Shortening of the Automata Cycle of Industrial Communication System Nodes

Marcin Sidzina
Faculty of Mechanical Engineering
and Computer Science
University of Bielsko-Biała
Bielsko-Biała, Polnad
msidzina@ath.bielsko.pl

Andrzej Kwiecień, Jacek Stój
Faculty of Automatic Control, Electronics
and Computer Science
Silesian University of Technology
Gliwice, Poland
akwiecien@polsl.pl, jstoj@polsl.pl

***Abstract*—The distributed computer systems applied in industry are usually of the real-time type. In those systems the maximal system response time has to be always well known in order to ensure the functional as well as the safety parameters of the system. However, the system response time may be subject to change as a result of system development during its life cycle. As the given system grows, the response time lengthens. So it happens that in order to satisfy the real-time constraints, the response time T_{RES} has to be reduced after the system development. One of the factors that have an influence on the T_{RES} time, is the automata cycle of the communication system nodes. The shorter the automata cycle is, the more frequent is the data exchange between the CPU module and the communication module of the given system node, which improves the T_{RES} value. However, having the node's application well implemented, shortening of the automata cycle may seldom be done by the application optimization. Then, the only solution would be to temporary shorten the automata cycle by execution of some of the most crucial parts of the user application. In this paper, a method for shortening the automata cycle is described.**

Keywords-automata cycle, events avalanche, events burst, communication network, distributed real-time system, programmable logic controller, PLC, real-time, temporal determinism.

I. INTRODUCTION

Industrial computer systems are used for monitoring and control of industrial processes. The characteristic feature of those system is their operation during the actual time that the industrial process occurs. As a result, the proper operation depends not only on the right result of computation but also on the time when the result is generated [1]. Only those systems which operates in a timely manner, i.e. the real-time systems, may be used in order to meet the functional and safety demands expected in industry.

The most important temporal parameter which describes every real-time system is the maximal system response time described further as T_{RES} [2]. It may be defined as the time which elapses from an occurrence of any event in the real-time systems, to the moment of generation of the response to that event [3], [4]. The shorter the delay is, the better the system quality indexes may be. It applies to both the control systems, where the control quality is of the most importance, and the monitoring systems, where the input vectors has to be acquired periodically with jitter small enough to maintain proper reporting quality [5].

Industrial computer systems are ever changing. Their constant development is necessary in order to meet new requirements associated with new production technologies, production efficiency and to keep the system availability and maintainability on a desired level [6]. Authors experience shows, that during the system development, the temporal characteristics of the system usually change and most often degrade. To avoid loosing the system usability, the change of its temporal characteristics (e.g. the system response time) should be monitored and their value kept on the desired level accordingly to the real-time constraints defined for the given system [7], [8].

In other words, the system response time may degrade together with the system development in such a way, that the operation of the system is not possible for the functional or (what is even more important) safety reasons. Then, it may be necessary to redesign and rebuild the whole computer system in order to satisfy the temporal constraints. As the redesign would most often apply to both the hardware and the software, it is expensive and very time consuming.

If one would like to avoid the redesign and try to reduce the response time without changes in the hardware, then some changes in the software should be considered. To do so, the factors that has an influence on the response time have to be recognized.

The response time depends on the data flow in the communication network of the distributed system, so the communication bus sweep time is of the most importance [9]. Moreover, the actual response is generated by the system node (or the nodes in more complex cases), where the data processing time plays a significant role.

Real-time system needs a real-time communication network [10]. Operation of this kind of network may be described in a function of time. Three main methods for achieving the temporal determinism may be distinguished: token passing, master-slave and producer-distributor-consumer (PDC) [11], [12], [13]. All the existing known to authors real-time protocols applies some variation of one or more of the above mechanisms. For example, TokenRing is strictly token passing protocol (see: [14], [15]), Modbus is the master-slave network and FIP is PDC network (see: [16], [17], [18] and [13], [19] respectively). In Profibus network both the token passing and the master-slave mechanisms are used [20]. Meanwhile, in the CAN network, communication

is based on data exchange between the producers and the consumers of data, just like in the PDC networks, but with no distributor [21]. EtherCAT network is described as master-slave network. However, there is only one communication frame being sent in the network to be filled in with data by every network subscriber [22], [23], [24].

The data sent in the communication network is processed in the system nodes, i.e. in the communication network subscribers. Those devices are generally called PES – programmable electronic systems. In industrial distributed systems most often programmable logic controllers PLC [25] are used. The PLC is usually a time-triggered device [26], [27]. Its operation is based on automata cycle execution as shown in Figure 1 and in Figure 2.

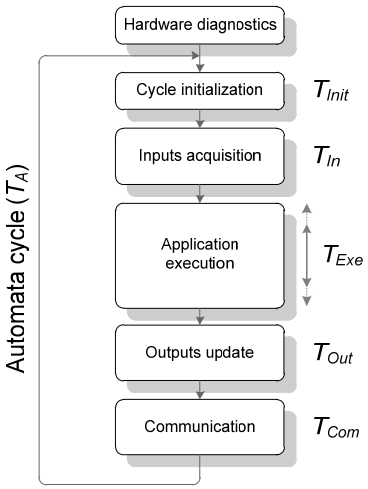

Figure 1. Main tasks executed during every PLC automata cycle [25]

At the beginning of every automata cycle (apart from cycle initialization) inputs acquisition is done. The inputs may be considered here as requests for a given PLC. Then the inputs are processed during the application execution task and the response for the inputs is calculated. After this step is finished, the outputs are copied to the output memory. The last step of the automata cycle is the communication with other nodes in the distributed system. In general, this paper is concerned with the PLC devices described by the IEC 61131-1 [25] as:

> *"digitally operating electronic system, designed for use in an industrial environment, which uses a* programmable *memory for the internal storage of user-oriented instructions for implementing specific functions such as logic, sequencing, timing, counting and arithmetic, to control, through digital or analogue inputs and outputs, various types of machines or processes. Both the PLC and its associated peripherals are designed so that they can be easily integrated into an industrial control system and easily used in all their intended functions".*

What is important to point out, is that the communication tasks are performed once per every automata cycle (in event-driven systems [28] the case may by different, but that is beyond the scope of this paper). In other words, it is a decisive factor on how frequent the communication network access is done. The frequency may be defined as follows:

$$f_{NA} = \frac{1}{T_A} = \frac{1}{T_{Init} + T_{In} + T_{Out} + T_{Com} + T_{Exe}}$$
$$f_{NA} = \frac{1}{T_{A0} + T_{Exe}} \quad (1)$$

where the elements in the denominator are the duration of the tasks executed in every automata cycle (sweep) of the PLC (Figure 1):

- T_A – Automata cycle (sweep) time,
- T_{Init} – Cycle initialization time,
- T_{In} – Inputs state acquisition time,
- T_{Out} – Outputs state update time,
- T_{Com} – Communication tasks realization time,
- T_{Exe} – Application execution time,
- T_{A0} – Duration of the system tasks.

Every sweep task but the application execution task, i.e. cycle initialization, inputs acquisition, outputs update and the communication task, are called the system tasks. The system tasks has got an execution time defined here as T_{A0} which depends on the hardware and software configuration of the node, but is constant in every automata cycle (sometimes not all inputs are configured to be acquired in every cycle, but that case is not considered here). The T_{Exe} is usually much greater than the system tasks duration. As a result, it has the greatest impact on the network access frequency f_{NA}. By changing the T_{Exe}, one may have an influence on the f_{NA} frequency and thus change the system response time T_{RES}. Of course it applies only to the cases when the response depends on inter nodes communication.

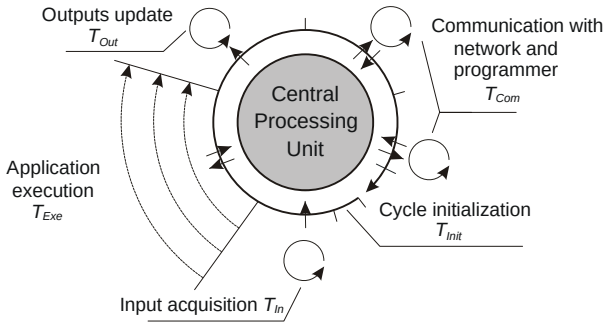

Figure 2. Communication network node operation model

The network access frequency f_{NA} defines the communication network efficiency η, its throughput P and, as a result, the response time T_{RES}. That is the reason why good nodes' application implementation techniques are of great significance. However, when the application optimization is no longer possible, then some other methods for reducing the application execution time may be helpful.

This is an entry point for shortening the automata cycle SAC which is described in some more details in the next section.

The origins of the research on shortening of the automata cycle are practical cases encountered by the authors during applications industrial computer systems for water treatment plants, heat and power plants, distributed pumping stations, etc. (for example see: [29]). It sometimes happened that the communication network did not satisfy the real-time constraints. In one of the cases, when network operation optimization was of no help, it was needed to shorten the automata cycle in order to reduce the system response time. That was the direct basis of the piece of research presented in this paper.

II. THE SAC IDEA

In this paper one of the methods for Shortening the Automata Cycle SAC is presented. It is based on the idea of segmentation of the application implemented in the system nodes onto several independent parts. The application segments can be executed according to current computer system needs. More information on the segmentation problem may be found in [30], [31]. Here, the usage of segmentation for the SAC purposes will be the point of interest.

Authors made an observation that in many systems it is possible to temporarily suspend the execution of some application segments. In the application, there are always critical segments CAS which must be executed in every automata cycle. However, most often there also exists non critical segments NCAS. Those segments are not crucial for the system operation and may be executed for example in every second sweep or their execution may be temporarily suspended. With such application segmentation, the program structure will be as shown in Figure 3.

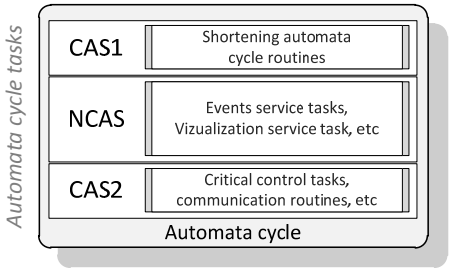

Figure 3. Application segmentation concept

In general, the segmented application consists of two CAS segments. In the first CAS1 segment there are implemented procedures for maintenance of the segmented application realization. This segment decides when and how to shorten the automata cycle by switching off the NCAS segment execution. The CAS1 segment includes, among others things, functions for communication network diagnostics and rules for starting or stopping the realization of automata cycle shortening routines (realization of the shortened cycle SC). In the second CAS2 segment all the critical functional routines of the PLC are implemented.

The duration of the automata cycle with segmentation of the application with and without execution of NCAS segments and may be defined as follows:

$$T_{Exe} = T_{CAS1} + T_{NCAS} + T_{CAS2} \tag{2a}$$

$$T_{Exe0} = T_{CAS1} + T_{CAS2} \tag{2b}$$

where:

- T_{Exe} – Duration of the full automata cycle,
- T_{Exe0} – Duration of shortened automata cycle SC (without NCAS routines executed),
- T_{CASi} – Execution time of the CAS no. i segment,
- T_{NCAS} – Duration of NCAS segment execution.

Equations (2a) and (2b) applies to the application accordingly with and without non-critical NAS application segment execution.

The shortening of the automata cycle starts when an instant data exchange with other communication system nodes is requested. It may be the result of execution of the implemented control algorithms. The CPU starts the shortened cycle SC by setting the SCF flag. From that point only CAS segments are executed in the application (the NCAS segment is temporarily suspended). After realization of the requested instant data exchange, the SCF flag is cleared, and the NCAS segments is active again. From that point communication is performed in normal fashion and the full automata cycle is executed (both CAS and NCAS segments).

In order to service the instant data exchange realization, the CAS1 segment includes queues of cyclic and acyclic network exchanges and routines associated with the SCF flag.

III. SAC METHOD DESCRIPTION

One of the method implemented for SAC purposes is RTEA5 – Reduction of Time Execution of Application, version 5. The principle of that method is shortening the automata cycle with proper and complete application execution. The previous version of the RTEA method are evolution stages leading to RTEA5 and will not be discussed here in details. Nevertheless, the general principle is common for every version of the RTEA method.

As mentioned before the RTEA methods were designed for the needs of improving the data exchange time in real-time systems with the network nodes based on the PLC devices as defined in the IEC 61131-1.

The PLC program segmentation is not indifferent for the control algorithm implemented in the PLC. Additional routines implemented for the needs of application segmentation service makes the automata cycle to lengthen and that alone degrades the response time of the system node. To minimize that temporal costs, the RTEA methods had to be kept as simple as possible and so the RTEA routines operates only on a few parameters such as:

- Automata cycle time T_A – measured value of automata cycle; use of constant value is also possible

to reduce the computational complexity, however bad T_A estimation may cause the RTEA improper operation,

- Elapsed (current) data exchange duration T_{CDEX},
- Expected data exchange duration T_{EDEX} – a constant value that should be measured during the system startup and then given in the CAS1 routines.

During the system operation the elapsed data exchange time T_{CDEX} should be measured as the difference between current CPU time and the time when the data exchange was started. The calculated value should be then compared to the expected data exchange time T_{EDEX}. When these two values are different from each other for less than one automata cycle, then the shortening of the automata cycle SC should be started. The condition is as follows:

$$\left(T_{EDEX} - T_{CDEX}\right) < T_A \Rightarrow SC \tag{3}$$

The above condition defining the moment when the SC routine should be executed is shown in Figure 4. The curved arrows in the figure shows the starting and the ending point of successive automata cycles. During the second automata cycle the data exchange is sent and the measurement of the data exchange duration is started. When the elapsed data exchange time is near the estimated data exchange duration T_{EDEX} then the shortened cycle SC starts.

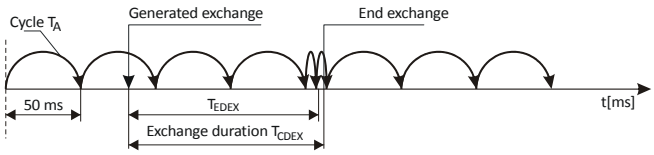

Figure 4. Full and shortened cycle in RTEA mechanism

It is clear that precise estimation of the expected data exchange duration is crucial for proper operation of the RTEA mechanism.

When the shortening of automata cycle is performed, there is always the possibility of starvation of the NCAS segment of the application. To avoid that situation, in RTEA5 method divides the NCAS segment into many smaller NCAS*i* subsegments. During realization of the shortened cycle SC not the whole NCAS segment is excluded from the execution, but only some of the NCAS*i* subsegments. As a result, all the NCAS*i* subsegments are still executed (however less often) while the SC is active and the network access may be more frequent.

The RTEA5 routines must include proper NCAS*i* subsegments execution scheduling [32], [33], [34]. It is based on a FIFO principle. Once a given subsegment is executed, it is moved to the end of the queue of the segments waiting for the execution. A sample application execution with n NCAS subsegments is shown in Figure 5.

The realization of the application with short cycle is in the RTEA5 method dynamic, not static as described in the previous section that applied to the general case of automata cycle shortening routines. It makes the application more adaptable to current circumstances and makes it more invulnerable to an event avalanche (event burst). In that case, many communication requests will affect the duration of NCAS execution time, but won't cause starvation of NCAS*i* subsegments.

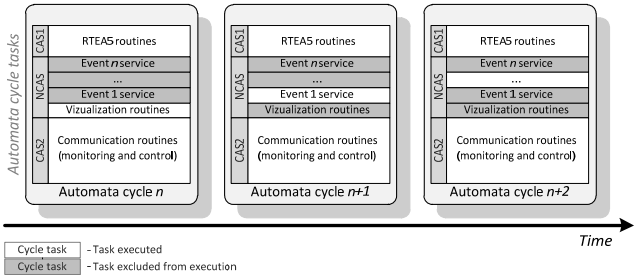

Figure 5. RTEA5 method – sample operation

The network access frequency f_{NA} in case of SAC methods is the multiplicative inverse of the sum of the total temporal costs of the program segmentation. The f_{NA} value with the realization of both CAS and one NCAS*i* subsegment per every automata cycle is as follows:

$$f_{NA} = \frac{1}{T_A} = \frac{1}{T_{CAS} + T_{NCASi} + T_{A0}} \tag{4}$$

where :

T_A – The automata cycle time,
T_{CAS} – Duration of the CAS segments execution,
T_{NCASi} – The NCASi subsegments execution time,
T_{A0} – Duration of the system tasks.

When there are n subsegments (where $n > 0$) and the service time is equal for every NCAS*i* subsegment, the network access frequency with shortened automata cycle will be:

$$f_{NAS} = \frac{1}{T_{AS}} = \frac{1}{T_{CAS} + \frac{T_{NCAS}}{n} + T_{A0}} \tag{5}$$

where T_{AS} is the duration of the shortened automata cycle.

The delay ΔT in realization of whole application with n NCAS*i* subsegments will be:

$$\begin{aligned} \Delta T &= nT_{AS} - T_A \\ \Delta T &= nT_{CAS} + T_{NCAS} + nT_{A0} - (T_{CAS} + T_{NCAS} + T_{A0}) \\ \Delta T &= (n-1)(T_{CAS} + T_{A0}) \end{aligned} \tag{6}$$

To summarize, the main goal of RTEA5 method is to reduce the duration of automata cycle with the guarantee of the execution of the whole application preventing starvation of any of the application segments.

IV. EXPERIMENTAL RESEARCH FOR TOKEN-RING NETWORK

The presented RTEA5 method of shortening the automata cycle was verified in a laboratory real-time system. The results of the research are summarized in the next section. First of all, the temporal operation of the system have to be described.

The laboratory system used Genius as a communication network (however any other communication protocol would suffice). The temporal determinism in Genius is guaranteed by the implementation of token-passing mechanism with implicit token for passing over the transmission right. Some more information on the network may be found in [35].

The system consisted of 6 nodes: three programmable logic controllers, two remote I/O stations and one Genius communication network monitor (Hand Held Monitor HHM) as shown in Figure 6.

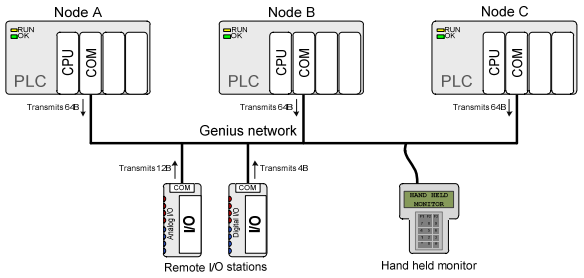

Figure 6. Diagram of the system used during the experimental research

The PLC nodes were receiving input data and sending output data to the I/O stations. Moreover, they were exchanging some data between each other using Genius Global Data exchange mechanism. The communication modules of the PLCs had coprocessors so the communication was independent of CPU module operation. The I/O stations were not programmable. The HHM was only for the network monitoring purposes.

Configuration of the communication network included definition of data exchanges for every communication module in every PLC. For every exchange the source and destination was defined (one or more destination nodes, as the data is sent in the broadcast mode). For every network node the number of transmitted and received bytes were configured.

During the research, the data exchange between two PLC nodes was monitored – the PLC A requested data from PLC B. The time of data exchange with the PLC B was measured in the PLC A as the time from sending the request to the PLC B and receiving the response. The measurement was done with the use of the system clock with 10µs precision.

The methodology of the measurement is shown on Figure 7. The measured data exchange time is the sum of the following elements:

$$T_{DEX} = T_{A(A)} + T_{Bus1} + 2 \cdot T_{A(B)} + T_{Bus2} + 2 \cdot T_{A(A)} \quad (7)$$

where :

T_{DEX} – Duration of the data exchange,
$T_{A(A)}$ – Automata cycle of the requesting node,
$T_{A(B)}$ – Automata cycle of the answering node,
$T_{Bus1,2}$ – Communication bus cycle time,

In the first automata cycle $T_{A(A)}$ specified in the (7) the data exchange request is generated in the PLC A. Then, the request is sent to the PLC B during one communication bus cycle time (T_{Bus1}). A response to that request is generated in the PLC B in two PLC sweeps $2 \cdot T_{A(B)}$ (the request could be received by the communication module COM of the PLC B just after the exchange of data between the CPU and the COM, so the request in the CPU memory will be with one automata cycle delay). The generated response is send in one communication cycle (T_{Bus2}) and received by requesting PLC A again in two automata cycles $T_{A(A)}$.

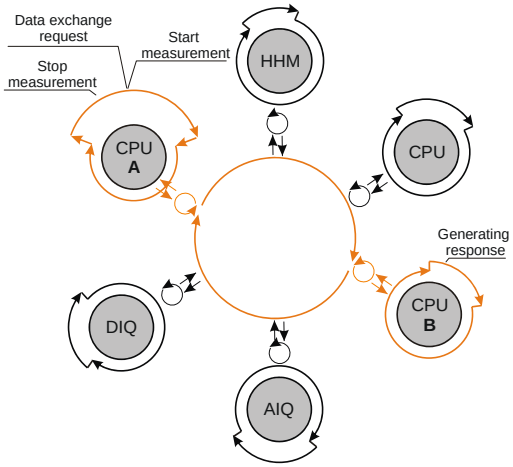

Figure 7. Data exchange measured during the experimental research

During the experimental research the following tasks were done:

- Monitoring of the automata cycle duration (measurement done in the PLC application),
- Monitoring of the Genius communication bus sweep time done by the HHM module,
- Measurement of the data exchange duration, which is the time that elapses from the moment of sending data request from node A to node B, to the moment of receiving the requested data into the CPU memory (measured is done by the PLC A),
- Measurement of the data exchange for different automata cycle duration.

The summary of the taken measurements for different automata cycle durations is shown in Figure 8. The results confirms that the longer the automata cycle is, the more time is needed for the data exchange. The one high maximum data exchange duration (for the automata cycle duration time of about 22ms) is caused by an overlapping of communication network cycle and automata cycle.

It should be noted at this point that implementation of aperiodic data exchanges may be a threat to real-time

parameters and a detailed Worst Case Response Time analyses should be realized [36].

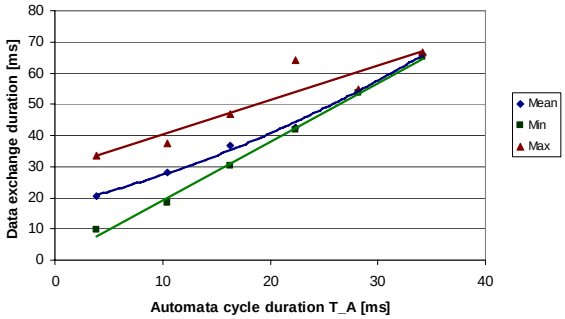

Figure 8. The summary of measurements of the data exchange duration without RTEA5 method of shortening the automata cycle.

V. DATA EXCHANGE WITH RTEA5 METHOD IMPLEMENTED

In the next step of laboratory research, the RTEA5 methods of shortening the automata cycle was implemented in the PLC A (the PLC that was requesting for the data exchange as described in the previous section). The communication requests were generated in a random manner. After triggering the data exchange, the PLC A was starting the automata cycle shortening routines in order to minimize the data exchange duration.

The summary of the experiments result is shown in Table 1. It includes the maximum duration of the data exchange with and without the RTEA5 routines activated for two different automata cycles. During standard PLC operation (without CAS routines), with 9,9ms automata cycle, the data exchange took 46,7ms. By using the RTEA5 method of shortening the automata cycle, the data exchange duration was at most 40,0ms. However, the mean value of the exchange duration would be even lower. In Figure 9a the data exchange duration for 100 data exchanges is shown. It may be noticed that the data exchanges for about half of the probes took less than 35ms and for one of them even less than 15ms (that depends on how the operation cycles of the communication network and the system nodes overlap each other). What is worth noticing is that the use of RTEA method is more efficient for longer automata cycle (Figure 9a and Figure 9b).

TABLE 1. RESULTS SUMMARY FOR RTEA5 METHOD RESEARCH

SAC method	T_A [ms]	Max (T_{DEX}) [ms]
(none)	9,9 ms	46,7 ms
(none)	15,6 ms	46,2 ms
RTEA5	9,9 ms	**40,0 ms**
RTEA5	15,6 ms	**33,1 ms**

Another thing that should be commented at this point is the fact, that in the taken measurements the maximum data exchange duration was greater for shorter automata cycle. The explanation is the cyclic and asynchronic operation of every system elements. The cycles of the system elements overlaps each other and it may happen that additional delay in one element cycle causes some total duration to be reduced as the cycles better "fits" together. What is important, during worst case analysis the overlapping of the cycles must not be taken into consideration. However, in real operation the overlapping may cause an effect like here, where shortening of one cycle (so the reduction of the temporal cost of the cycle), makes the total data exchange longer (an increase of the total temporal cost). Nevertheless, in practical application the worst case is always the condition of real-time requirements satisfaction.

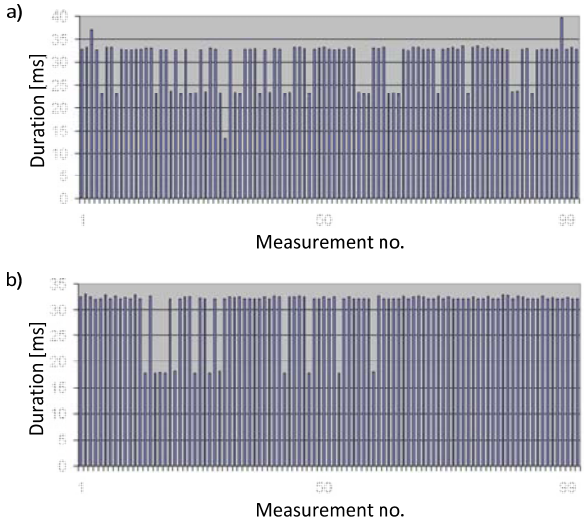

Figure 9. The data exchange duration with RTEA5 method of shortening the automata cycle, a) measurements for TA = 9,9ms, b) TA = 15,6ms

VI. CONCLUSION

The paper presents the RTEA5 as a solution for the reduction of the data exchange time in distributed computer networks which leads to increasing the responsiveness of the computer system. The presented method was tested in the laboratory experiments with a thousand of probes (measurements) for various automata cycles and various amounts of data being exchanged through the communication network. For the needs of this paper, only some most important and interesting results were selected.

The idea of RTEA method is based on the observation, that usually some of the task being realized in the system nodes are not critical and it is possible to skip their execution. That allows the shortening of the automata cycle and as a result shortening the data exchange duration which reduces the system response time. The time gained could be then used for realization of other, more critical task, e.g. associated with the service of an events avalanche. What is as much important is the fact, that the RTEA5 method dynamically adapts the automata cycle shortening routines to current circumstances, i.e. to the current workload of the communication node, the communication network, etc.

The RTEA5 method was implemented and testes on a created laboratory setup. The results showed that the

method is efficient and may be implemented in practical applications.

REFERENCES

[1] "IEEE Standard Glossary of Software Engineering Termonilogy", IEEE Std. 610.12-1990, IEEE Computer Society, September 1990

[2] L.A. Gutierrez, C.A. Franco, R. R. Jacinto, C. A Gutierrez, "Minimizing the Response Times of Aperiodic Tasks in Hard Real-time Systems with EDF", Electronics, Robotics and Automotive Mechanics Conference, pp. 268-273, September 2006

[3] "IEEE Guide for Computer-Based Control for Hydroelectric Power Plant Automation", IEEE Std. 1249-1996, IEEE Power Engineering Society, December 1996

[4] "IEEE Standard Glossary of Software Engineering Terminology", IEEE Standards Board 1990

[5] W.A. Halang, K.M. Sacha, "Real-Time Systems. Implementation of Industrial Computerised Process Automation", World Scientific Publishing 1993

[6] H. Kopetz, "Real-Time Systems Design Principles for Distributed Embedded Applications", Second Edition, Wien, Springer 2011

[7] J. Liu. W., "Real-Time Systems", Prentice Hall, New Jersey, 2000

[8] J. Real, A. Crespo, "Mode Change Protocols for Real-Time Systems: A Survey and a New Proposal", Real-Time Systems, 26(2), pp. 161-197, March 2004

[9] C. Siegfried, R. Constantin, S. Stancescu, "Evaluation of protocol for industrial informatics systems", Communications (COMM), 2010 8th International Conference on, pp. 293-296, June 2010

[10] P. Gaj, J. Jasperneite; M. Felser, "Computer Communication Within Industrial Distributed Environment-a Survey", IEEE Transactions on Industrial Informatics, Volume: 9 Issue: 1, pp. 182-189 , February 2013

[11] M. Conti, L. Donatiello, M. Furini, "Design and Analysis of RT-Ring: a protocol for supporting real-time communications", Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on, pp. 91-88, Sep 2000

[12] D. Miorandi, S. Vitturi, "Analysis of master-slave protocols for real-time-industrial communications over IEEE802.11 WLANs", Industrial Informatics, 2004. INDIN '04. 2004 2nd IEEE International Conference on, pp. 143-148, June 2004

[13] P. Raja, L. Ruiz, J. D. Decotignie, "On the necessary real-time conditions for the producer-distributor-consumer model", Factory Communication Systems, 1995. WFCS '95, Proceedings., 1995 IEEE International Workshop on, pp. 125-133, Oct 1995

[14] J. Cao, R. Steele, L. J. Yao, W. Jia, "Design and simulation of a reliable token-ring protocol for realtime communications", Computer Performance and Dependability Symposium, 1996., Proceedings of IEEE International,pp. 130-138, Sep 1996

[15] Zude Zhou, Bing Tang, Cheng Xu, "Design of Distributed Industrial Monitoring System Based on Virtual Token Ring", Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on, pp. 598-603, May 2007

[16] "Modbus Application Protocol Specification", v1.1b, Modbus-IDA, Dec 2006

[17] Song Xuehua, Lu Min, Wu Hesheng, Wang Hong, Liu Fei, "The solution of hybrid electric vehicle information system by modbus protocol", Electric Information and Control Engineering (ICEICE), 2011 International Conference on, pp 891-894, May 2011

[18] Hu Sideng, Zhao Zhengming, Zhang Yingchao, Wang Shuping, "A novel Modbus RTU-based communication system for adjustable speed drives", Vehicle Power and Propulsion Conference, 2008. VPPC '08. IEEE, pp. 1-5, September 2008,

[19] P. Pedro, A. Burns, "Worst case response time analysis of hard real-time sporadic traffic in FIP networks", Real-Time Systems, 1997. Proceedings., Ninth Euromicro Workshop on, pp. 3-10, Jun 1997

[20] E. Tovar, F. Vasques, "Guaranteeing real-time message deadlines in PROFIBUS networks", Real-Time Systems, 1998. Proceedings. 10th Euromicro Workshop on, pp. 79-86

[21] Lou Guohuan, Zhang Hao, Zhao Wei, "Research on designing method of CAN bus and Modbus protocol conversion interface", BioMedical Information Engineering, 2009. FBIE 2009. International Conference on Future, pp. 180-182, Dec 2009

[22] Tong Zhou, Jingtao Hu, "Design and realization of EtherCAT master", Electronic & Mechanical Engineering and Information Technology, 2011 International Conference on, pp. 173-177, Aug. 2011

[23] M. Rostan, J.E. Stubbs, D. Dzilno, "EtherCAT enabled advanced control architecture", Advanced Semiconductor Manufacturing Conference (ASMC), 2010 IEEE/SEMI, pp. 39-44, July 2011

[24] Lei Wang, Junyan Qi, "The Real-Time Networked Data Gathering Systems Based on EtherCAT", Environmental Science and Information Application Technology, 2009. ESIAT 2009. International Conference on, pp. 513-515, July 2009

[25] IEC 61131-1: Programmable controllers – Part 1: General information, 2003.

[26] H. Kopetz, "Why Time-Triggered Architectures will Succeed in Large Hard Real-Time Systems", 5th IEEE Workshop on Future Trends of Distributed Computing Systems, August 1995

[27] H. Kopetz, "The Time-Triggered Architecture", Proceedings of the IEEE, Vol. 91, No. 1, January 2003.

[28] Y. Itami, T. Ishigooka, T. Yokoyama, "A Distributed Computing Environment for Embedded Control Systems with Time-Triggered and Event-Triggered Processing", pp 45-54, Aug 2008

[29] A. Kwiecień, J. Rysiński, M. Sidzina, "Applicationof Distributed Systemin Control andDiagnostic Toothed Gears", Vol 39, Springer CCIS 2009

[30] A. Kwiecień, M. Sidzina, "The Method of Reducing the Cycle of Programmable Logic Controller (PLC) Vulnerable to Avalanche of Events", Vol 160, pp. 379-385, Springer CCIS 2011

[31] A. Kwiecień, M. Sidzina: "Dual Bus as a Method for Data Interchange Transaction Accelaration in Distributet Real Time Systems", Vol. 39, pp. 252-263, Springer CCIS 2009

[32] G.C. Butazzo, "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications", 2nd Edition. Springer, New York 2004

[33] C.J. Fidge, "Real-Time Scheduling Theory", The University of Queensland, Brisbane, Australia 2002

[34] M.H. Klein et al., "A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems", Kluwer Academic Publ., Boston 1993

[35] J. Stój, B. Kwiecień, "Real-time System Node Model – Some Research and Analysis", chapter 23th in monograph "Contemporary Aspects of Computer Networks Volume II", pp. 231-240, WKŁ Warszawa 2008

[36] D. A. Khan, N. Navet, B. Bavoux, J. Migge, "Aperiodic Traffic in Response Time Analyses with Adjustable Safety Level" Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on, pp. 1-9, September 2009