Building effective formal models to prove time properties of networked automation systems

Silvain Ruel, Olivier de Smet, Jean-Marc Faure

Building effective formal models to prove time properties of networked automation systems

Silvain Ruel, Olivier de Smet, Jean-Marc Faure

***Abstract*— This paper proposes a method to build formal models of networked automation systems, in the form of sets of communicating timed automata, which are reduced enough to avoid (or limit) combinatory explosion, but accurate enough to provide meaningful proof results, when they are checked. This method starts from a detailed initial model, which includes all behaviours of all components of the system, and comprises two steps. First, given a property to prove, the structure of the model is simplified so as to keep only the components models which impact directly this proof. Then the formal models of the remaining components are modified to take the previous simplification into account; the resulting models are worst-case models which guarantee trustworthy results. Experiments show the effectiveness of this modeling.**

I. Introduction

Networked automation systems (NAS) with Ethernet-based fieldbuses instead of traditional fieldbuses are more and more often used in industry, even for critical systems such as chemical or power plants. To ensure dependability of these systems, not only the functionalities but also the time performances must be validated.

Time performances validation may be achieved by simulation [1] or timed model-checking of a NAS model. For critical systems, this latter technique is more promising because it is based on an exhaustive analysis of the model. However, even if it has been used to determine features of NAS components or validate communication protocols [2] [3], timed model checking is not employed to evaluate global time performances of NAS, to the best of our knowledge. This comes from the well-known combinatory explosion issue; the formal models of real NAS are too large to be analysed by the existing timed model-checkers.

To limit combinatory explosion, new resolution methods, such as over-approximation, and evolutions of model-checkers [4] have been proposed. Another promising solution is to build formal models which capture all the useful behaviours of the modelled system, so that the proof results which are obtained with these models would be meaningful, but which are reduced enough to be analysed by existing model-checkers. This is the basic idea of this work. More precisely, this paper proposes a method to build, from a detailed formal model of a NAS, in the form of a set of UPPAAL [5] timed automata, a reduced model which can be checked more easily and quickly. This effective model will contribute to improve scalability of timed model-checking techniques.

S. Ruel, O. de Smet, J.-M. Faure are with LURPA, ENS Cachan, 61 av. du President Wilson, 94235 Cachan Cedex, France, {ruel,de_smet,faure,}@lurpa.ens-cachan.fr

Section II presents the class of networked automation systems that is considered and the time performances which are focused on; a simple but non-trivial example illustrates this presentation. Then, the different steps of our modeling method are described in section III and exemplified in section IV. Experimental results obtained with the detailed and reduced models are provided and discussed in section V.

II. Networked automation systems

A. Considered class of NAS

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

The main features of the physical components of these architectures 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 periodic scanning of some RIOMs, termed I/O scanning. 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 PLCs and RIOMs; there is no other additional traffic.
- 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 PLCs.

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 operation conditions. This explains why probabilistic model-checking [6], an interesting verification technique for architectures which require probabilistic modelling, was not considered in this work.

The architecture of the case study encompasses three PLCs and nine RIOMs. PLC1 communicates with four RIOMs

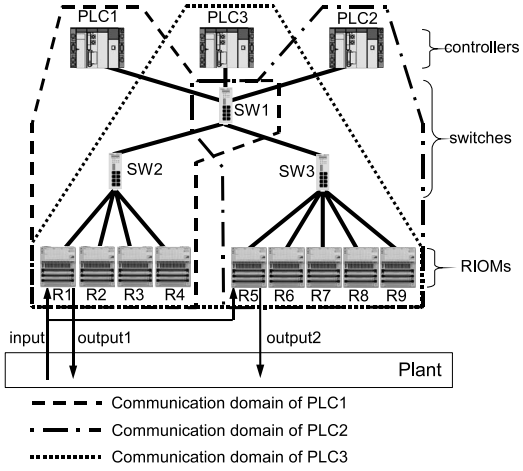

Fig. 1. Case study

(RIOMs 1 to 4), PLC2 communicates with five other RIOMs (RIOMs 5 to 9); thus, these two PLCs do not share any RIOM. PLC3, which runs monitoring functions, communicates with all RIOMs and consequently shares respectively four and five RIOMs with PLC1 and PLC2. To evaluate the time performances of this architecture, an input event and two output events are introduced. The input event, which could correspond to a global start or stop event, is observed by two RIOMs: RIOM R1, which is scanned by PLC1, and RIOM R5, scanned by PLC2. The responses to the input event, from PLC1 and PLC2, are respectively output1 and output2 and are sent to the plant through the same RIOMs.

The configurations of the 3 PLCs are given in table I.

TABLE I
CONFIGURATIONS OF THE 3 PLCS

	PLC1	PLC2	PLC3
calculus duration	2 to 3 ms	3 to 4 ms	5 to 6 ms
I/O scanning time	10 ms	10 ms	50 ms
RIOMs scanned	R1 to R4	R5 to R9	R1 to R9

B. Required performances

Several global time performances of a given NAS are to be considered, such as response times (delays between input events and output events which are consequences of those events, like rt1 or rt2 in figure 2) or the minimal duration of an input event which can be always detected, for example. This paper focuses on a more complex time performance: the delay between two output events which are both consequences of the same input event (fig. 2). This performance will be called difference of response times and noted d; it is worth evaluating when synchronisation between two elements of the plant must be ensured.

The final objective of this study is to find an upper bound of this difference, by checking with the UPPAAL model-checker whether the following property

$$d \text{ is always lower than } \tau,$$

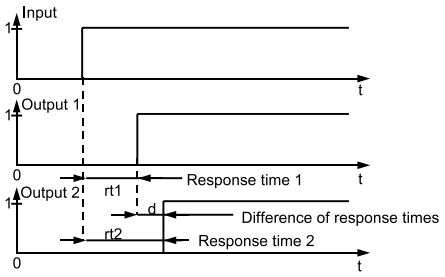

Fig. 2. Time performances

where τ is a variable parameter, holds on a formal model of the system. If the property does not hold for a given value $\tau 1$ of the parameter and holds for another value $\tau 2$ which is higher than $\tau 1$, then $\tau 2$ is an upper bound of the difference of response times. An accurate value of this upper bound can be easily obtained by reducing the difference between these two values by classical dichotomy techniques.

It must be underlined that, to verify this property, a formal model of the system which includes several automata which evolve in an asynchronous manner (formal models of PLCS, of RIOMS, ...) must be built. Building such a model that can be analysed by UPPAAL without combinatory explosion, in a short time and with reasonable computing resources is not a trivial issue.

III. MODELING METHOD DESCRIPTION

To obtain tractable models of real indutrial NAS, the method depicted figure 3 has been set up. This method includes three steps that might be automated.

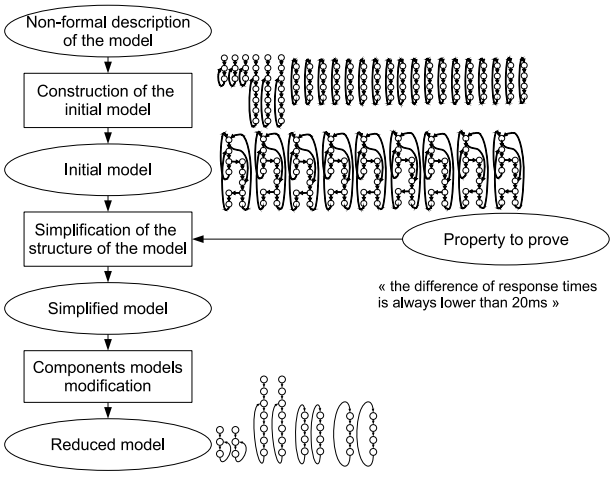

Fig. 3. Proposed method

The aim of the first step is to build a detailed model of the system, in the form of a set of communicating timed automata [7] [8]. Each timed automaton, named component model, describes precisely the behaviour of one physical component (calculus processor, communication processor, RIOM,. . .) or communication function (run by switches between one RIOM and one communication processor) and

includes parameters that represent features of this component or function, such as duration of the I/O scanning cycle or processing time. The detailed model of a given NAS can be easily obtained from a non-formal description of this system by instantiating generic components models.

The structure of a NAS model can be represented by a graph where nodes are components models and edges represent communications between these models. This representation is used by the second step of the method.

This second step aims indeed to simplify the structure of the model by keeping only the components models that introduce delays (treatment times, waiting times for resource availability or synchronization) which impact directly the property to prove; 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 [9] – or hybrid systems [10].

When the property to prove refers to a response time or a difference of response times, the components models that must be kept can be easily found: they are on the route of data, i.e. they generate, modify or propagate data (frames or variables) which are functions of the input or output events which the considered time performance is based on. Hence the principle of this step is to search, in the graph that represents the structure of the model, the shortest path, from an input event to an output event, which crosses the model of the PLC which computes the output event. Only the components models whose corresponding nodes belong to this path are retained in the simplified model.

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 behaviours, e.g. communications with removed components. The role of the third step of the method is to remove these behaviours from the remaining components models.

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

- all the locations and transitions that correspond to meaningless behaviors, e.g. treatment of requests from removed components, are deleted;
- when a transition which belongs to a concurrency structure in the detailed model is deleted, the maximal duration of the locations that follow immediatedly the remaining transitions of this structure is increased by the time of the removed concurrent treatment. This permits to obtain a reduced component model that encompasses all real behaviours, for instance treatment delay due to another concurrent treatment, even if it includes not realistic ones that correspond to a worst-case modelling, e.g. it is possible one concurrent process be always selected first, as it will be shown in the next section.

These three steps are exemplified in the following section.

IV. Application to the example

For the example which is dealt with in this paper, the structure of the initial model (fig. 4) comprises three PLC models, each one including one calculus processor model and one communication processor model, nine RIOM models and eighteen communication function models, what requires thirty three clocks for the whole system. Previous studies [11] and [12] showed that the switches can be modelled by this set of independent communication functions because the traffic due to the exchanges between PLCs and RIOMs is far lower than the maximal throughput of the Ethernet network.

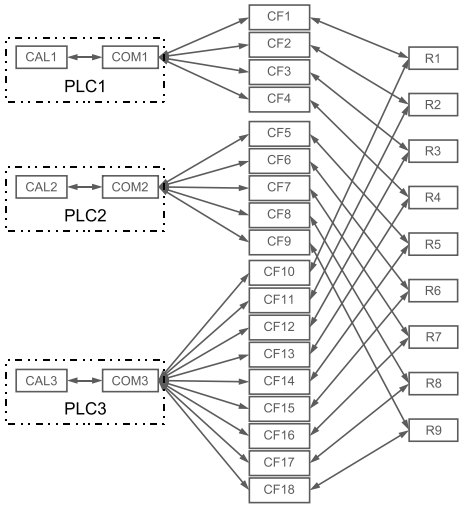

Fig. 4. Structure of the initial model

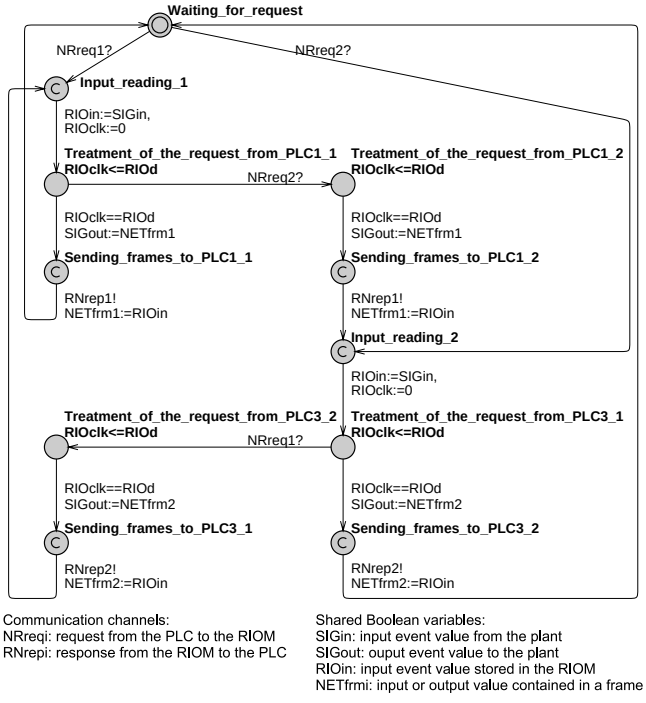

Fig. 5. Detailed UPPAAL model of a RIOM

Figure 5 shows an example of a component model: the model of a RIOM which communicates with two PLCs (PLC1 and PLC3). Formal models of RIOMs which communicate with more than two PLCs can be easily constructed

from this basis.

The response of the RIOM to a request from PLC1 is represented by three locations – ”Input reading 1”, ”Treatment of the request from PLC1 1” and ”Sending frames to PLC1 1” –. When a request from the other PLC3 occurs during the treatment of this request, the new request must be stored, what is modelled by the transition from location ”Treatment of the request from PLC1 1” to location ”Treatment of the request from PLC1 2”. The response to a request from PLC3 is modelled in the same way.

As the duration of the response to an isolated request (input reading, treatment of the request and sending frames) is much lower than the period of each I/O scanning, nearly 0.7 ms versus 10 ms or 50 ms, a request from any PLC can be delayed by one and only one request from the other PLC. This explains the structure of the automaton.

Communication with the other component models is performed both with communication channels (NRreq1?, RNrep1!, ...), for synchronisation purposes, and with shared Boolean variables (RIOin, SIGin, ...), to exchange values of input or output events.

A parameter, RIOd, is used on guards and invariants to represent the duration of the treatment of one request; then, the formal model can be specialized for different types of RIOMs.

A. Simplification of the model structure

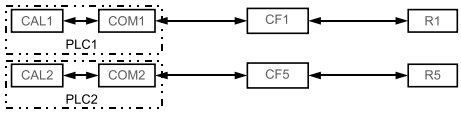

Fig. 6. Structure of the simplified model

Figure 6 shows the result of this simplification step for the case study and the property presented in subsection II-B. Only the models of the following components remain:

- RIOMs R1 and R5, which both receive the input event, send to PLCs frames that contain the value of this event, and generate, from the response frames sent by PLC1 and PLC2, the output events;
- PLCs PLC1 and PLC2, which receive frames that contain the value of the input event, process these frames and reply respectively to RIOMs R1 and R5 by sending frames that contain the values of the output events;
- communication functions CF1 and CF5 between the two models couples (R1, PLC1) and (R5, PLC2).

The other components models (PLC3 model, seven RIOMs models, and sixteen communication function models) are not kept because they are not on the route of data for the considered property. However, it is necessary to preserve the impacts of the removed components models on the behaviour of the remaining components models.

Although the removed components models have no direct action on the considered data, they can indeed impact strongly the proof results. This will be exemplified on the basis of model R1. In the initial structure (figure 4), this RIOM model communicates with two PLCs models (PLC1 and PLC3) via communication functions CF1 and CF10; hence, a request from PLC1 can be delayed by a previous request from PLC3. Removing model PLC3, and the communications from this model, from the structure without modification of the RIOM model would lead not to take into account this behaviour, which is surely a modelling flaw and would give rise to meaningless proof results.

Hence, simplification of the structure of the NAS model implies to modify the remaining components models, as explained below.

B. Modification of the components models

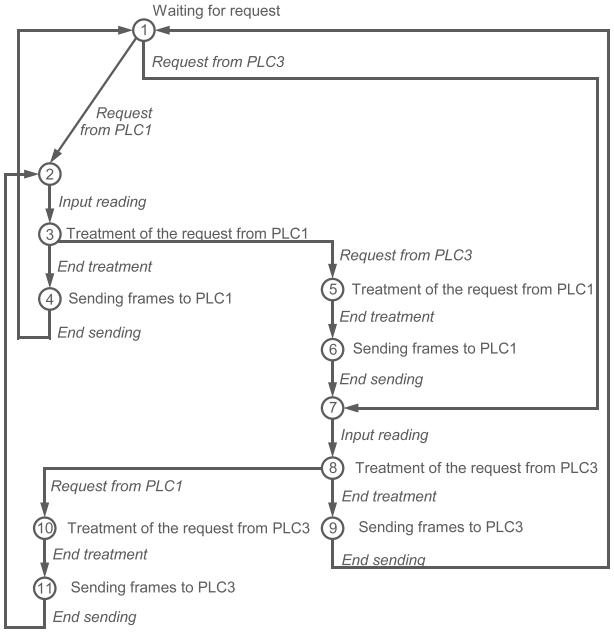

Fig. 7. Detailed model of a RIOM

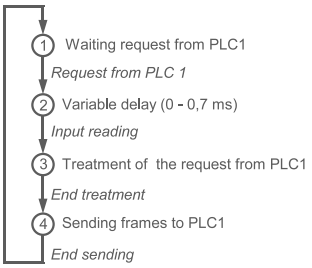

Fig. 8. Modified model of a RIOM

Components models modification will be exemplified on the basis of a RIOM model only, for room reasons. Figures 7 and 8 depict respectively the initial and modified models. For these two models, only locations and transitions are represented; the other modelling elements (guards, invariants, ...) are omitted for clarity reasons.

Comparison between the two models shows first that all the locations that represent the response to a request from PLC3 do not appear in the modified model, which is quite

normal. Moreover, concurrency between requests from the two PLCs, a behaviour that must be absolutely preserved in the modified model as explained above, is modelled in two different manners. Concurrency between requests from PLC1 and PLC3 is indeed explicitly modelled by the structure of the initial model, by two concurrent transitions starting from location 1, while it is represented by a variable delay in the modified model; the duration of location 2 may vary from a void value to the request treatment time (0.7 ms), by means of appropriate invariant and guard. Hence, the behaviour of the modified model is simpler than that of the initial model, because only the response to a request from PLC1 is modelled, but this response is sent with a variable delay, which is mandatory to obtain a meaningful model.

These two different solutions to model concurrency can lead to different time behaviours when the two I/O scanning cycles are synchronized, however. Even if this situation is not at all usual for a real system, it deserves to be discussed because it may happen when dealing with formal models. When the I/O scanning cycles are synchronized indeed, and with the numerical values of the example (I/O scanning cycle periods of PLC1 and PLC3 equal respectively to 10 ms and 50 ms, and duration of the request treatment equal to 0.7 ms), only one request from PLC1 among five can be delayed by a request from PLC3 in the initial model; if the n^{th} request is delayed, then the $(n+1)^{th}$ to $(n+4)^{th}$ requests cannot (fig. 9). This is not true for the modified model; two consecutive requests can be delayed in that case.

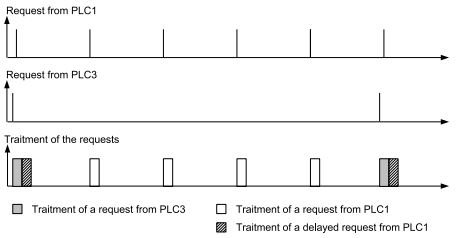

Fig. 9. Synchronized I/O scannings

Nevertheless, for the two models, the minimal and maximal values of the delay between the reception of a request from a PLC and the response to this request are the same: one time and two times the duration of the request treatment. Hence, the modified model is a *worst-case model*, in the sense that it provides the exact bounds of the particular delay which is introduced by the component for one request but can provide pessimistic bounds when several requests are considered.

When verifying the considered timed property of the architecture model (d is always lower than τ), which involves several requests to RIOMs, a positive proof on the reduced architecture model is meaningful because all the modified components models are worst-case models. On the opposite, a negative proof will require analysing the diagnostic trace so as to determine whether this result comes from a possible behaviour or not, e.g. two consecutive requests delayed for the maximal value. This will be addressed in the next section.

V. Properties proof

A. Experimental method

To verify whether the studied property (d is always lower than τ) holds (or not) on the initial and reduced models, observer automaton OBS1 has been designed (Fig. 10.a). For clarity reasons, only the structure of this automaton is depicted on this figure; invariants, guards, communication channels, shared variables are omitted.

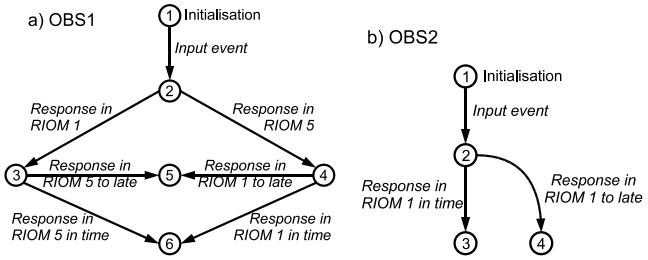

Fig. 10. Observer automata OBS1 (a) and OBS2 (b)

If the difference of response times is lower than τ, then location 6 is reached from the initial location; on the opposite, if this difference is greater than τ, location 5 is reached. Therefore, property checking is equivalent to verify whether the following statement, in the UPPAAL syntax:

$$A[]notOBS1.5 \tag{1}$$

is true, i.e. for all evolutions of the architecture model, location 5 of OBS1 is never reached.

Moreover, for the reduced model, the considered property can be checked in another manner. The structure of this model (figure 6) shows indeed that it can be decomposed in two independent sub-models, each of them including four components models. Hence, it is possible to analyse separately these sub-models and in particular to find the lower and upper bounds of the response times of each sub-model, noted respectively rt1m and rt1M, for the first sub-model (CAL1, COM1, CF1 and R1), and rt2m and rt2M, for the second one (CAL2, COM2, CF5 and R5). Then, the maximal value of the difference of response times is:

$$d = Max((rt1M - rt2m), (rt2M - rt1m)) \tag{2}$$

Four properties are to be checked, with this second solution:

- rt1M is always lower than $\tau 1$
- rt1m is always higher than $\tau 2$
- rt2M is always lower than $\tau 3$
- rt2m is always higher than $\tau 4$

where $\tau 1$, $\tau 2$, $\tau 3$, $\tau 4$ are variable parameters. This seems to increase the overall verification time. However, as the state spaces of the sub-models are smaller than the one of the global model, this is not the case, as it will be shown in the next sub-section.

Observer automaton OBS2 has been designed (Fig. 10.b) to verify the first above property; this property holds if the statement

$$A[]notOBS2.4 \tag{3}$$

is true. Similar automata and statements enable to check the other properties.

B. Results and discussion

Table II contains the results of three experiments which were made by using UPPAAL on the same computer, with a 2.8 GHz Pentium 4 processor and 4 GB of RAM, so that the computing times can be compared. The first experiment was aiming at checking the initial model. The second and third experiments were carried out with the reduced architecture model; in the second experiment, the initial form of the property (d is always lower than τ) was used, whereas the objective of the third one was to find the parameters values so that the four elementary properties of the two sub-models would be verified.

It must be also mentioned that the under- and over-approximation possibilities of UPPAAL [5], which can really reduce verification time, have not been implemented in these experiments. This work focuses only on effectiveness of the formal representation of a real system, like a NAS, and not on that of state space analysis algorithms, whatever their benefits. The overall objective is to construct effective formal models which can be analysed by different techniques. Moreover using these approximations might hinder the interpretation of the results.

TABLE II
RESULTS COMPARISON

	experiment 1	experiment 2	experiment 3
verification time	impossible	28 hours	1 second for each property
obtained results		τ = 21.4 ms	rt1m =10 ms rt1M = 21.4 ms rt2m = 10 ms rt2M = 31.4 ms d = 21.4 ms

The following conclusions can be drawn up from the obtained results:

- The initial, detailed model cannot be checked in the conditions of these experiments, for combinatory explosion reasons; the RAM size of the computer is not sufficient.
- The properties of the reduced model can be verified. In addition, diagnostic traces in case of negative proofs show that the modifications of the components models do not lead to erroneous results; negative proofs are obtained only for realistic behaviours. Then, the method that is proposed to construct reduced models yields effective models which can be used to verify timed properties and provide meaningful results.
- The two verification strategies (verification of one global property or verification of four elementary properties) yield the same result. They can be both selected.
- Nevertheless, verification of the global property with observer OBS1 is far longer than verification of the four elementary properties. This latter solution, that is only possible for the reduced model, shall be privileged because it strongly speeds up verification.

VI. CONCLUSION

This paper has presented a generic method to construct reduced timed models of networked automation systems; these models are small enough to avoid (or limit) combinatory explosion, while providing meaningful and trustworthy results, when they are checked. Construction of a reduced model includes two steps. First, the structure of the model is simplified, according to the property which must be proved: the models of components which are not on the route of data are removed. Then the remaining components models are modified to integrate the influences of the removed components models, in the form of worst-case behaviours.

It has been also shown that a complex property can be decomposed in a set of simpler elementary properties, when using a reduced model. Comparison of the experimental results obtained with detailed and reduced models shows clearly the effectiveness of the reduced model. Verification of a complex timed property on a non-trivial system is possible in a short time.

On-going works are aiming at developing a library of components models for different kinds of networks and at automating the method; the overall objective of these works is automatic construction of the reduced architecture model, once the property to prove is given. A second outlook is to combine our approach, which focuses on improvement of the effectiveness of models, and approximation techniques, to fully benefit from these two approaches.

REFERENCES

[1] N. Pereira, E. Tovar, and L. M. Pinho, "Timeliness in COTS factory-floor distributed systems: what role for simulation?" in *Proc. of 9th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Sept. 2004.

[2] B. Bordbar and R. Anane, "An architecture for automated QoS resolution in wireless systems," in *AINA '05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 774–779.

[3] F. W. Vaandrager and A. L. de Groot, "Analysis of a biphase mark protocol with Uppaal and PVS," *Form. Asp. Comput.*, vol. 18, no. 4, pp. 433–458, 2006.

[4] R. B. Salah, M. Bozga, and O. Maler, *CONCUR 2006 Concurrency Theory*, 2006, ch. On Interleaving in Timed Automata.

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

[6] J. Greifeneider and G. Frey, "Probabilistic timed automata for modeling networked automation systems," in *proc. 1st IFAC DCDS, Cachan*, 2007, pp. 143 – 148.

[7] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[8] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, 1994.

[9] D. Kroening, "Computing over-approximations with bounded model checking," *Electronic Notes in Theoretical Computer Science*, 2006.

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

[11] G. Marsal, "Evaluation of time performances of Ethernet-based automation systems by simulation of high-level Petri nets," Ph.D. dissertation, ENS Cachan (France) and Kaiserslautern University (Germany), december 2006.

[12] B. Denis, S. Ruel, J.-M. Faure, G. Marsal, and G. Frey, "Measuring the impact of vertical integration on response times in Ethernet fieldbuses," in *Proc. of ETFA'07*, 2007.