fig. 4. Histogram of the response time

[1] French patent # 01 110 933

Fig. 4 shows the result of 10,000 measures of response time on the NAS described in the previous subsection. Given this large number of measures, the minimal and maximal values of this histogram will be taken as the lower and upper bounds of this time performance and will serve as references to validate the method to obtain response time bounds by iterative proofs which is described in the next section.

3. USING ITERATIVE PROOFS TO FIND RESPONSE TIME BOUNDS

3.1 Principle

The reader is reminded that the objective of model-checking is to prove whether a formal model satisfies (or not) some properties. This implies that model-checkers yield only Boolean results (the considered properties hold or not), and not numerical values. Hence, the method depicted in Fig. 5 [2] was developed to obtain response time bounds by using model-checking.

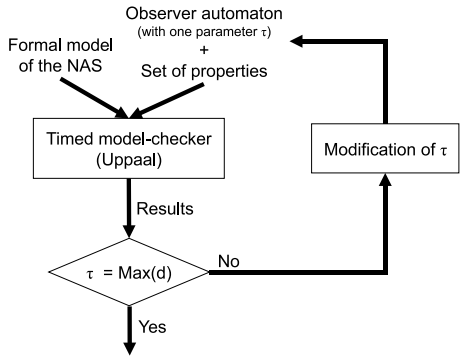

Fig. 5. Principle of the iterative method to obtain the upper bound of the response time

This method relies on UPPAAL (Larsen et al. (1997b)), a widely used timed model-checker. Formal models which are input into UPPAAL are to be represented in the form of communicating timed automata (Alur and Dill (1994)) and properties are written with a subset of CTL (Computational Tree Logic). In our case, formal models must describe the behavior of NAS in a detailed fashion and be tractable without state space explosion. This issue is addressed in section 4.

To ease properties writing, a parametric observer automaton has been introduced; the next subsection explains the role of this observer. Some guards of this observer depend on a time parameter τ whose value is modified for each iteration. The aim of subsection 3.3 is to present the algorithm that modifies this value and ensures iterations convergence.

3.2 Observer automaton

Generally speaking, the evolution of this automaton comprises three steps: input event emission, waiting for the corresponding output event, and comparison of the waiting time, that represents a response time, to a parameter τ. This is illustrated on Fig. 6 on the basis of the case study.

[2] This figure considers the upper bound search; to obtain the lower bound, only the test is to be changed in τ=Min(rt).

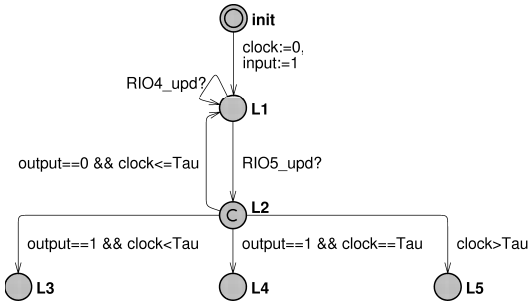

Fig. 6. Observer automaton OBS1

The transition from location Init to location L1 models the input event emission; the considered input (of RIOM 4 in the case study) is set and the clock is initialized. It is possible to leave location L1 only when the value of the considered output (of RIOM 5 in the case study) is updated (communication channel RIO5_upd between the observer and the NAS model). If this value is false and the clock value is smaller than the parameter τ, the observer waits for the next update (transition from L2 to L1). If it is true (the output event occurred), location L3 or L4 is reached according to the clock value and provided that this value is not greater than τ. If the clock value is greater than τ, location L5 is reached.

It matters to underline that the above description corresponds to one and only one evolution of the overall model which includes this observer and the NAS model. As a model-checker considers all possible evolutions, two or three locations may be reached when analyzing a NAS model. If both locations L3 and L4 are reached for instance, this means that the response time of the NAS is always lower than or equal to τ, i.e. τ is the upper bound of the response time.

Once this observer designed, three formal reachability properties can be stated:

$$E <> OBS1.L3 \quad (1)$$

$$E <> OBS1.L4 \quad (2)$$

$$E <> OBS1.L5 \quad (3)$$

These statements could be translated in natural language by "there is at least one evolution to reach location L3 (respectively location L4 and location L5) of automaton OBS1".

Then, proof of these three properties allows comparing the time parameter with the upper and lower bounds of the response time (Table 1).

Table 1. Proof results meaning

P1 is	P2 is	P3 is	meaning
false	false	true	τ is lower than Min(rt)
false	true	true	τ is equal to Min(rt)
true	true	true	τ is between Min(rt) and Max(rt)
true	true	false	τ is equal to Max(rt)
true	false	false	τ is higher than Max(rt)

where Min(rt) and Max(rt) represent respectively the lower and upper bounds of the response time.

3.3 Time parameter computation

The time parameter τ must be modified after each proof, if no bound has been found. To modify this parameter, two dichotomy research algorithms have been proposed, depending on the searched bound. Algorithm 1 is used for the upper bound; the algorithm to find the lower bound can be easily derived from this description.

Data: $L = Linit$ and $H = Hinit$
Result: Max(rt) the maximal response time
while $H! = L$ **do**
  $\tau = L + int((H - L)/2)$
  check properties
  case *P3 is true*
    $L = \tau$
  case *P3 is false AND P2 is false*
    $H = \tau$
  case *P3 is false AND P2 is true*
    $L = H = \tau$

$Max(rt) = \tau$

Algorithm 1. Dichotomy search algorithm to find the upper bound of the response time

In this algorithm, all variables are integers, because only this type of data can be defined in UPPAAL models. τ is computed as the mean value of two limits L (low limit) and H (high limit). From their initial values Linit and Hinit, L and H are modified according to the proof results. When τ is lower than the upper bound of the response time (P3 is true in Table 1), the next value of L will be the current value of τ, H remaining unchanged. When τ is strictly higher than the upper bound of the response time (P3 and P2 are false), the next value of H will be the current value of τ, L remaining unchanged. The algorithm stops when P3 is false and P2 is true; the number of iterations depends on the initial values of the two limits (Linit and Hinit).

Following this presentation of the overall principle of the method and of the use of proof results, the next section is devoted to NAS formal models construction.

4. MODELLING THE AUTOMATION SYSTEM

Building a formal model of a real system requires both to model accurately all the useful behaviors, so as to obtain trustworthy proof results, and to design a model that can be treated by the model-checker, without combinatory explosion. In the case of NAS, the modelling strategy which is depicted in Fig. 7 has been set up. The three steps of this strategy, that might be automated, will be sketched in this section and are explained with more details in Ruel et al. (2008).

The aim of the first step is to build a detailed model of the system in which each timed automaton, named component model, describes precisely the behavior of one physical component (calculus processor, communication processor, ...) or of one communication function, and includes parameters that represent time features (cycle time, processing time) of this component or function. In this detailed model, switches and cables are modelled by a set of independent communication functions, because the

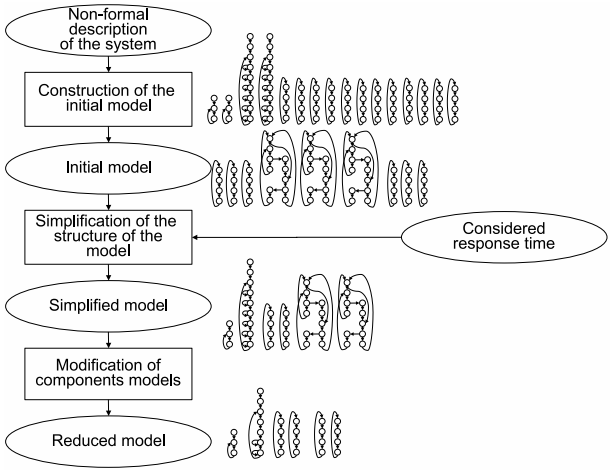

Fig. 7. Modelling strategy

traffic due to the exchanges between PLCs and RIOMs is far lower than the maximal throughput of the network, as experimentally shown in previous studies (Marsal (2006) and Denis et al. (2007)); then, the duration of an exchange between one PLC and one RIOM does not depend on the other exchanges.

The structure of a NAS model can be represented by a graph in which nodes are components models and edges represent communications between these models. The second step aims to simplify this structure by keeping only the components models that introduce delays which impact directly the considered response time, i.e. which are on the route of data flowing from the input to the output; all the other components models are removed. This simplification step relies on an interpretation abstraction which is similar to those developed for checking hardware systems - localisation reduction (Kroening (2006)) - or hybrid systems (Clarke et al. (2003)).

Therefore, this step yields a simplified model that contains a smaller number of components models; however, each one of these models is a detailed one that includes meaningless behaviors, e.g. communications with removed components. The role of the third step is to remove these behaviors from the remaining components models.

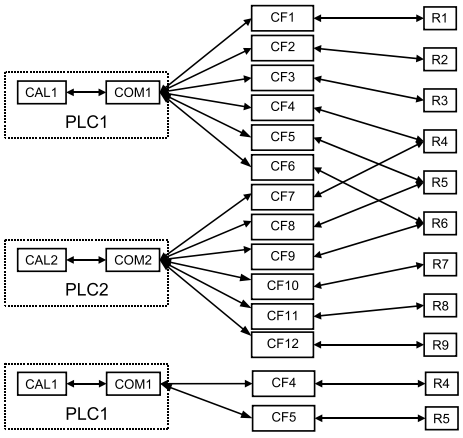

Fig. 8. Structure of the whole (top) and simplified (bottom) automation system model

To clean up the components models, the following actions are carried out during the last step:

- all the locations and transitions which correspond to meaningless behaviors, e.g. treatment of requests from removed components, are deleted;
- when a transition which belongs to a concurrent structure in the detailed model is deleted, the maximal duration of the locations that follow immediately the remaining transitions of this structure is increased by the time of the removed concurrent treatment. This allows to obtain reduced components models that encompass all real behaviors, e.g. treatment delay due to another concurrent treatment, even if they include not realistic ones which correspond to a worst-case modelling, e.g. it is possible that one concurrent process would always be selected first.

By using this modelling strategy on the case study, the structure of the whole model can be seen on the top of Fig. 8 whereas the structure of the simplified one is depicted on the bottom. To illustrate the modification of components models (step 3), the detailed and reduced models of a RIOM are depicted Fig. 9.

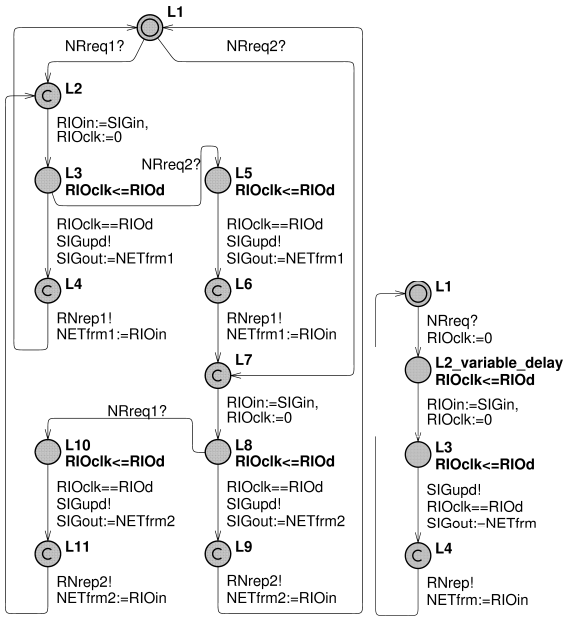

Fig. 9. Detailed and reduced models of a RIOM

It can be noted that, in this latter model, only the locations which correspond to the treatment of a request from PLC1 (L2, L3, L4) are kept, because the PLC2 component model has been removed. To account for concurrency between requests from PLC1 and PLC2, the duration of location L2 is variable from zero to one request treatment time.

5. EXPERIMENTAL RESULTS

5.1 Experiments configurations

Two kinds of data structures may be used to code the state space of a formal model within Uppaal. Difference bound matrices (DBM) is the standard state space representation (Behrmann et al. (2002)). Compact data structure (CDS) is a representation that reduces memory consumption, especially for models with many clocks (Larsen et al. (1997a)), but slows down verification. Only DBM coding was selected for the experiments. Using CDS leads to far too long verification times that are not acceptable in the prospect of an industrial application of the proposed method.

Uppaal offers also the possibility to prove properties on an over- or under- approximation of the state space. State space over-approximation uses convex-hull approximation of zones (Balarin (1996)); state space under-approximation is based on bit-state hashing (Wolper and Leroy (1993)). In the experiments which are described below, only over-approximation was retained; under-approximation is inappropriate for this work. As the objective is to find the bounds of the response time of a critical system, the whole state space must be explored indeed; this constraint excludes the use of under-approximation.

Four experiments were performed on the basis of the case study described in section 2. Two of them were aiming at analyzing the whole detailed formal NAS model; the two other ones used the reduced formal model obtained as explained in section 4. In both cases (analysis of a detailed or a reduced model), DBM coding, without approximation of the state space, and over-approximation were tested (table 2).

Table 2. Experiments configurations

	whole model	reduced model
without approximation	configuration 1	configuration 3
with approximation	configuration 2	configuration 4

5.2 Results comparison

The values of the response time bounds obtained in these four experiments as well as the computation time of these bounds are given in Table 3. All these experiments were made on a computer with 2 GB RAM, a Core 2 duo processor (E6400) and Windows XP professional; the initial values of the low and high limits of the dichotomy search algorithm were set respectively at 0.00 ms and 50.00 ms and the resolution (physical value of the time unit) to 10 μs.

Table 3. Results comparison

	response time bounds	verification time
configuration 1	impossible	not enough RAM
configuration 2	9.49 and 23.13 ms	12 min 26 s
configuration 3	9.49 and 23.13 ms	3 min 30 s
configuration 4	9.49 and 23.13 ms	2 s

The first observation that can be made from this table is that no result was obtained with the first configuration (whole model and no state space approximation) due to combinatory explosion. This shows clearly the need for model reduction or state space approximation techniques.

The three other experiments yielded the same bounds: 9.49 ms and 23.13 ms; none of the techniques that underlie bounds computation in these experiments can be preferred from these results. These values which come from models analysis must be obviously compared to the measures on the real NAS: 10.65 ms and 22.25 ms (minimal and maximal values of the histogram Fig. 4). Then the bounds

values computed on formal models are under- (for the lower bound) and over- (for the upper bound) estimations of the real values; this is not really surprising, given the choices made for these three experiments (state space over-approximation or/and worst-case modelling). The difference between theoretical results and measures is not too large (around 10 %) and, moreover, theoretical results are pessimistic, that is not an issue and even sometimes required when considering critical systems. Hence, this comparison validates clearly the proposed method of bounds computation by iterative proofs.

Focusing now on the computation time, the following conclusions can be drawn up:

- The modelling strategy which has been proposed in the previous section is more efficient than the state space over-approximation proposed by UPPAAL, when NAS models are considered. The computation time in configuration 3 is about 3.5 times smaller than that necessary in configuration 2.
- Combining the state space over-approximation and this strategy (configuration 4) is a very efficient solution. The computation time is divided by approximately 100 and 370 in comparison to configurations 3 and 2. This solution is the best one and can allow analysis of larger networked automation systems in reasonable times.

6. CONCLUSION

This paper has shown how the bounds of a significant time performance of automation systems: the response time, can be obtained by proving iteratively formal timed properties. The main contribution of this work is the definition of a parametric observer automaton; guards of this observer depend on a time parameter whose value is iteratively modified according to the previous proof results. This method can be extended to other time performances, like the difference of response times, by modifying the observer.

To avoid combinatory explosion when verifying formal models, a modelling strategy has been proposed. A detailed model of the automation system, composed of communicating components models, is first constructed. Then, the structure of the system model is simplified, in function of the considered response time, and the components models which remain in the simplified structure are modified to account for removed behaviours which can impact the response time.

Comparison of formal analysis results to measures validates the approach. The values that are yielded by the proposed method are close enough pessimistic approximations of measures and can be then used when designing critical systems. Moreover, these values are obtained in quite reasonable times.

Future works are aiming at extending this method, whose principle is to obtain numerical values from Boolean proofs results, to analysis of other quantitative properties of timed or hybrid systems.

REFERENCES

Alur, R. and Dill, D.L. (1994). A theory of timed automata. *Theor. Comput. Sci.*, 126(2), 183–235.

Balarin, F. (1996). Approximate reachability analysis of timed automata. In *RTSS '96: Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS '96)*, 52. IEEE Computer Society, Washington, DC, USA.

Behrmann, G., Bengtsson, J., David, A., Larsen, K.G., Pettersson, P., and Yi, W. (2002). Uppaal implementation secrets. In *FTRTFT '02: Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, 3–22. Springer-Verlag, London, UK.

Clarke, E., Fehnker, A., Han, Z., Krogh, B., Ouaknine, J., Stursberg, O., and Theobald, M. (2003). Abstraction and counterexample-guided refinement in model checking of hybrid systems. *International Journal of Foundations of Computer Science*.

Combs, G. (2007). Wireshark. In *http://www.wireshark.org*.

Denis, B., Ruel, S., Faure, J.M., Marsal, G., and Frey, G. (2007). Measuring the impact of vertical integration on response times in Ethernet fieldbuses. In *Proc. of ETFA'07*, 532–539.

Georges, J.P., Rondeau, E., and Divoux, T. (2002). How to be sure that Ethernet networks will satisfy the real-time requirements? In *Proc. of IEEE Int. Symposium on Industrial Electronics*.

Hung, M.H., Tsai, J., Cheng, F.T., and Yang, H.C. (2004). Development of an Ethernet-based equipment integration framework for factory automation. *Robotics and Computer-Integrated Manufacturing*, 20, 369–383.

Kroening, D. (2006). Computing over-approximations with bounded model checking. *Electronic Notes in Theoretical Computer Science*, 79–92.

Larsen, K.G., Larsson, F., Pettersson, P., and Yi, W. (1997a). Efficient verification of real-time systems: compact data structure and state-space reduction. In *RTSS '97: Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS '97)*, 14–24. IEEE Computer Society, Washington, DC, USA.

Larsen, K.G., Pettersson, P., and Yi, W. (1997b). Uppaal in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1-2), 134–152.

Marsal, G. (2006). *Evaluation of time performances of Ethernet-based automation systems by simulation of high-level Petri nets*. Ph.D. thesis, ENS Cachan (France) and Kaiserslautern University (Germany).

Pereira, N., Tovar, E., and Pinho, L.M. (2004). Timeliness in COTS factory-floor distributed systems: what role for simulation? In *Proc. of IEEE International Workshop on Factory Communication Systems*, 13 – 21.

Ruel, S., de Smet, O., and Faure, J.M. (2008). Building effective formal models to prove time properties of networked automation systems. In *Proc. of WODES 2008*, 334–339.

Wolper, P. and Leroy, D. (1993). Reliable hashing without collision detection. In *Computer Aided Verification. 5th International Conference*, 59–70. Springer-Verlag.