PERFORMANCE ANALYSIS OF INDUSTRIAL ETHERNET NETWORKS BY MEANS OF TIMED MODEL-CHECKING

Daniel Witsch, Birgit Vogel-Heuser, Jean-Marc Faure, Gaëlle Marsal

PERFORMANCE ANALYSIS OF INDUSTRIAL ETHERNET NETWORKS BY MEANS OF TIMED MODEL-CHECKING

Daniel Witsch[1], Birgit Vogel-Heuser[1], Jean-Marc Faure[2], Gaëlle Marsal[2]

[1] *University of Wuppertal, Germany*
{witsch; bvogel} @uni-wuppertal.de

[2] *Ecole Normale Superieure de Cachan, France*
{faure; gaelle.marsal} @lurpa.ens-cachan.fr

Abstract: Ethernet networks are promising for the harmonization of the communication technologies in manufacturing automation but they have not been specifically intended for industrial control applications. Investigations have thus become necessary to evaluate their performance. Most analysis approaches use probabilistic models to validate the system's behavior. However, if the deterministic behavior of a system needs to be ensured exhaustively, formal verification techniques, like model-checking are more appropriate. Unfortunately the state-space explosion problem constitutes a serious obstacle. Considering a real-time Ethernet network as a case-study, this paper describes sophisticated modeling-techniques, which help to alleviate the state-space explosion, for the timed model-checker Uppaal.

Keywords: Ethernet, Queuing network models, Real-time communication, Formal verification

1. INTRODUCTION

Ethernet networks are promising for the harmonization of the communication technologies in manufacturing automation. Indeed, heterogeneity between the various levels of the communication hierarchy causes problems of incompatibility, which complicate the exchange of information throughout the automation pyramid. The use of a single network could overcome the limits of current systems (Lo Bello and Mirabella, 2001).

Common approaches towards the implementation of a real-time Ethernet solution can be classified in three different categories (Jasperneite, 2005). The Ethernet solutions belonging to class one make use of the common TCP/IP protocol-stack. Only an automation specific application protocol is added. But passing the TCP/IP stack introduces drastic delays to messages. Therefore, this solution is only used for soft real-time data with time frames larger than 100ms. This solution is implemented for example in Profinet Non-RT (Profinet V1), ModBus/IDA, Ethernet/IP. The second class is formed by those solutions, which do not use the standard TCP/IP stack for the exchange of real-time data. Through the reduction of the protocol stack, real-time messages can be exchanged within a time-frame of about 10ms. In order to guarantee that non-real-time traffic does not disturb the real-time traffic, the priority mechanisms described in IEEE 802.1Q/D are applied. This idea is realized (among others) by Profinet SRT (Profinet V2) and the Modbus I/O-scanning mechanism. Class three of the real-time Ethernet concepts introduces proprietary scheduling mechanisms and hardware in order to reach high performance. Ethernet concepts like Profinet IRT (Profinet V3), Powerlink, EtherCAT can exchange real-time messages with delays smaller than one millisecond. As a drawback, these implementations require special, proprietary Ethernet devices. So it can be stated that high performance and determinism

can be obtained for the price of proprietary solutions. Therefore, if the harmonization of heterogeneous communication structures with a convenient performance is an aim Ethernet implementations belonging to the second class are promising.

However, the roots of these Ethernet solutions still lay in office communication. As a consequence they have not been specifically intended for industrial control applications. The physical architecture and the behavior of the networks obtained are quite different of those of specific fieldbusses. The use of IP protocols and Ethernet implies to add active components in the network such as switches. Moreover, IP-based communication is based on client-server relationships in contrast to specific fieldbusses where often master-slave structures are used. A number of studies and investigations have thus become necessary to evaluate the performance of the control system in order to check if it meets the requirements of industrial control applications such as manufacturing automation systems.

Such automation systems can be roughly structured in three parts: a control level, a process level and the network connecting these levels. Within this paper the network includes Ethernet-switches, I/O-modules and PLC Ethernet couplers. This communication system forms the scope of the performance analysis within this paper (Fig. 1, dashed rectangle). Within this network, two PLCs communicate with five I/O-modules each, whereby two I/O-modules communicate with both PLCs (Fig. 1 middle). Data-exchange between the PLCs and the I/O-modules is initiated by the PLCs. They demand process-data periodically from the I/O-modules. One I/O-module can be polled from different PLCs, simultaneously. This occurs for example in a manufacturing process when workpieces are passed by means of a pick-and-place unit between two parts of a plant, which are controlled by their own PLC (Fig. 1, process). For communication, the Modbus I/O-scanning service, as a member of the second class of industrial Ethernet solutions, is used and analyzed.

Most analysis approaches dealing with Ethernet networks use simulation techniques to analyze the system's behavior. However, a simulation is not exhaustive, i.e. it does not cover all possible cases. Therefore, if the deterministic behavior of a system needs to be ensured, simulation is not sufficient. However, model-checking - as an exhaustive model-analysis technique - can provide unambiguous results. This technique implies to define both, the specification of the system and the properties to be checked. The Model-checker returns whether the system fulfills those properties in all possible cases or not. If the system violates the properties defined, the model-checker returns an error trace. Due to the so-called state space-explosion problem model-checking is strongly restricted regarding the model's complexity. The term state-space explosion describes the circumstance that the state space of transition systems grows exponentially with the size of the model. Therefore raising the system's complexity leads to an enormous amount of states. This behavior turns into a problem if the size of the state space cannot be handled anymore by the computing system available or calculated within an appropriate time frame. Alleviating this obstacle and by this enabling the application of model-checking for analysis purposes in an industrial context, constitutes the main challenge regarding model-checking.

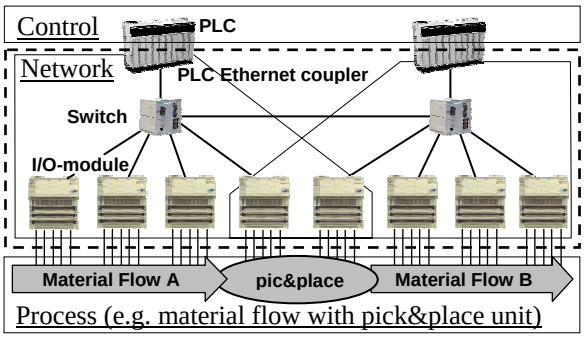

Fig. 1 Context and system within consideration

2. UPPAAL MODELING TECHNIQUES

Uppaal is an integrated toolset for the modeling, simulation and verification of real-time systems. It offers an intuitive graphical user interface for modeling, simulation and verification. It belongs to the class of timed, symbolic model-checkers and uses timed automata (Alur and Dill, 1990) for modeling purposes. The properties to verify, have to be formulated in Computation Tree Logic (CTL). The algorithms implemented in Uppaal use a symbolic model-representation in order to perform model-checking efficiently.

2.1 Known Modeling Techniques

Uppaal is based on an asynchronous execution model, i.e. several edges in different automata can be executed simultaneously. During the verification process, all possible combinations that result from these interleavings have to be considered. This leads to a drastic enlargement of the model's state space. Thus, one central concept to reduce the state space is the intensive usage of so-called committed locations. Marking a location as committed, (which is indicated by a C in the center of the location) means that time cannot pass while this location is active and no interleaving with other automata will occur (Behrmann, et al., 2004).

Generally, there are at least two ways to reduce the model's size. One principle to cope with the size, follows the divide and conquer strategy. The idea is to split up the whole system into independent system parts, which can be verified separately. When all parts of the system are verified subsequently, it can be assumed that the functionality of the entire system is assured. In (Giese, et al., 2003) and (Hirsch, 2004) such a successive component based verification approach is derived. Two main difficulties emerge when applying this strategy. It is difficult to find independent partial systems and it has to be verified that it is sufficient to verify partial systems in order

to verify the whole system because combinations of correct working components can also lead to faulty behavior.

Another more general principle, which needs to be applied in all cases, is abstraction. The main difficulty is, to decide which details can be neglected or subsumed and which have an important influence to the system's behavior. It is difficult to formulate well-considered abstractions exactly or in a way, the model-checker is able to calculate the model efficiently. To ease this process, modeling patterns provide useful expertise. In (Behrmann, et al., 2004) several modeling pattern for Uppaal are described. Furthermore, the size of the state-space, which has to be explored in order to verify the model, is crucial for the computability of model-checking problems. The size of the state-space is a result of all possible variable combinations. Therefore, if an action is modeled to be executed at an arbitrary point in time all these possibilities have to be taken into account during the verification process. This leads inevitably to state explosion in non-trivial models. So it is necessary to narrow such degrees of freedom in the model. Therefore, all edges should be modeled to fire within well-defined time intervals, ideally at one distinct point in time. There are at least three different possibilities to force an edge to fire at a certain point in time. An edge can be executed at an absolute point in time, as soon as it becomes enabled or synchronized with another edge, which has to be in turn executed at an absolute point in time or as fast as possible.

Unfortunately applying these timing restrictions and the modeling patterns is not sufficient to obtain computable models. More effort has to be made in order to meet the special algorithmic properties of the model-checker. A phenomenon peculiar for symbolic real-time model-checking is that the repetition of control situations (e.g. for-loops, case) in the exploration of the model can lead to an unfavorable segmentation of the state space's data structure (Möller, 2002). Based on this observation, techniques were developed to solve this problem and by this speed up the verification drastically. This circumstance is exploited by the approximation techniques presented by Möller and in (Hendriks and Larsen, 2002). As a drawback, these approximation techniques cannot be used together with important modeling patterns. Therefore they can only be applied in special cases.

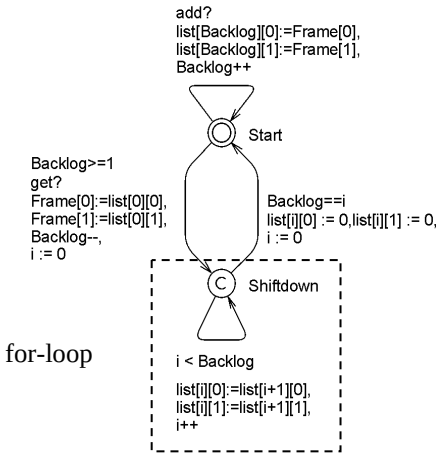

Fig. 2 Standard Buffer Automaton [BDL04]

2.2. Developed Modeling Guideline

Considering all commonly known modeling guidelines for Uppaal, it was not possible to handle even a relatively small model of an Ethernet control system containing two switches, two Ethernet couplers and eight I/O modules. Hence other modeling techniques were needed. Keeping in mind the problem of segmented state-spaces caused by control structures and by comparing the verification time of a subsequently changed model along numerous experiments, the following modeling rationale was developed.

Counter-variables (e.g. from control structures, like for-loops) should be expressed by a chain of committed locations, even though this leads to more locations or edges.

This guideline will be illustrated on an automaton modeling a buffer (Fig. 2). Due to the fact that a buffer realizes the input-side of each Ethernet port, this is the automaton which occurs most frequently in the system. Even though this automaton is relatively simple, small changes will lead to significant differences in the verification time. The buffer automaton represents a FIFO queue. It can carry five[1] elements.

Elements of this buffer are simplified Ethernet-frames, which are implemented as a two dimensional array, where *Frame[0]* stands for the source address and *Frame[1]* for the destination address of the Ethernet frame. Reading or writing to a buffer is not time-consuming. Writing to this buffer can be performed by copying the new value to the variable *Frame* and subsequently activating the *add?* synchronization. By setting the *get?* synchronization, the first Ethernet-frame is copied to the variable *Frame* and all following elements in the buffer shift down one place (see for-loop in Fig. 2) subsequently. For this shiftdown, a bounded integer variable is used. The variable *Backlog* indicates how many messages reside in the buffer.

The verification process of the whole system, which contains among others 16 of these buffer automata took 81 minutes[2]. The shiftdown algorithm implemented in the buffer is based on a for-loop. In Fig. 3 an adapted version of the buffer is given according to the modeling rationale mentioned above. The for-loop was replaced by a chain of committed locations, i.e. each iteration of the for-loop is modeled by one committed location. Hence, this new buffer automaton contains three more locations and seven more edges but no counter variables. Because the only difference between these

[1] However, this factor does not constitute a loss of generality.

[2] Calculated with an Intel Xeon, 2,6GHz, 1GB RAM.

two automata is, that the for-loop is modeled explicitly in the optimized automaton, it can be asserted that their functionality is identical.
In fact, the verification of the system with the adapted buffer automaton can be verified within 4:30 minutes instead of 81 minutes. The same behavior regarding the verification performance was obtained with other automata. So it can be assumed, that this constitutes a general modeling rationale. As a drawback, this modeling principle leads to models, which are more difficult to extend and to parameterize. For example, if the buffer described above, needs to be extended in its size, in the primal version, this can be done by simply changing a parameter value. Using the proposed modeling principle, new locations and new edges have to be introduced. This may lead to the necessity to create more slightly different model templates, where one template with different parameterizations was sufficient before. However, this effort is reasonable in the case studied here.

3. OPTIMIZED UPPAAL MODELS FOR ETHERNET NETWORK COMPONENTS

The observed system is an industrial switched Ethernet Network including I/O-modules, which couple the physical process to the network, PLC Ethernet devices, which connect the network to the controlling devices and store-and-forward Ethernet switches. The protocol considered is Open Modbus TCP, a client-server protocol designed for soft real-time communication between the control and the field level (Schneider, 2004). Consequently, data-exchange between the PLCs and the I/O-modules is initiated by the PLCs. They demand process-data periodically from the I/O-modules. One I/O-module can be polled from different PLCs, simultaneously.
Objective of the analysis is to compare the different network architectures (e.g. a line-, star-, or tree-architecture with a given count of I/O-modules and PLC-Ethernet couplers) regarding their performance. The indicator for the performance is chosen according to the special needs of determinism of real-time Ethernet solutions in an industrial context. Therefore model-checking is used to prove for a given architecture that the answers to each polling

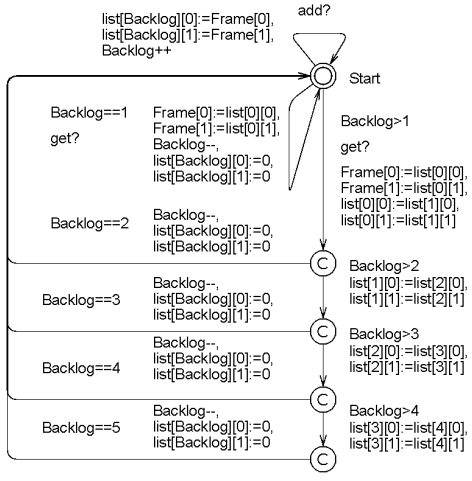

Fig. 3 Adapted Uppaal Buffer Automaton

cycle, initiated by each PLC, returns before the next cycle is executed, i.e. the polling cycles configured by the user can be hold under all circumstances (quality of service). Such circumstances are for example the increase of the response-time which results from the blocking behavior of a simultaneously polled I/O-module.

To obtain these results, each network component was modeled within the model-checker Uppaal according to the modeling guidelines mentioned in section two. The network components considered (switches, PLC-Ethernet couplers and I/O-modules), consist of at least two subcomponents, an input-buffer and a processor, which realizes the main function of the network component. The input-side of each network component is implemented as a buffer. Its Uppaal model is given in Fig. 3. For each network component a single delay value is used to model its response-time behavior, which represents a worst-case approximation including all occurring delays within a component.

3.1 The Switch Automaton

The switches modeled, provide eight ports. Each switch contains one single buffer for all input-ports and the dispatcher, which fulfills the switching function. The dispatcher automaton (Fig. 4) of the switch detects if a messages resides in the buffer. Starting with the first message arrived, the message's target-address is extracted. The delay d_{SW} stands for this procedure. Afterwards the dispatcher automaton forwards the message to the switchport, where the target address can be found. This target can be a terminal device, which is directly connected to the switch or the input-port of another switch, if the addressed component is not reachable directly. Thanks to this mechanism, switches can be arbitrarily cascaded. The sending rate of the switch is given to $r_{SW} = 1/d_{SW}$. The n-th element in the buffer is forwarded with a delay of $t_{SW,n} = n \cdot d_{SW}$.

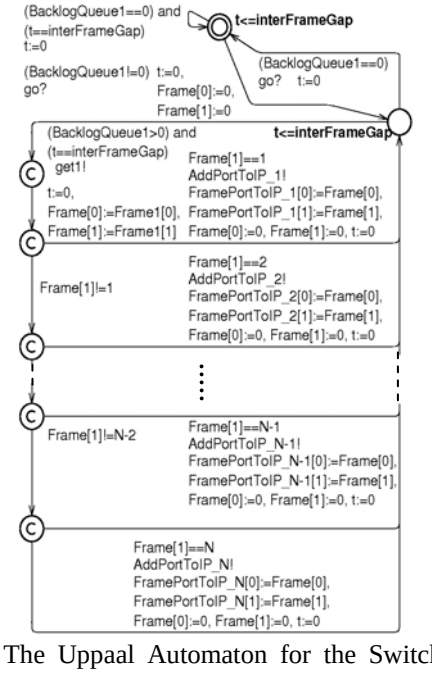

Fig. 4 The Uppaal Automaton for the Switch (the Variable N stands for the number of addressable Network components)

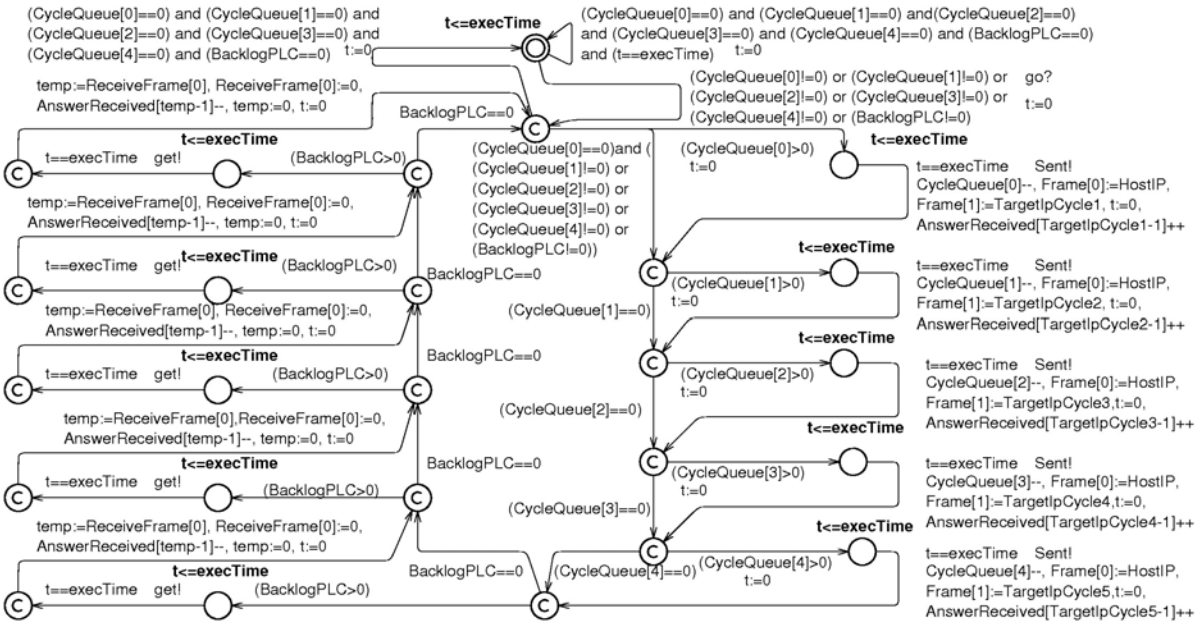

Fig. 5 Uppaal Automaton for the PLC's Ethernet coupler

3.2 The I/O-Module Automaton

An I/O-Module represents a terminal device, which collects digital process data and sends them as one Ethernet frame back to the PLC, which requested the data. This process takes the time d_{IOM}. An I/O-module contains one buffer at its input and another component called data-engine, which realizes the main functionality. Due to the fact that one I/O module can only be connected to one switch, which can in turn send two messages with a minimum gap of d_{SW}, it can be assured that two messages arrive at one I/O-Module with a minimum $\Delta t = (t_2 - t_1) \geq d_{SW}$. Therefore, the answer to a second message which arrives within $(t_1 + d_{SW}) < t_2 < (t_1 + d_{IOM})$ will be sent from the I/O-module at $t_{A,2} = t_1 + 2d_{IOM}$. The sending rate of an I/O-module is given to $r_{IOM} = 1 / d_{IOM}$.

3.3 The PLC's Ethernet Coupler Automaton

The PLC's Ethernet coupler (Fig. 5) periodically sends a burst of N (which corresponds to the number of scanned I/O-modules of this PLC) messages to the different I/O-modules within its scope. This scanning period t_{SC} is configured by the user. Moreover, processing one message (sending or receiving) takes the time d_{PLC}. For the PLC's Ethernet coupler it is assumed that the treatment of incoming and outgoing messages is handled by one CPU, so its maximal sending-rate depends on the frequency of incoming messages and vice versa. For the correct behavior of the system, it has to be assured that treating incoming messages from scancycle $k-1$ do not disturb sending messages in cycle k. Hence, all answers to scancycles $k-1$ have to be processed by the PLC's Ethernet coupler before scancycle k is initialized. If the value for t_{SC} is chosen conveniently high, no message will be sent within $N \cdot d_{PLC} < t < t_{SC}$. So the sending process will not be disturbed by the receiving process and vice versa. Therefore the sending rate of a PLC is given to $r_{PLC} = 1 / d_{PLC}$. Consequently, the receiving rate equals the sending rate. If the value for t_{SC} is too small undesired behavior occurs. This will be detected during the formal verification.

3.4 Performance Analysis Results

Fig. 6 shows the results of the performance analysis through model-checking of three different network topologies, whereby all architectures contain the same number of PLC Ethernet devices and I/O-modules and the same I/O-scanning configuration. The results in Fig. 6 show three different areas. The leftmost part of each diagram (dark grey) contains all configurations of scanning-times of PLC1 and PLC2 which lead to undesired behavior. The rightmost part of each diagram (white) shows all configurations where only desired behavior occurs. Between these two areas the border is fuzzy. In the light grey part in the middle of the diagrams, the results toggle due to beneficial synchronizations among the single processes. But such exact synchronizations cannot be realized in a real network, so these points represent only theoretical values. Regarding the performance analysis it can be stated that a topology with less switches is beneficial.

These results were validated against a simulation, which was in turn validated against measures on a real network. This indirect validation had to be used because, it was not possible to configure the hardware available in a way that undesired behavior can occur. In order to obtain undesired behavior it would be necessary to an Ethernet network with more than 30 I/O-modules (Schneider, 2004). Such a number of I/O-modules was not available. Another possibility would have been to configure polling cycles with the values given in Fig. 6 (t_{SC} < 5ms). Due to the restriction of the engineering software the fastest scanning periods are 10ms. Here the simulation developed by Poulard (Poulard, et al., 2004) offered much more flexibility. The comparison between the simulation results and the results obtained by means of model-checking showed a very good coherence.

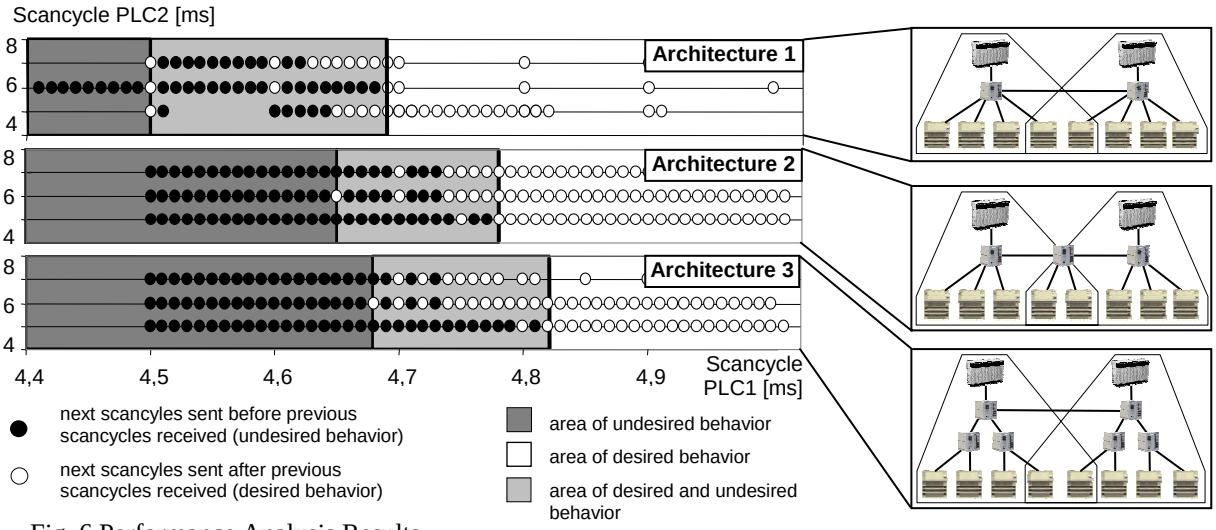

Fig. 6 Performance Analysis Results

4. SUMMARY AND OUTLOOK

This paper presented commonly known and new modeling techniques which can be used in order to cope with the state-space-explosion problem. Further, optimized Uppaal models for standard Ethernet network components and their according real-time behavior were given. Finally the developed models were applied within a case-study, where the real-time capability of a standard Ethernet network with client-server communication was discussed.

The results of the performance analysis are surely not new nor very surprising. But the way these results were obtained is new and offers another quality compared to those values obtained by non exhaustive analysis processes like simulations. In fact the more interesting result is that it is possible to verify models, which have a degree of complexity similar to other analysis-models, like simulations.

Regarding the real-time performance of Ethernet, model-checking can be used complementary to simulation approaches like discussed by Jasperneite (Jasperneite, 2002) or Poulard. But the area of application of model-checking and the discussed modeling guidelines is not restricted to a certain domain. Therefore this paper showed also that and how model-checking can be used in order to verify the real-time behavior of critical systems, which occur usually in the field of manufacturing automation.

5. REFERENCES

Alur, R., D.L. Dill (1990). *Automata for modeling real-time systems. In: Proc. of Int. Colloquium on Algorithms, Languages, and Programming, volume **443** of LNCS*, pages 322–335.

Behrmann, G., A. David and K.G. Larsen (2004). *A Tutorial on Uppaal*, Department of Computer Science, Aalborg University, Denmark.

Giese, H., M. Tichy, S. Burmester, W. Schäfer and S. Flake (2003). *Towards the Compositional Verification of Real-Time UML Designs. In Proceedings of ESEC/FSE'03*, September 1–5, 2003, Helsinki, Finland, ACM Press, New York, NY, USA.

Hirsch, M. (2004). *Effizientes Model Checking von UML-RT Modellen und Realtime Statecharts mit UPPAAL*, Diploma Thesis Universität Paderborn, Germany.

Hendriks, M., K.G. Larsen (2002). *Exact acceleration of real-time model checking. In Theory and Practice of Timed Systems*, volume **65** of Electronic Notes in Theoretical Computer Science. Elsevier Science Publishers, Oxford, England.

Jasperneite, J. (2002). *Performance Evaluation of a Class-of-service based Local Area Network for using at the Field device level*. Dissertation, IFAK Magdeburg, Germany, Shaker, Aachen, Germany.

Jasperneite, J. (2005): *Echtzeit-Ethernet im Überblick.* In atp - Automatisierungstechnische Praxis, Oldenbourg, No. **3**, pp. 29 - 34.

Lo Bello, L., O. Mirabella (2001). *Design Issues for Ethernet in Automation.* In: Proc. of ETFA'2001, 8th IEEE International Conference on Emerging Technologies and Factory Automation, Antibes Juan-Les-Pins, France.

Madl, G., S. Abdelwahed and D.C. Schmidt (2005). *Verifying Distributed Real-time Properties of Embedded Systems via Graph Transformations and Model Checking*, International Journal of Time-Critical Computing Systems, invited paper, accepted.

Möller, M.O. (2002). *Structure and Hierarchy in Real-Time Systems*, Dissertation at the University of Aarhus, Denmark.

Poulard, G., B. Denis, J.M. Faure. (2004) *Modélisation par réseau de Petri coloré des architectures de commande distribuées sur réseau de terrain Ethernet et TCP/IP*. 5ème Conférence francophone de MOdélisation et SIMulation, MOSIM04, Nantes (France), pp. 405-412.

Schneider Electric (2004). *Telemechanique: Automation and control, Ethernet TCP/IP and Web technologies Catalogue.* Available online: http://coffer.elmatik.ee/info/kataloogid/Telemecanique/Ethernet%20TCPIP%20and%20Web%20technologies%20-%2007.04.pdf (27/06/05).