Reactivity Analysis of different Networked Automation System Architectures

Jürgen Greifeneder und Georg Frey
University of Kaiserslautern
Erwin-Schrödinger-Straße 12, 67663 Kaiserslautern
{greifeneder, frey}@eit.uni-kl.de

Abstract

The reactivity of Networked Automation Systems (NAS) has direct influence on safety and quality aspects. It can be determined by a response time analysis, which itself can be calculated using probabilistic model checking (PMC). The analysis of a NAS has to account for several coupled components, each of them exhibiting its own behavior. Consequently, in this article the influences of (1) the controller's execution behavior including its network interface card (cyclic vs. interrupt based) and (2) the communication model (Producer-Consumer vs. Client-Server) on the response time behavior will be analyzed. For this purpose the systems are represented by abstract models using probabilistic timed automata. Probabilistic model checking is used to compare the reactivity of different possible combinations of the basic behaviors.

1. Introduction

Nowadays, control systems typically are realized as a superposition of several time driven processes (e.g. PLC, IEC 61131). In contrary thereto in embedded systems normally event driven (interrupt based) processes are used. The same is true for some modern control architectures especially those in the distributed systems area (IEC 61499). However, particularly in the latter area, there are also mixed architecture as e.g. time driven implementations of IEC 61499 in IsaGraF.

Designing a control system, not only the functional but also the temporal aspects must be considered. That's because information not arriving within given time bounds mostly has the same consequences as lost information. For this reason it is necessary to have a very precise knowledge of occurring delays.

The use of Networked Automation Systems (NAS) instead of traditional, directly wired systems, leads to a situation which necessitates the consideration of a multitude of effects. Due to those effects significant delays may occur. As the worst delays only occur with a very small probability, the use of worst-case-analytic-methods would direct to unwarranted high hardware requirements. Fortunately, it is sufficient for most applications to meet temporal boundaries most of the time, e.g. in 99.9% of all cases.

By determining the overall response time probability distribution instead of only the boundary values has the advantage that the result can be used for the analysis of quality aspects directly [1]. It can be shown, that not only the speed of the components, but also the wiring and the functional architecture do have crucial influence on the overall system's reactivity.

Consequently, the reactivity of different Networked Automation System's architectures will be analyzed in this article. The thereby discussed systems have several input signals I_i, each of them with a possibly varying triggering probability. If an input signal got activated, the PLC has to execute a predefined action f_i in which's curse the output O_i (i.e. the input value of an actuator) will be changed. The corresponding analytical interest lies in the time necessary to activate the output O_i counting from the triggering moment of the input signal I_i. This delay is called response time.

To do so, not only the static system has to be discussed. The types of signals- and processes dealt with have to be considered too. While the input part usually is event driven, the process part shows periodically recurring processes, as e.g. cyclic control algorithms.

A lot of work has been done in the area of network analysis already, e.g. [2, 3, 4, 5]. However, only very few of them did address the whole control loop and most of them were focused on a specific bus protocol (as e.g. ModBus-TCP or CAN), whereas in this work the behavior itself is used abstracted from the concrete implementation towards its principal behavior.

In section 2 four possible architectures are introduced and the corresponding characteristics are discussed. In section 3 the structure of an example is introduced. Based on this structure section 4 explains the response time analysis. Section 5 gives the corresponding modeling aspects and section 6 the response time distributions, the reactivities and the characteristics of the four architectures introduced in section 2. The underlying analysis has been done, using probabilistic model checking (PMC, [6]). Section 7 finally comprises those results in short, while section 8 gives a summary and an outlook on future works.

1-4244-1506-3/08/$25.00 ©2008 IEEE

2. Basic architectures

The response time analysis is done for the four architectures shown in Table 1. Those four architectures result from the differentiation of event and time based behavior on the one hand and the consideration of the communication- as well as control-level on the other hand. The communication thereby is distinguished between a Client-Server-model (e.g. ModBus TCP) and a Producer-Consumer-model (e.g. CAN). Note that the Producer-Consumer-model is also known as Publish-Subscriber-model. For the control the execution procedure is decisive which may either be interrupt based (e.g. Microcontroller-based systems) or cycle based (e.g. Programmable Logic Controllers, PLCs). For simplicity in the following the term PLC is used for all controllers regardless of their execution model.

Table 1. Basic behavior modes of control systems

	Communication	Control
event driven	Producer/Consumer	Interrupt
time driven	Client/Server	PLC-cycle

While in the Client-Server-model the I/O-modules are requested cyclically – which results in considerable delays due to synchronization [7] – the Producer-Consumer-model is event driven, i.e. messages will be sent as soon as they are generated, i.e. as soon as there is new information available to be sent. The same seems to be true for the Control-level, as in an interrupt based execution the information has to wait for processing only, if the controller is busy, i.e. still handling an information that arrived before, while in the cyclic execution waiting delays occur in most cases.

Besides this signal registration the result of the execution must be transferred to the corresponding actuator. Thereby it also can be distinguished between a cyclic and an event driven structure. While it makes less sense, to couple an event-based sensor's value capturing with cyclic actuator activation, the reverse case is indeed interesting and suitable in the real world.

Consequently the six architectures shown in Table 2 are investigated. For those six, the response time behavior of a reference example is analyzed in more detail in the following sections.

Note: Obviously, there are some more combinations conceivable. For example different cases of synchronization between PLC and PLC-I/O which would lead towards a behavior in between the cases 1c and 2 or 2 and 4i/c, respectively. Moreover, most realizations of D/A, A/D converters are based on cyclic processing. However, their cycle times normally are much faster than the smallest time resolution used in this paper and therefore are neglected. For an example of a system with cyclic behavior in the field-I/O, the network transmission and the PLC, it is referred to the multi-rate system discussed in [8]. A multi-rate system thereby is defined as a system, including different sampling or cycle times.

Table 2. Meaningful basic architectures of control systems

Case	PLC	Field-I/O	PLC-I/O	
			Sensor	Actuator
1 c	Cyclic	on request	Cyclic	
1 i	Cyclic	on request	Cyclic	Immediately
2	Cyclic	Producer-Consumer		immediately
3	interrupt	Producer-Consumer		immediately
4 c	interrupt	on request	cyclic	
4 i	interrupt	on request	Cyclic	immediately

3. System Description

For the comparative analysis a typical example from the area of Networked Automation Systems is used as shown in Figure 1. The parameters used are derived from real system parameters, but shall only be dealt in an academic way, in here.

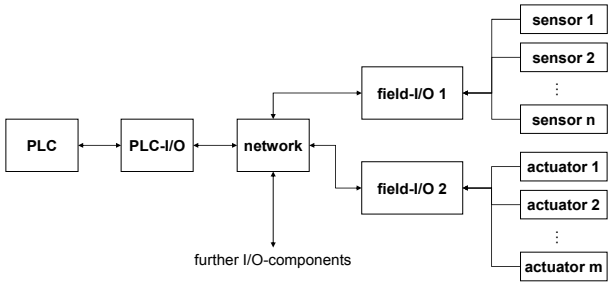

Figure 1. Case Study for the comparative analysis. Note: It is possible but not necessary to separate sensors and actuators.

On the left hand side the behavior of the PLC is presented, which is given by a cycle time of (constant) 10 ms in the cyclic case and a variable execution time in the event based case.

If a sensor's information arrives while the PLC is busy, this information has to wait until the PLC returns to its state „read inputs". While this is mostly the case for the cycle based operation, in the event based one it only occurs if other sensors values have triggered a PLC-execution just before. The execution sequence thereby generally contains one time slot each for reading the inputs and for writing the outputs. Both time slots are assumed to last 1 ms. The cycle based PLC runs an execution algorithm which contains the execution code for all (here: 5) sensors and needs a processing time of 8 ms.

The event based PLC on the other hand runs an independent code block for each sensor. Such a block only will be executed if the corresponding sensor provides information. For the execution of each independent block 2 ms are necessary, while it is assumed, that all possible values of a sensor will be handled by the same program block or need at least the same time respectively. While the cycle based PLC processes its cycle continuously, the event based PLC starts an execution only if information from one of the sensors is provided. After finishing an execution the latter PLC therefore changes to an idle state, where it remains until (at least) one of the sensors provides information.

The functionality of the PLC-I/O is much simpler. In the Producer-Consumer-case it waits for arriving sensor values and writes them immediately into the PLC's input cache. This process keeps the PLC-I/O busy for 1 ms. In the case of an immediate output transfer, the PLC-I/O passes the PLC's output values immediately (respectively after a processing time of 1 ms) to the network. In the case of a cycle based sensor's value capturing the PLC-I/O requests the associated field-I/Os every 8 ms (Client-Server-mode). The received sensor values will be passed to the PLC at the end of each PLC-I/O's cycle, i.e. the PLC receives actual values every 8 ms. The cycle based PLC's execution time therefore is always larger than the input refresh rate; and it is mostly larger in the case of an event based PLC's behavior. By choosing the sensor value's hold time to 16 ms it is ensured that a) each sensor value will be registered and b) each sensor value will be processed before it can be overwritten by a new one. In the case of a cyclic output the information is passed to the network together with the requests for new sensor values, i.e. the time at which the actuator information is send is directly dependant of the (cyclically recurring) sensor's values request sending time.

Besides the hold time of a sensor value its change probability p is important, i.e. that parameter which gives the probability that the sensor's value changes within a given time interval. If only one sensor is discussed, this probability has no influence, as the response time will be measured starting with the change and will not be influenced by later entities of the sensor value's behavior any more. If more than one sensor is considered, the probability p is a measure for the probability that several sensor values change within a short period of time. In the example up to five sensors are considered. The change probability is chosen to 10 and 60 changes per second (by using a discretization step width of 1 ms this results in a change probability of $p=0.01$ respectively $p=0.06$).

The transfer over the network will be represented by a constant delay of 2 ms within this paper. This value corresponds to the average value of a TCP/IP-transmission as shown by measurements at the authors' institute [9]. For the use of a variable network transmission time it is referred to [10], for a discussion on the influence of transmission failures to [7]. It is possible to attach further PLC-I/Os and field-I/Os to the network. Their influence on the network transfer time can be neglected [11] but additional PLC-I/Os accessing the same field-I/O will induce waiting times.

The field-I/O has a double role. First, it records the sensor values and transfers them over the network. Second, it handles the information passing to the actuators. Thereby it is assumed that the information from the sensors does not interfere with that one passed to the actuators. This could be guaranteed for example by using different field-I/Os. The field-I/O needs 2 ms to receive, process and pass information to an actuator. The same value is used for the recording of the sensor values, including reading, packaging and sending. The field-I/O owns two waiting queues: The first one located on the network side. This will be neglected in the frame of this paper as only one PLC-I/O will send requests to the field-I/O. For examples dealing with access conflicts, it is referred to [7]. The second waiting queue is located on the sensor or process side and is necessary to buffer a sensor signal until the field-I/O is able to process it. While this is not necessary in the Client-Server-Case (as all sensor values will be requested at the same time and consequently sent in the same telegram), waiting times can occur in the Producer-Consumer-Case as the field-I/O processes each sensor's value separately.

Based on this structure, in the following a response time analysis will be done for the six system architectures discussed in Table 2.

4. Response Time Analysis

The response time of a Networked Automation System (NAS) is defined as the time which will elapse starting from an input change until all corresponding algorithms have been executed and the subsequent outputs have all been activated. For the analysis of response times several possible methods and approaches are available. However, as shown in [12], only simulation and probabilistic model checking (PMC, [6]) are suitable for the analysis of NAS. For most systems PMC is much faster and obviously much more precise (i.e. exact) as simulative approaches. The main advantages of simulation on the other hand is that a wide range of graphical tools is available and with simulation continuous systems can be modeled, while this is not possible using PMC. For more details on the use of PMC for the temporal analysis of NAS it is referred to [9].

PMC is an extension of the classical model checking (MC), which allows transitions and initial states to be weighted with probabilities. While in MC only a worst case analysis is possible, PMC can be used to generate probabilistic distributions. To use (P)MC, first an automaton-model of the system has to be built. In addition, properties to be checked are formalized through the use of some kind of formal logic. These two descriptions

are input to a model checking algorithm which checks whether the properties hold on the system. The main problem for applying PMC on NAS originates from the fact, that a probabilistic discussion relies on a sequence of states, i.e. is not able to deal with cyclic behavior. The problem can be solved, using signal-tracking. Thereby, the basic behavior of all the system components will be modeled separately and then connected as given by the system structure. Finally, a signal-tracking module is added which represents the path of the process supervised. For more information on signal-tracking and the corresponding modeling process it is referred to [7, 9].

The main idea behind this modeling process is to build all models as independent and general as possible. This guarantees an easy (re)use of the models as well as the possibility to generate the PMC-code automatically. To the latter end a description language (DesLaNAS) got defined, which includes (1) a graphical front-end to define and connect the components as well as to generate the signal-tracking. Furthermore, it contains (2) a generalized definition of a probabilistic timed automaton for both, the continuous and the discrete world. This generalization is necessary, to cover all possible behavior present in NAS. Finally (3) a transformation process from the graphical front-end towards the PMC-code is defined [7, 9, 13].

Note: By dealing with more than one cyclic component, it also has to be accounted for the drift in between their local clocks. Due to the time discretization necessary for applying a PMC analysis on NAS, this only has to be considered for systems in which the maximum clock difference (over the period of the determined response time) is larger than half of the discretization step width.

5. Modeling Aspects

The signal-tracking for the response time analysis based on the architecture shown in Figure 1 is given in Figure 2.

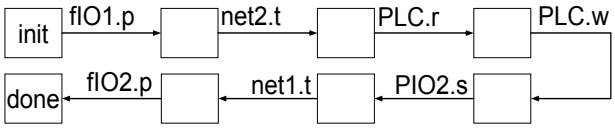

Figure 2. Signal-tracking module

The corresponding process starts with the change of the signal at the sensor (init-state). Then the process has to wait, until the field-I/O has processed this information. Thereby fIO1.p corresponds to the event p generated by the module fIO1 (field-I/O1) as shown in Figure 3. After the field-I/O has processed the information, it has to be transported by the network (net2.t – note: as the network is assumed to be duplex, there is an identical module for each of the directions, namely net1 from the PLC to the field-I/O and net2). When arriving at the PLC-I/O the information is being written directly to the input cache of the PLC, so the signal-tracking only has to wait until the PLC reads this information (PLC.r), writes the corresponding outputs (PLC.w), which then is send by the PLC-I/O (PIO2.s) over the network (net1.t). Finally the output assignments have to be processed by the field-I/O (fIO2.p).

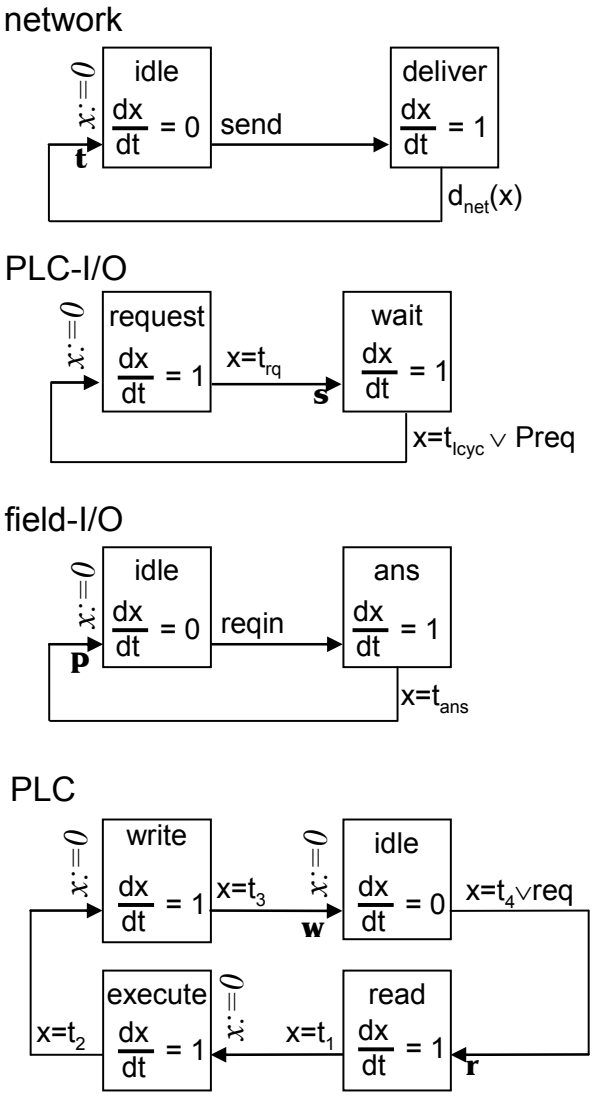

Figure 3. DesLaNAS component modules

This system requires four different module types: network, PLC-I/O, field-I/O and PLC which are shown in Figure 3. For a detailed description of the automata syntax it is referred to [7]. The network normally is in the state idle in which the local clock x is not active (dx/dt=0). If the event renamed to send is generated by the connected module (for example the fIO1.p event already discussed) the automaton changes to the deliver-state in which it stays until the stochastic delay $d_{net}(x)$ has been elapsed. By changing to the idle-state again, the event t is generated and the clock x reset to 0. The PLC-

I/O module has the same structure and only a different transition condition. Here the request-state will be entered either every $x=t_{lcyc}$ or when the request Preq is generated by the PLC. Obviously, only one of the two may happen, depending on the mode the PLC-I/O is running in (cyclic respective interrupt). In the request-state this automaton stays for the time t_{rq} necessary to send the requests. By entering the wait state, finally the event s is generated. This leads over to the field-I/O-module, which stays in the idle-state until a request requin arrives. This request either may be received from the PLC-I/O or from the sensor. The automaton then changes to the ans-state where it stays until the request is processed (given by the processing time t_{ans}).

The most complex module is the PLC which contains four states. Let's start in the read state in which the automaton stays for t_1; that is the time necessary for reading the inputs. As soon as t_1 has elapsed the automaton changes to state execute in which it stays for the parameter dependent time t_2. Afterwards all the outputs will be written (state write), which requires t_3. By changing to the idle state the event w (write) is generated. The idle state is necessary to model both, a cyclic as well as an interrupt based behaviour. In the cyclic version t_4 is set to 0 and therefore the idle state is left immediately. In the interrupt based case t_4 is set to infinity and the event req is connected to the t event of net2.

Those modules are then instantiated according to their occurrence frequency. Finally they must be connected and parameterized which allows discussing different cases on the same system structure. This generality is one of the most important strengths of DesLaNAS.

6. Case Study

Typically the response time of a NAS does not equal a constant value, but describes the stochastic transfer-function of the system. Consequently, the best way to discuss response times is to display their stochastic distribution (over the time axis). Due to the discrete time axis of the PMC-analysis, this „distribution" must be represented by a set of discrete points, each of them representing the probability that the response time can be found in the previous time interval, i.e. a value of 5% at time 20 ms means that the response time can be found in the interval (19, 20] ms with 5% probability, if the discretization step width equals 1 ms. From the right graph (open rhombi) in Figure 4 it thus can be followed, that the probability of a response time value situated in the interval (43,44] ms is 3.75%.

Figure 4 shows the response time distributions of the cases 1 c and 1 i, cf. Table 2. Those cases are using a cycle based PLC and a cycle based Client-Server-protocol for the sensor value capturing. The difference of the two cases is that for the left graph (filled squares) the result of the PLC calculations got sent immediately after finishing the PLC-cycle, while in the case investigated

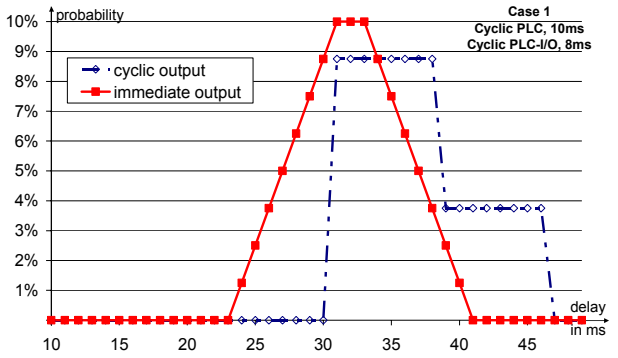

Figure 4. Response time distribution for the cases 1 c and 1 i.

for the right graph (open rhombi) the results had to wait until the PLC-I/O sent this information together with the next (cyclically recurring) sensor value request. While the length of the PLC-I/O-cycle can be extracted from both graphs (8 ms), the length of the PLC-cycle only is visible in the left graph (filled squares). For the following comparison with other cases, it is important to note, that the two graphs in Figure 4 are independent from the signal change probability *p*, as well as from the number of sensors (as long as the previously stated assumptions are fulfilled). From the distributions shown in Figure 4, characteristic values – as shown in Table 3 – can be determined. Maximum and minimum designate the largest and smallest abscissa value with a probability value larger than zero. Average and deviation are self explanatory terms. And spread is the largest minus the smallest value plus one.

Please realize that the case using a cyclic output (1 c) is better only in the spread-column. The immediate output (1 i) case however is better in all the other columns.

Table 3. Characteristic response time values (Cases 1 i and 1 c)

Case	Average (ms)	Deviation (ms)	Minimum (ms)	Maximum (ms)	Spread (ms)
1i	32	3.674	24	40	17
1c	36.9	4.323	31	46	16

However, the cyclic case has its advantages as can be verified by analyzing Figure 5, which shows the response time distribution for the case 4 c: Due to the cyclic behavior, the influence of a varying number of sensors is relatively small. Same is true for the signal's value change probability *p*. Both factors mainly influence the maximal value (cf. also Table 4).

Other than in the first case, the PLC of the fourth case does not exhibit a cyclic behavior, but starts working if requested. Consequently, the duration of the PLC's execution time is varying and the PLC will start the execution immediately after a sensor value arrived – at least as

Table 4. Characteristic response time values (Cases 4 c and 4 i)

Case	# Sensors	*p*	Average (ms)	Deviation (ms)	Minimum (ms)	Maximum (ms)	Spread (ms)
4 c	1		26.5	2.29	23	30	8
	5	0.01	26.504	2.30	23	38	16
	3	0.06	26.506	2.30	23	38	16
	5	0.06	26.533	2.34	23	38	16
4 i	1		22.5	2.29	19	26	8
	5	0.01	22.569	2.32	19	34	16
	3	0.06	22.606	2.33	19	30	12
	5	0.06	22.713	2.37	19	34	16

long as there is no competition (i.e. other sensor values to be processed). Conspicuously, case 4 c is faster than both cases 1, even under competition!

The influence of the PLC-I/O-cycle's length can be found; both in Figure 5 as well as in Table 4 (the maximum values have a difference of 8 ms, i.e. the plateau' width equals 8 ms). This is true for case 4 i (immediately output) also, as long as there is no competition (cf. Figure 6).

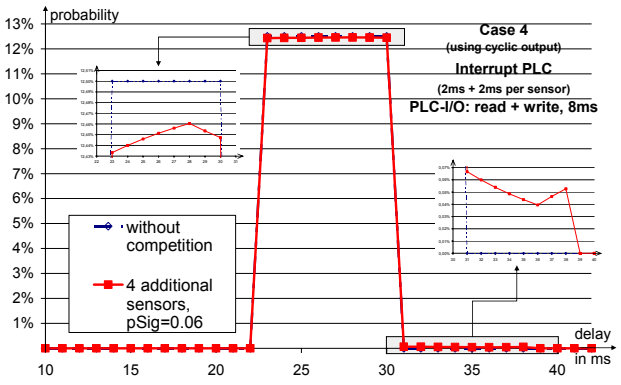

Figure 5. Response time distribution for the case 4 c.

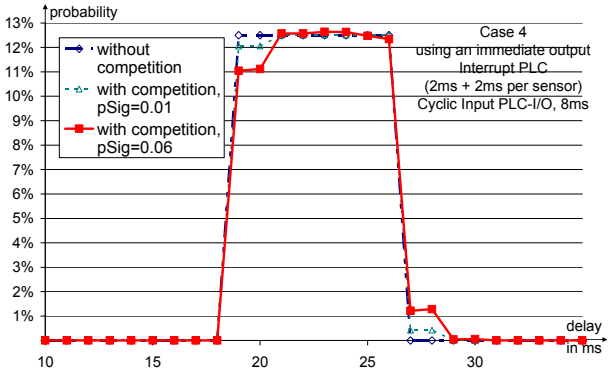

Figure 6. Response time distribution for the case 4 i

However, it is distinctly realizable that both, an increasing number of sensors, as well as an increasing value of the sensor's value change probability *p*, are delaying the information. The reason for case 4 i still outclass its cycle based correspondent results from the whole distribution being shifted 4 ms to the left. This equals the average waiting-for-the-PLC-I/O-to-sent-the-output-information-time of the 4 c case. However, deviation and spread of the two cases 4 differ only a bit: While the deviations in the case 4 c are smaller or equal than in the case 4 i, the opposite is true for the spread values.

This leads over to the remaining two cases, which use a Producer-Consumer-protocol instead of the Client-Server-protocol for the sensor value coverage. I.e. a change in the sensor's value leads directly toward a field-I/O-activity (instead of a waiting period for the arrival of the PLC-I/O's request). Figure 7 shows the response time distributions of case 2, using a cyclic PLC.

Note: As shown in [7] this leads to synchronization delays already in the competition-free case.

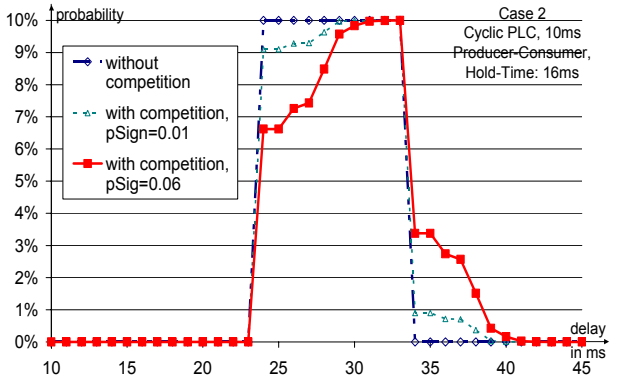

Figure 7. Response time distribution for the case 2

Now, the duration of the PLC-cycle (10 ms) can be clearly indicated. Starting from the original rectangular shape, which represents the competition-free case, increasing competition leads to more and more responses being delayed for another PLC-cycle (cf. Figure 7). This is due to waiting times in front of the field-I/O as in consequence the next possible PLC-cycle-start (read-state) might not be caught any more, i.e. the information has to wait for next reading state (which is meet every 10 ms). Moreover, Table 5 points out that the influence of additional sensors is smaller than the influence of an increased change probability *p*.

Thereby it is conspicuous that case 2 comes off worse than all other previously discussed cases in respect to the dependence of the number of sensors, the change probability *p*, and the spread value. In respect to average and deviation values it does more poorly than the cases 4 i and 4 c, but better than the cases 1 i and 1 c.

So, it remains case 3 where all parameters are turned to "as the need arises". I.e. the PLC is based on interrupt,

Table 5. Characteristic response time values, case 2

# Sensors	*p*	Average (ms)	Deviation (ms)	Minimum (ms)	Maximum (ms)	Spread (ms)
1		28.5	2.87	24	33	10
5	0.01	28.864	3.12	24	44	21
3	0.06	29.192	3.29	24	40	17
5	0.06	29.920	3.58	24	44	21

the sensor values are transmitted by the use of the Producer-Consumer-protocol and the actuator information is passed immediately from the PLC-I/O to the network. This leads – as expected – to the fastest behavior of all test cases (cf. Figure 8 and Table 6). Though, the prize for this speed is that case 3 also comes with the highest dependency on the number of sensors as well as on the change probability.

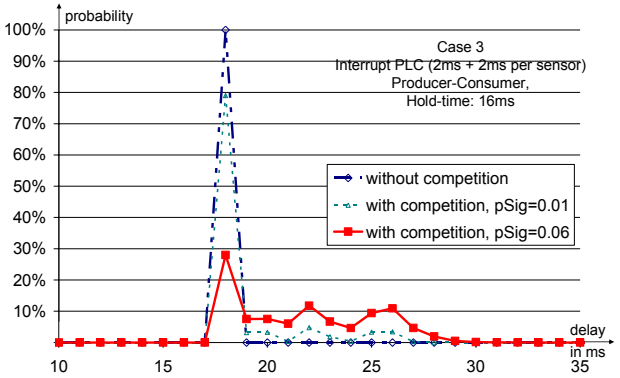

Figure 8. Response time distribution for the case 3

7. Discussion

Discussing the question which of the six basic architecture presented in Table 2 is the optimal one, requires the determination concerning the change probability on the one hand, and the frequency with which the controller processes events (respectively how often it requires new information) on the other hand. Without doubt, a signal which's value changes in average once a day can be implemented interrupt based. However for a signal which's value changes in average every 40 ms this answer is much less obvious.

In particular the influence of competition and change probability should not be neglected. This is pointed out with some details in Table 7. In the left column of the table (performance) the six cases are ordered based on their absolute average, deviation and spread values. In the right column (robustness) in contrast, they are ordered concerning their dependency of the system's load.

Table 6. Characteristic response time values, case 3

# Sensors	*p*	Average (ms)	Deviation (ms)	Minimum (ms)	Maximum (ms)	Spread (ms)
1		18	0	18	18	1
5	0.01	18.959	2.19	18	34	17
3	0.06	19.384	2.75	18	28	11
5	0.06	21.728	3.25	18	34	17

Three aspects should be mentioned:

1. In both categories case 2 (cyclic PLC, Producer-Consumer for the sensor value acquisition) comes out constantly on rank four, while the order of all other cases in the two categories is reversed, i.e. rank 1 in the left category becomes rank 5 in the right category and vise versa.
2. As case 2 thereby changes its position with the cases 4 i and 4 c, this can be formulated the other way round also: Case 4 (interrupt based PLC and Client-Server for the sensor's value acquisition) comes out always on rank 2 / 3.
3. Given the maximum load discussed in here, the maximum value of case 3 is still smaller or equal than other case's maximum value – despite the lack of robustness of this case.

Table 7. Evaluation of the six cases with respect to performance and robustness

	Performance	Robustness
++	Case 3	Case 1i/c
+	Case 4s	Case 4z
0	Case 4z	Case 4s
-	Case 2	Case 2
--	Case 1i/1c	Case 3

8. Summary and Outlook

The analysis of different NAS-architectures demonstrates that the traditional architecture (case 1 using a cyclic PLC and the Client-Server-protocol) is slow but robust (cf. Table 8). In contrast an architecture totally based on "as the need arises" (case 3) is extremely fast but also extremely dependent on the system's load. The surprise of this analysis is an unusual constellation: It uses an interrupt based PLC, although the PLC-I/O is using a Client-Server-protocol. In particular by dealing with complex systems exhibiting a varied set of influences and possibly a changing system load, this architecture does best.

Table 8. Characteristic response time values without competition (top part of the table) and relative increase comparing the case with and without competition (5 sensors, p=0.06)

# Sensors	Case	Average (ms)	Deviation (ms)	Minimum (ms)	Maximum (ms)	Spread (ms)
1	1 i	32	3.67	24	40	17
1	1 c	36.9	4.32	31	46	16
1	2	28.5	2.87	24	33	10
1	3	18	0	18	18	1
1	4 i	22.5	2.29	19	26	8
1	4 c	26.5	2.29	23	30	8
5	1	no influence				
5	2	5 %	25 %	no influence	33 %	1.1
5	3	21 %	∞		89 %	16.0
5	4 i	0.9 %	3 %		31 %	1.0
5	4 c	0.1 %	2 %		27 %	1.0

Future work will generalize the analysis by successively remove the simplifications introduced in section 3. Moreover, the test conditions will be tightened. Hereunto belongs for example failures in the system as well as shared access of several PLCs to the same field-I/O in the Client-Server-case (leading to a much better relative performance of the Producer-Consumer-case). Other than the competition of several sensors attached to the same field-I/O, which has no effect on the minimum response time, the competition between different PLCs requesting the same field-I/O may have, as demonstrated for example in [7].

Finally, it should be checked whether or not the assumption of dealing with a robust network is true also for Producer-Consumer-protocol or an immediate actuator value output by the PLC-I/O (this assumption is yet validated only for the Client-Server-protocol).

References

[1] J. Greifeneder and G. Frey: "Optimizing Quality of Control in Networked Automation Systems using Probabilistic Models". *Proc. 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Prague, Czech Republic, pp. 372–379, 2006.

[2] J. Jasperneite and P. Neumann: "*Switched Ethernet for Factory Communication*". *Proc. 8th IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*, Antibes, France, pp. 205–212, 2001.

[3] D. Miorandi and S. Vitturi: "Performance Analysis of Producer/Consumer Protocols over IEEE 802.11 Wireless Links". *Proc. IEEE Int. Workshop on Factory Communication Systems* (WFCS), Vienna, Austria, pp. 55–64, 2004.

[4] L. Liu and G. Frey: "Simulation Approach for Evaluating Response Times in Networked Automation Systems". *Proc. 12th IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*, Patras, Greece, pp. 1061–1068, 2007.

[5] F. Ridouard, J-L. Scharbarg and C. Fraboul: "Stochastic network calculus for end-to-end delays distribution evaluation on an avionics switched Ethernet". *Proc. 5th IEEE Int. Conf. Industrial Informatics (INDIN)*, Vienna, Austria, pp. 559–564, 2007.

[6] M. Kwiatkowska: "Model Checking for Probability and Time: From Theory to Practice". *Invited Paper, Proc. 18th Annual IEEE Symposium on Logic in Computer Science (LICS)*, Ottawa, Canada, IEEE Computer Society Press, pp. 351–360, 2003.

[7] J. Greifeneder and G. Frey: "DesLaNAS – a language for describing Networked Automation Systems". *Proc. 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Patras, Greece, pp. 1053–1060, 2007.

[8] B. Wittenmark, K. J. Åström, K-E. Årzén: „Computer Control: An Overview" *IFAC Professional Brief*. 2002.

[9] J. Greifeneder: "Formale Analyse des Zeitverhaltens Netzbasierter Automatisierungssysteme". *Dissertation,* Department of Electrical and Computer Engineering, University of Kaiserslautern. Published by Shaker Verlag, Aachen, 2007.

[10] J. Greifeneder and G. Frey: "Analyse des Antwortzeitverhaltens netzbasierter Automatisierungssysteme", *atp Automatisierungstechnische Praxis, Oldenbourg*, Vol. 49(10), pp. 44–54, 2007.

[11] B. Denis, S. Ruel, J-M. Faure, G. Marsal and G. Frey: "Measuring the Impact of Vertical Integration on Response Times in Ethernet Fieldbuses." *Proc. 12th IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*, Patras, Greece, pp. 352–359, 2007.

[12] J. Greifeneder, L. Liu and G. Frey: "Methods for Analyzing Response Times in Networked Automation Systems", *Proc. 17th IFAC World Congress*, Seoul, South Korea, pp. 5113–5118, 2008.

[13] J. Greifeneder and G. Frey: "Probabilistic Timed Automata for Modelling Networked Automation Systems", *Proc. 1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS)*, Cachan, France, pp. 143–148, 2007.