Performance of HTTP Protocol in Networked Control Systems

Arkadiusz Jestratjew and Andrzej Kwiecień, *Member, IEEE*

***Abstract*—Integration of Networked Control Systems is always an engineering challenge. Heterogeneous hardware and software environments combined with long lifetime of control systems makes this job particularly troublesome. Modern concepts of integration of automation systems are related to Service Oriented Architecture (SOA). While its suitability is proven in IT systems, SOA has not been adopted yet in commercial Programmable Logic Controllers (PLC), and thus cannot be considered as a solution for integration with already deployed control systems. However, during past years thousands of PLCs with embedded HTTP servers were deployed in the field. These devices, used together with modern PLC that acts as the HTTP client, enable unique opportunity of integration for control systems with soft real-time constraints. In the present study, the performance of PLC-to-PLC communications based on HTTP is evaluated and compared to Modbus TCP.**

***Index Terms*—HTTP, integration, performance, programmable logic controller (PLC), networked control system (NCS), soft real-time.**

I. Introduction

ETHERNET has become widely used in industrial networks in the last decade [1]. Several factors have contributed to this popularity, e.g., relatively low cost of mass produced components, high throughput of up to 1 Gbps and bounded worst case communication delays if switched Ethernet is used [2]. A variety of Ethernet-based communication protocols used in Networked Control Systems (NCS) exist, as most vendors of Programmable Logic Controllers (PLC) use own, proprietary protocols, designed to support specific needs. New protocols are also continuously developed [3].

Existing Ethernet-based communication solutions can be roughly grouped as follows [3].

1) Protocols defined atop of the transmission control protocol/user datagram protocol/Internet protocol (TCP/UDP/IP) stack, e.g., Modbus TCP [4].
2) Protocols that directly use unmodified Ethernet hardware, e.g., Ethernet PowerLink [5].
3) Protocols built atop of modified Ethernet hardware, e.g., EtherCAT [6].

Type (1) protocols are used for communication tasks with soft-real time deadlines of the order of hundreds of milliseconds—e.g., data exchange with Management Execution Systems (MES) or SCADA systems (vertical communications). Type (1) protocols are also used to exchange data between cooperating process stations (horizontal communications). Type (2) and (3) protocols are used for hard-real time communications with short deadlines and low jitter, e.g., drive control or closed-loop process control.

In this paper, only type (1) protocols are considered.

Horizontal communications is commonly realized using proprietary network protocols with binary data encoding, encapsulated within TCP or UDP. This is an important drawback for integration of heterogeneous control systems, as both communicating peers have to support at least one common protocol, or protocol conversion gateways must be used. Over the years, Modbus TCP became the *de-facto* standard for integration of control systems. It is supported by variety of PLC vendors, either as a standard feature or an expensive option. However, even Modbus TCP is not universally adopted.

Vertical communications often use the same proprietary protocols described above, backed by OPC or OPC UA server software running on PC computers. The software abstracts the complexity of underlying NCS and provides a standard way to access field-level data for management-level systems.

To help IT professionals access field-level data, thus lowering management costs, PLC vendors have introduced embedded Web servers (in rare cases, also FTP servers)—e.g., GE Intelligent Platforms PACSystems [7], Saia-Burgess PCD [8], Siemens S7 [9], to name just a few.

Unfortunately, no standard way to access data exists. There are shared common concepts, e.g., using specifically formatted URIs and/or Server Side Include (SSI) tags placed within hosted Web pages for reading and/or writing data to PLC memory. However, details significantly differ, as each vendor delivers own set of features using own URI format and/or SSI tags format.

A. Previous Work

The need of integration of heterogeneous NCS is well-recognized among the industry.

An adaptive approach to integration is presented in [10], by means of a gateway that translates raw field network data into XML and adapts it to specific needs of corresponding peer with translation dictionary expressed as XSLT.

Another approach can be generally described as implementing Service Oriented Architecture (SOA) in NCS environment [11]. SOA typically consists of Web services that expose functionality and clients that invoke them using standard protocols as HTTP, SOAP, and WS-*.

Manuscript received August 31, 2011; revised November 15, 2011; accepted December 05, 2011. Date of publication January 09, 2012; date of current version December 19, 2012. Paper no. TII-11-490.

The authors are with the Institute of Informatics, Silesian University of Technology, Gliwice 44-100, Poland (e-mail: ajestratjew@polsl.pl; akwiecien@polsl.pl).

Digital Object Identifier 10.1109/TII.2012.2183138

Functionality of the part of the control system can be encapsulated as the Web service hosted by the PLC. That leads to real-time SOA solutions [12], [13]. Rich metadata that comes with Web services makes integrating scenarios that involve analysis of data semantic possible [14], [15].

Above concepts have been implemented by means of server [16] and client [17]. Described prototype implementations run as extensions to the firmware (i.e., CodeSys runtime) of the specific PLC platform (IPC@CHIP from Beck IPC GmbH).

SOA approach is generally advantageous in the long term, however to the best authors' knowledge, neither exposing Web services nor consuming them is officially supported by PLC vendors in the production environment. That makes the SOA approach unfeasible when a new control system must be integrated with the already deployed system, even if the new PLC supports SOA.

B. Authors' Contributions

To address above concerns, it is proposed to leverage embedded Web servers implemented and deployed in large number of NCS over the years, for horizontal communication tasks. HTTP client software that is required for data exchanges can be implemented within application software of modern PLC if TCP/IP application programming interface (API) is available, posing no additional requirements on PLC firmware.

Having HTTP client software, vertical communication solutions can be improved especially for data acquisition and monitoring applications [18]. Cloud computing solutions have also been considered [19].

For horizontal communications, HTTP performance was compared with native, UDP-based, binary communication protocol Ether-S-Bus of Saia-Burgess PCD Series PLC [20] in homogenous environment. These experiments reveal that HTTP communications perform well enough to be feasible for soft-real time systems with deadlines of the order of several hundreds of milliseconds up to seconds, and can even perform better than native protocol for sufficiently large amount of exchanged data.

In this paper, the performance of HTTP communications is evaluated in the context of integration of PLC devices coming from independent vendors. Modbus TCP is used as a baseline for performance comparisons.

II. Protocols Comparison

HTTP and Modbus TCP protocols share the basic principles of operation, as both are Client-Server (or Master-Slave) protocols. Each data exchange begins when the Client (Master) issues a request, then the Server (Slave) processes the request and sends back the response. If PLC is involved, exchanges are synchronized with processing cycle of Client (Master), Server (Slave), or both PLCs. HTTP 1.1 protocol allows for pipelining of requests and response, however this feature is rarely (if at all) available for PLCs.

Comparing HTTP and Modbus TCP, the following important differences exist.

- Modbus TCP is binary protocol, whereas HTTP is a text based protocol.
- Modbus TCP is able to transfer up to 126 16-bit data values in a single frame. In case of HTTP, the number of individual data values that fit into a single frame may vary. It depends on the maximum size of Ethernet frame supported by both PLC devices, the size of the HTTP response header, used data encoding scheme and transferred values themselves.

For PLC devices used in the experiments, each HTTP frame can transfer from 190 (each value encoded as 5 decimal digits with sign and separator character, the frame contains the HTTP response header) up to 720 values (each value encoded as a single digit with separator character, without HTTP response header that was transferred previously).

III. Measurement Principles

Observed performance of network communication depends on schema of exchanges, that is driven by the requirements of specific production plant and its NCS. Therefore, it is crucial to evaluate performance of network communications to ensure it meets expectations for the plant. However, it makes general protocol comparisons hard, as each and every NCS poses its own set of requirements and has individual schema of network exchanges.

Schema of network exchanges (amount of data, deadlines) is derived from communication requirements of the existing, already commissioned NCS of clean water treatment (demineralization) process in one of power plants in southeast Poland. The process has been carefully chosen, in authors' work experience, as a sample of moderate-size NCS with soft real-time deadlines.

The process can be logically divided into three substations. Each substation consist of several process control devices, e.g., valves, mixers, inverter-driven pumps, etc.

- Fresh water substation (SPSS), its vector of state consists of 40 data words (16-bit).
- Demineralization substation 1 (SPMW1), using 124 data words (16-bit).
- Demineralization substation 2 (SPMW2), identical with demineralization substation 1.

Substation data must be transferred with the soft deadline of 1000 ms. It is assumed that the system is working correctly if less than 1% of individual data transfers exceed the deadline.

The real NCS is built with the single PLC that use distributed, nonintelligent remote I/O modules, based on EtherCAT communication protocol, to control each substation.

For the sake of experiment, slightly different approach is taken (Fig. 1). Control system of substations SPSS and SPWM1 is simulated by PLC1, whereas control system of substation SPMW2 is simulated by PLC2. As a result, the amount of data exchanged with PLC1 (164 data words) exceeds Modbus TCP protocol limit and two individual Modbus TCP exchanges must be used for PLC1, whereas a single Modbus TCP exchange is sufficient for PLC2. In case of HTTP, a single exchange is used either for both PLC1 and PLC2.

Processing time of substation control logic is a parameterized constant in the experiment and is expressed as *PLC sweep time*, i.e., the duration of a single PLC processing cycle.

Both programmable controllers are identical GE Intelligent Platforms (formerly GE Fanuc) PACSystems Rx7i devices with

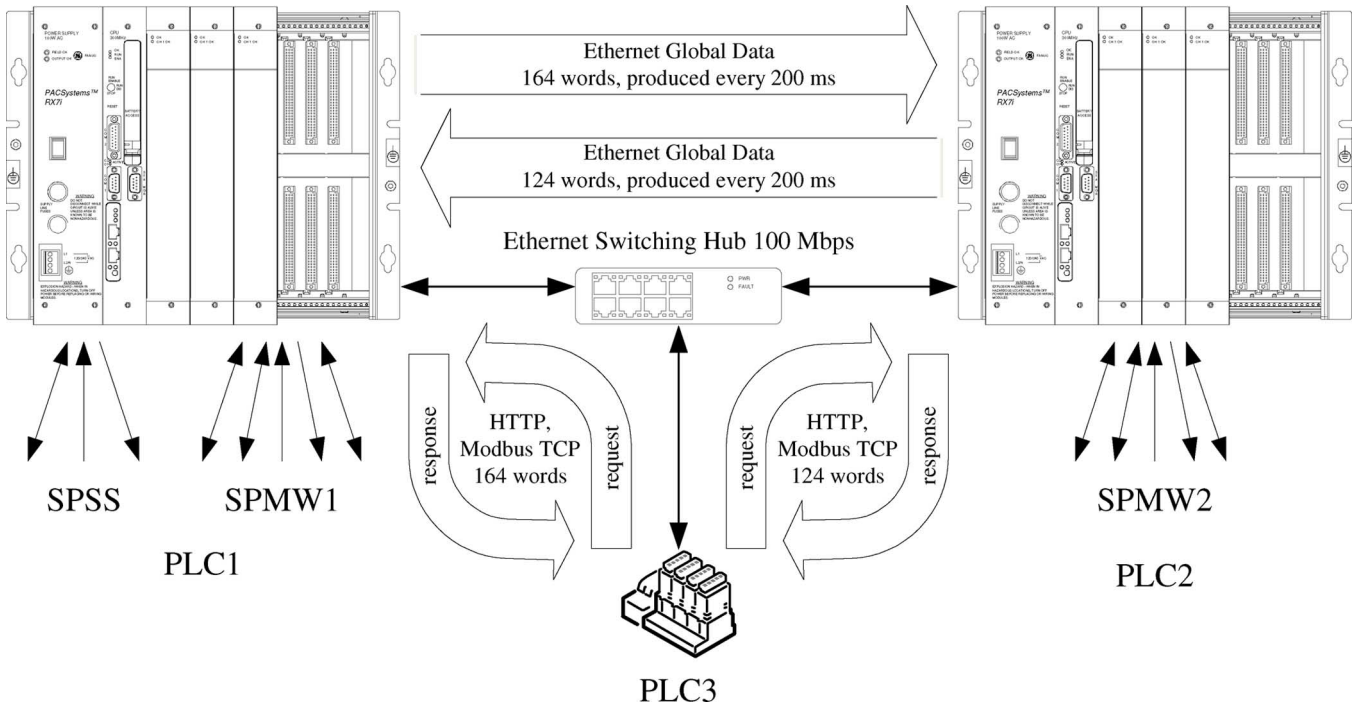

Fig. 1. Network configuration and communication parameters during the experiment.

CPE020 (700 MHz) processor modules and RMX016 memory transfer modules (not used in the experiment), running firmware version 5.50. Both controllers transmit the whole vector of state to the other one using proprietary Ethernet Global Data (EGD) protocol with the period of 200 ms. Each transmission fits in a single UDP frame.

Described NCS is extended with PLC3, that is the Saia-Burgess PCD3.M5540 device with PCD3.S100 I/O module running firmware version 1.16.45. This device is integrated with PLC1 and PLC2 by means of reading state vector using Modbus TCP and HTTP protocols. PLC3 acts as the Master (Client) station. Modbus TCP is processed with standard function blocks library available for PCD controllers with firmware support. HTTP client communications is handled entirely by the application code. Processing time of control logic of the PLC3 is a parameterized constant in the experiment. Total sweep time of PLC3 may vary as it also includes processing time of the communication protocol itself.

It is well known that the majority of data exchange time is spent on processing of TCP/IP protocol stack within the nodes, rather than Ethernet network transmission itself [21]. For described NCS, Ethernet transmission time (of the order of a fraction of a millisecond) is negligible.

IV. Results of Measurements

Maximum performance of communication in a network based on Client-Server (Master-Slave) principle is limited by the response time of the Server (Slave) station. Direct response time of PLC1 and PLC2 for HTTP requests was measured as follows. HTTP requests for reading of 164 data words were issued by a simple .NET application running on the PC computer (Intel Core2 6600 2.4 GHz, 2 GB RAM, Windows XP SP3). Next request is sent immediately after receiving of the response to previous request. Wireshark (version 1.0.4, SVN Rev 26501, using WinPcap version 4.0.2) was used to monitor the exchange of data.

The response time was calculated as the time that passed from transmission of the HTTP request to reception of the last packet of HTTP response (the response is sent by this particular PLC type as two packets, the first one contains "HTTP/1.0" and is sent immediately after reception of the request, the second one contains the rest of the response and is sent later, when data becomes available). Timestamp for the Ethernet frame is assigned by the WinPcap driver using `KeQueryPerformanceCounter()` system function with precision of the order of microseconds [22], while measured response times are of the order of tens or hundreds of milliseconds. Therefore, timestamp error due to clock precision is negligible for response time measurement.

Two series of measurements were executed against both PLC1 and PLC2. Each series consists of 1000 individual requests for total 4000 measurements. These measurements were repeated for sweep times of 10, 25, 50, and 100 ms.

The results of measurements are shown on the Fig. 2. Maximum observed response time was 1706 ms, and a total of 8 of 16 000 measurements exceeded 250 ms. These large response times are caused by implementation details of PLC1 and PLC2 embedded HTTP servers, and may vary if other PLC types are used.

At the first stage, data exchanges with a single remote node were measured. Two independent series of measurements, that consists of 1000 exchanges each, were executed with varying parameter values: PLC1 and PLC2 sweep time and processing time of PLC3 control logic, for total of 64 series. Only selected measurement series that represent worst-case and best-case conditions are presented in figures to conserve space.

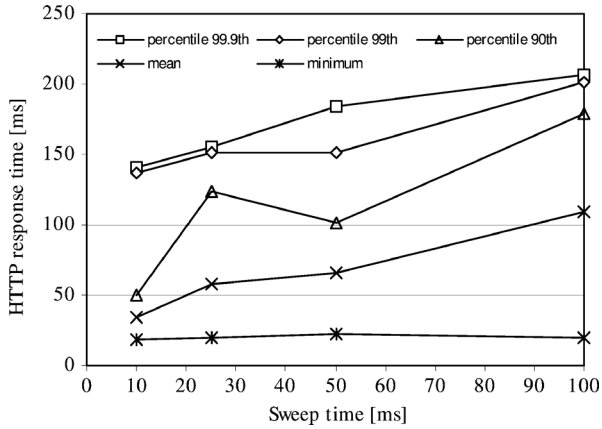

Fig. 2. Response times of built-in HTTP server of the PLC1 and the PLC2.

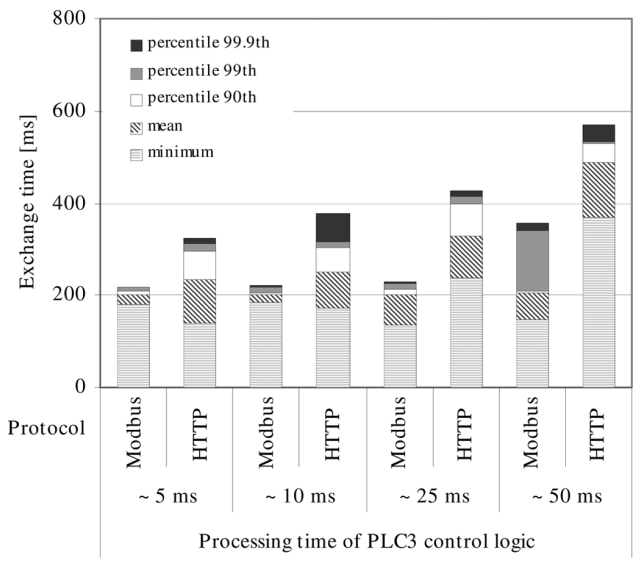

Fig. 3. Total exchange times of 164 data words with the PLC1 observed by the PLC3. Sweep time of PLC1 equals 100 ms. Note that Modbus TCP protocol supports up to 126 word in a single data exchange, thus a sequence of two exchanges of 40 and 124 words were executed.

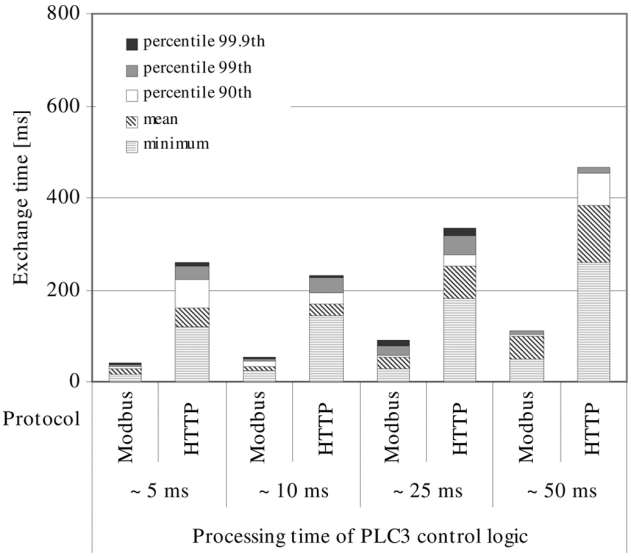

Fig. 4. Total exchange times of 124 data words with the PLC2 observed by the PLC3. Sweep times of PLC2 equals 10 ms.

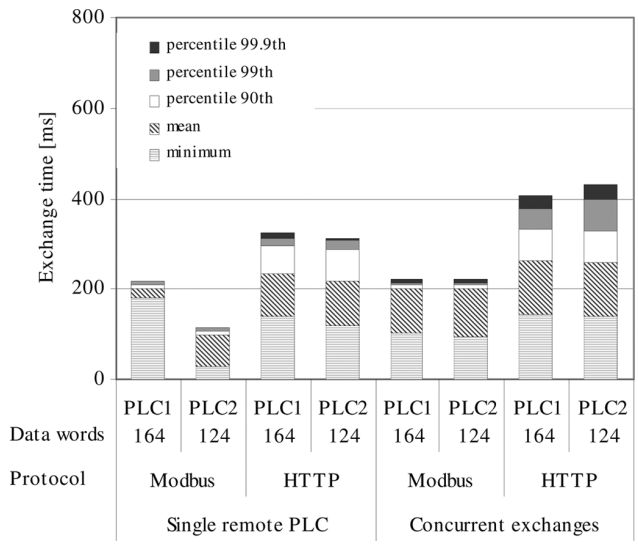

Fig. 5. Total exchange times observed by the PLC3. Sweep times of PLC1 and PLC2 equals 100 ms, processing time of PLC3 control logic ≈ 5 ms.

The results of measurements for SPSS and SPMW1 substations running on the PLC1 with sweep time of 100 ms are shown in Fig. 3. To transfer 164 words of data, two Modbus TCP exchanges must be used, while individual HTTP exchange is able to carry all the data. Combined with a long sweep time of the remote PLC, this makes the worst conditions for the Modbus TCP protocol. In such a case, shown on two leftmost bars on Fig. 3, the minimum observed response time was even better for HTTP than Modbus TCP, whereas mean response times were comparable (235 ms for HTTP versus 200 ms for Modbus TCP).

Comparing the raw HTTP server performance (Fig. 2) with total HTTP exchange times (Fig. 3) for percentile 90th and above, it is obvious that the HTTP server limits performance in this case, not the protocol itself or its client implementation.

The opposite situation is shown in Fig. 4, where exchange times of 124 data words are revealed. Short sweep time of the remote PLC2 (10 ms), lower amount of data that fits into a single Modbus TCP packet, and fast implementation in the firmware all contribute to the significant advantage of Modbus TCP over HTTP in this case.

In the second stage, data exchanges with both PLC1 and PLC2 were executed concurrently. The results of measurements are shown on Figs. 5 and 6 and compared with results obtained in the first stage of the experiment. As expected, the performance of HTTP exchange processing implemented in application code strongly depends on the sweep time of PLC3, whereas performance of firmware implementation of Modbus TCP remains relatively stable.

Mean processing time of the communication code relative to the control logic of PLC3 is shown in Fig. 7. Processing time required by communication code for Modbus TCP communication is relatively low and stable for all conditions, as almost all

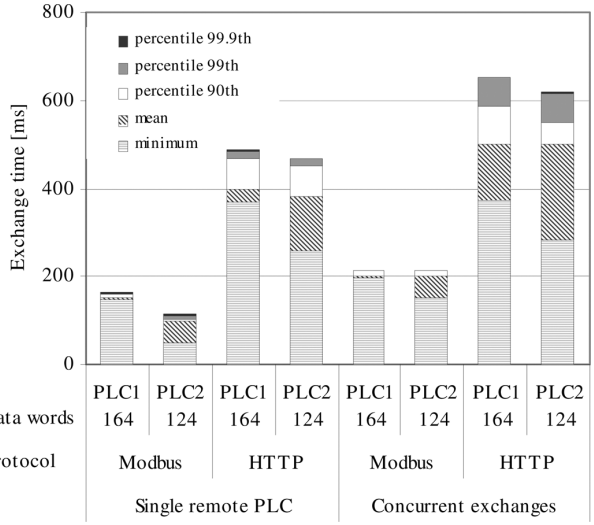

Fig. 6. Total exchange times observed by the PLC3. Sweep times of PLC1 and PLC2 equals 10 ms, processing time of PLC3 control logic ≈ 50 ms.

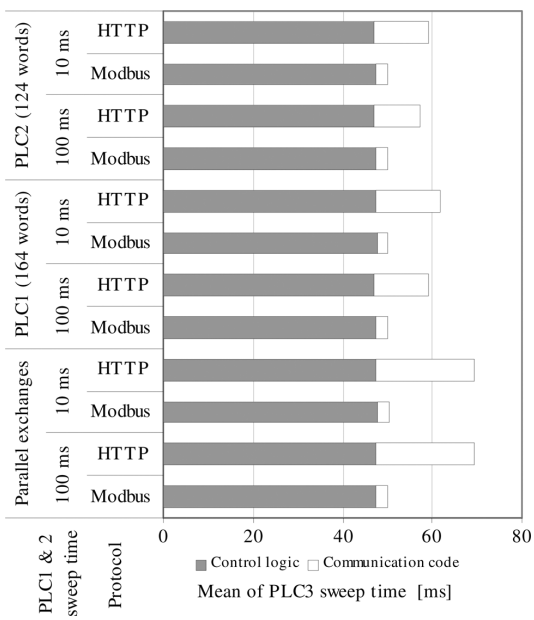

Fig. 7. Sweep time of the PLC3 with active data exchanges, processing time of control logic ≈ 50 ms.

work is done at the firmware level and data is transferred in the binary form.

Processing time of HTTP protocol exchanges in the application code is significantly longer and depends on the amount of transferred data as all transferred values must be converted from string representation before they can be used.

V. Conclusion

Measured performance of the Modbus TCP protocol is significantly better than HTTP. While several factors contribute to this result, the most important is relatively low performance of PLC application code executing complex string processing required by the HTTP protocol, compared to high performance of firmware code implementing Modbus TCP.

In best conditions, HTTP can achieve performance similar to Modbus TCP. These conditions are: short cycle time of Client (Slave) PLC, long cycle time of Server (Master) PLC and transfer of such an amount of data (including HTTP response header) that fits into a single Ethernet frame, but requires two or more Modbus TCP exchanges.

Although HTTP performs worse than Modbus TCP, it performs well enough to meet specified soft real-time constraints of the sample NCS, as 99.9% of measured HTTP data exchanges are completed in less than 700 ms. That makes the HTTP communications an alternative that is worth evaluating for soft real-time NCS.

It must be noted that the presented results are based on the specific individual case of NCS and cannot be moved automatically to other systems. Eventual decision on application of HTTP protocol requires feasibility analysis for the specific case, based on performance measurements of automation components used, just like all applications of industrial networks.

References

[1] R. A. Gupta and M.-Y. Chow, "Networked control system: Overview and research trends," *IEEE Trans. Ind. Electron.*, vol. 57, no. 7, pp. 2527–2535, Jul. 2010.

[2] K. C. Lee, S. Lee, and M. H. Lee, "Worst case communication delay of real-time industrial switched Ethernet with multiple levels," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1669–1676, Oct. 2006.

[3] J. Jasperneite, J. Imtiaz, M. Schumacher, and K. Weber, "A proposal for a generic real-time Ethernet system," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 75–85, May 2009.

[4] *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*, Modbus-IDA, 2006.

[5] *Ethernet POWERLINK Communication Profile Specification Version 1.1.0* , Draft Standard 301, Ethernet PowerLink Standarization Group, 2008.

[6] *Industrial communication networks—Fieldbus specifications*, IEC 61158-3/4/5/6-12 (Ed.2.0)—Part 3-12: *Data-link layer service definition*—Part 4-12: *Data-link layer protocol specification*—Part 5-12: *Application layer service definition—Part 6–12: Application layer protocol specification—Type 12 elements (EtherCAT)*, 2010.

[7] *TCP/IP Ethernet Communications for PACSystems User's Manual.* GE Intelligent Platforms, Jan. 2010.

[8] *Ethernet Manual for the PCD Series*. Murten, Switzerland: Saia-Burgess Controls AG, Feb. 2010.

[9] *S7-1200 Programmable Controller System Manual*. Nürnberg, Germany: Siemens AG, Jul. 2011.

[10] S. Eberle, "Adaptive Internet integration of field bus systems," *IEEE Trans. Ind. Informat.*, vol. 3, no. 1, pp. 12–20, Feb. 2007.

[11] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *IEEE Trans. Ind. Informat.*, vol. 1, no. 1, pp. 62–70, Feb. 2005.

[12] N. Komoda, "Service oriented architecture (SOA) in industrial systems," in *Proc. IEEE Int. Conf. Ind. Informat.*, Aug. 2006, pp. 1–5.

[13] T. Cucinotta, A. Mancina, G. F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusiná, "A real-time service-oriented architecture for industrial automation," *IEEE Trans. Ind. Informat.*, vol. 5, no. 3, pp. 267–277, Aug. 2009.

[14] J. L. M. Lastra and M. Delamer, "Semantic web services in factory automation: Fundamental insights and research roadmap," *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 1–11, Feb. 2006.

[15] A. P. Kalogeras, J. V. Gialelis, C. E. Alexakos, M. J. Georgoudakis, and S. A. Koubias, "Vertical integration of enterprise industrial systems utilizing web services," *IEEE Trans. Ind. Informat.*, vol. 2, no. 2, pp. 120–128, May 2006.

[16] M. Mathes, C. Stoidner, S. Heinzl, and B. Freisleben, "SOAP4PLC: Web services for programmable logic controllers," in *Proc. 17th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process.*, Weimar, Germany, 2009, pp. 210–219.

[17] C. Stoidner and B. Freisleben, "Invoking web services from programmable logic controllers," in *Proc. 2010 IEEE Conf. Emerging Technol. Factory Autom. (ETFA)*, 2010.

[18] A. Jestratjew, "Improving availability of industrial monitoring systems through direct database access," in *Proc. 16th Conf. Comput. Networks, CCIS'09*, A. Kwiecień, P. Gaj, and P. Stera, Eds., 2009, vol. 39, pp. 344–351, Springer.
[19] A. Jestratjew and A. Kwiecień, "Using cloud storage in production monitoring systems," in *Proc 17th Conf. Comput. Networks, CCIS'10*, A. Kwiecień, P. Gaj, and P. Stera, Eds., 2010, vol. 79, pp. 226–235.
[20] A. Jestratjew and A. Kwiecień, "Using HTTP as field network transfer protocol," in *Proc 18th Conf. Comput. Networks, CCIS'11*, A. Kwiecień, P. Gaj, and P. Stera, Eds., 2011, vol. 160, pp. 306–313.
[21] T. Skeie, S. Johannessen, and Ø. Holmeide, "Timeliness of real-time IP communication in switched industrial Ethernet networks," *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 25–39, Feb. 2006.
[22] L. Degioanni, M. Baldi, F. Risso, and G. Varenni, "Profiling and optimization of software-based network-analysis applications," in *Proc. 15th IEEE Symp. Comput. Architecture and High Performance Comput. (SBAC-PAD)*, 2003, pp. 226–234.

Arkadiusz Jestratjew received the M.Sc. degree (Hon.) and Ph.D. degree in informatics from the Silesian University of Technology, Gliwice, Poland, in 2004 and 2011, respectively.

He is currently with the Institute of Informatics at the Silesian University of Technology. He authored and coauthored over 25 journal and conference papers. His research interests include distributed industrial control systems and networks.

Dr. Jestratjew serves in the organizing committee of the Computer Networks Conference.

Andrzej Kwiecień (M'08) received the M.Sc. degree in electrical engineering in 1978, and the Ph.D. and D.Sc. degrees in informatics from the Silesian University of Technology, Gliwice, Poland in 1985 and 2003, respectively.

He has been a Professor at the Silesian University of Technology since 2006, and has been an Associate Professor at the Silesian High School of Informatics, Katowice, since 2000. His research interests include design of distributed real-time systems, industrial process visualization and industrial networks.

Prof. Kwiecień is a member of the Polish Information Processing Society, the Chief of the program committee of the Computer Networks Conference and a member of program committees of various conferences.