Soft Real-Time Communication with WebSocket and WebRTC Protocols Performance Analysis for Web-based Control Loops

Tomasz Karla
Department of Electrical Engineering, Control Systems and Informatics
Faculty of Electrical and Control Engineering,
Gdansk, Poland
tomasz.karla@pg.edu.pl

Jaroslaw Tarnawski
Department of Electrical Engineering, Control Systems and Informatics
Faculty of Electrical and Control Engineering,
Gdansk, Poland
jaroslaw.tarnawski@pg.edu.pl

***Abstract*—The web browser has become an access window for content and services. The browser is available on almost any device connected to the network, regardless of its intended use: desktop, mobile device, computing server, e-book reader etc. Browsers are used by people to read news, contact the world, to check a bank account, register a visit at the doctor, watching video content, electronic purchases, using web versions of the office package, etc. Universal availability, uniformity and intuitive handling, independence from hardware and operating systems are advantages over executable programs. Current applications in the field of automation include, for example, the implementation of user interfaces to present the state of the process and to accept operators' instructions in the form of web pages. The article considers network services and web browser as elements of the control loop. At least soft real-time (RT) work is needed for such applications. With the development of network protocols, the issue of providing RT transmission has appeared. There are currently two leading protocols taking into account the time aspects: WebSocket and WebRTC. The article compares their properties (mainly communication delays) in various network configurations and assess their suitability as an infrastructure of control systems. Selected techniques allowing to handle with delays in control loops are presented. Two control loops (simple PID loop and a multidimensional DMC) on web platform case studies are presented.**

Keywords— Real-Time Communication, Networked Control Systems, WebSocket and WebRTC Protocols, DMC

I. INTRODUCTION

Along with the development of communication techniques, a new field in control science was created, called Networked Control Systems. The main problems encountered in controlling via the network are unknown, variable in time delays, possible loss of sent packets or their coming in the wrong order. In [12] two main approaches to control in such systems are considered: "control over network" where the controller takes into account network specifications and "control of network" where the network is treated as active resource and can be controlled. In [13] three main approaches are presented: dynamic delay time compensators, multiple control loops via network, variable sampling rate to reduce network traffic, lossrate, delay variability and better represent model of the process.

The issues of considering communication delays in the control are usually solved by using delay time compensators, i.e. using a well-known Smith predictor or based on the theory of robust control.

Considerations regarding control in network systems prevail over UDP and TCP protocols, i.e. they often end in the fourth layer of the ISO / OSI model, i.e. in the transport layer. In this article, the main accents are located on the comparison and determination of the usefulness of protocols for web browsers, i.e. operating at the application layer of the OSI communication model.

The ability to transmit data in real time via the local network and via the Internet using web browsers on terminal connection devices opens up many possibilities related to simulations and control, both in industrial and research as well as educational applications [3]. Using existing hardware structures, it is relatively easy to expand their functionality by the possibility of attaching them to simulation loops or control loops with real objects / controllers. The ease of GUI implementations in the browser and the possibilities of visualization make educational purpose one of the most effective techniques for transferring knowledge and familiarizing users with the dynamics of real objects through interactive simulations. Such work was undertaken by the authors in order to provide a real-time simulator of basic nuclear reactor processes in the form of a website [4, 5, 6].

However, there is one major disadvantage of such solutions, namely communication delays that can significantly affect the quality of simulation / control loops. However, there are techniques that allow to compensate for the resulting delays to a large extent. The dynamics of objects impose certain requirements on the performance of technology in the case of real-time simulation. In this article, in order to determine the usability of these technologies in the simulation / control loops, attention has been paid to the delays of various communication techniques that directly affect the quality of the received results.

II. COMMUNICATION TECHNOLOGIES

There are many technologies related to data transmission using local networks and the Internet. The most popular and universal technologies are using IP and TCP or UDP protocols, which correspond to the network and transport layers in the ISO OSI model. Over the years, a number of other protocols of the higher layers of this model have been developed that are specialized in the transmission of various types of data. One of the most commonly used is the HTTP protocol for handling websites from the level of web browsers. Along with the development of the Internet and multimedia, there has been a need to transmit data in real time, initially

mainly due to audio-video transmissions and to support the so-called dynamic websites. The basic HTTP protocol mainly uses the TCP / IP protocol, operates in the client server technology, where the server only responds to client requests. This type of protocol is suited to two-way data exchange without delays. The TCP protocol is characterized by high reliability in data transmission, however, because of extensive data control and confirmation mechanisms, it generates significant delays. The UDP protocol, which does not have these mechanisms, transmits data with the least possible delays, however, the lack of any control over data consistency makes its use mainly reduced to data exchange in which occasional loss of data is acceptable, e.g. in the case of typical audio-video transmissions, where more important is the transmission of data in real time and minor distortions are acceptable.

Among the modern popular technologies to exchange data over the Internet using web browsers, there are two that meet the criteria of communication in real time while maintaining the basic control of data consistency. The first of these is WebSocket [1], a standard that is supported by virtually all major web browsers. The second popular technology for real-time data transfer is WebRTC (Web Real-Time Communication) [2] which is a free, open-source project. Its main purpose is to provide real-time communication between various browsers, mobile devices or IoT devices (Internet of Things) using a simple API. Right now, all major web browsers support WebRTC, either as default feature or as optional one activated through options and many others declare support for it in the future.

III. WEBSOCKET

WebSocket technology uses so-called "sockets" for client-server connections. It enables two-way communication between the server and the client where communication does not have to be initiated by the client. Like the HTTP protocol, it also uses the TCP protocol, however, due to the combination of only one connection and the appropriate optimizations, the transmission delays have been significantly reduced. WebSocket can operate using the same ports as the HTTP protocol, which makes it much easier to set up connections in the firewall settings typical for Internet browsing. The client connects to the server by means of an appropriate HTTP request, then the WebSocket protocol is updated and a single, permanent connection to data exchange is established, where both the server and the client can simultaneously send data / messages. The communication scheme is shown in Fig.1.

Messages exchanged between clients and server are defined by developers of applications and each of them usually trigger associated functions/actions on server or client side. Each message can have data associated with it, which can be processed by specific response functions. This kind of communication and processing is very easy to implement and develop by developers of applications.

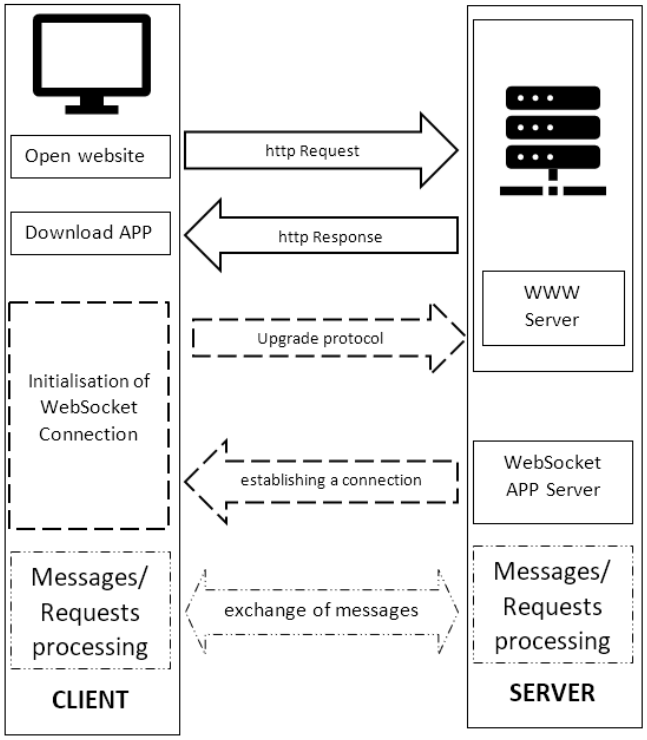

Fig. 1. The WebSocket communication scheme.

IV. WEBRTC

The technology has been developed mainly for real-time audio-video transmissions however, it also allows the transfer of other data. Depending on the type of data exchanged, various components are prepared for their use, such as the VP8 codec for video transfers or the iLBC codec for voice transfers. The communication itself takes place using the P2P network (Peer to Peer) where the connection is established directly between the clients. For this reason, for the simplicity of establishing connections between clients, an additional signaling server using the WebSocket technology as well as STUN servers (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT) are normally used to provide a connection when clients are in different networks. The direct connection between clients affects the data transfer performance. WebRTC works mainly on the basis of UDP protocol, however it allows for additional data control and use encryption of connections. In a typical scenario of using WebRTC technology, clients connect to a WWW server using web browsers from which they load the application to connect with each other. The applications connect to the signaling server to initiate a P2P WebRTC connection. Depending on the network configuration, it may be necessary to use STUN / TURN servers for communication. After establishing a permanent connection, clients can send messages to each other. Figure 2 shows the connection diagram.

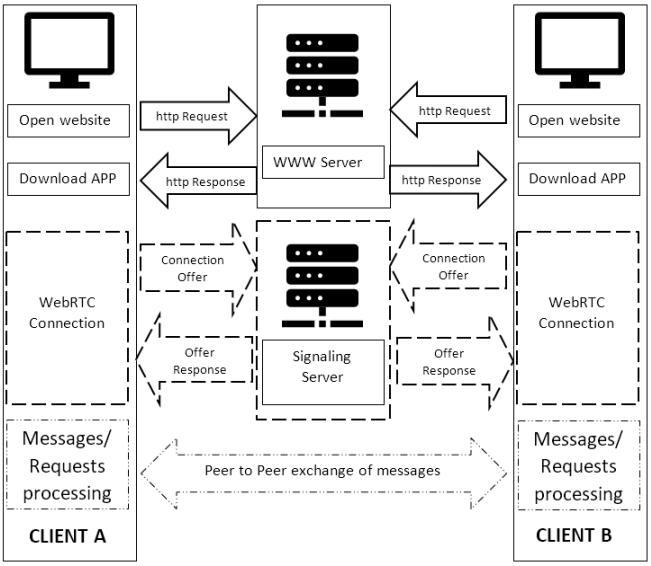

Fig. 2. The WebRTC communication scheme.

V. COMMUNICATION TESTS

Both discussed technologies can be effectively used for real-time data transmission. Depending on the network configuration, one or the other technology can potentially allow more efficient communication between devices, providing less latency and proper data control. Authors decided to examine the performance of both technologies in several different network configurations using a simple test scenario, where one unit is the "controller" and the other is the "object simulator". The "controller" at certain times sends information to be processed by the "object simulator" and then receives a response from it. In the simplest variant of the tests carried out, the "controller" transmits the time value, which by means of selected mathematical operations (sine operation) is processed in the "object simulator" and returned to the "controller". Both units, "controller" and "object simulator" are implemented using JavaScript or node.JS technology.

The tests of both technologies were carried out with the following assumptions:

1) WebSocket tests: "Object simulator" has been implemented on one computer as a simulation server using node.JS technology and io.socket libraries (supporting WebSocket) next to a web server containing the controller application.
The second computer connected to the WWW server in order to download the controller application, which then as a client connected to the simulation server.

2) WebRTC tests: a web server has been implemented on one computer containing the object simulator and controller applications. The signaling server has also been implemented on the same machine.
The second and third computers downloaded the applications of the controller and object simulator from the WWW server, then they both connected each other using a signaling server.

Communication in various scenarios has been analyzed mainly in terms of the correctness of data transmission and received transmission delays. Each scenario assumed the analysis of data obtained from continuous transmission between units within 1 minute of system operation. The controller sent next data to the object simulator every 5 ms if it received a response to the previous data set within 100ms. When the time was exceeded, another set of data was sent and the previous packet was considered lost. Tables 1, 2 and figure 3 show selected variants of network configuration and communication between devices. LAN network means that the devices were connected to the Internet network via a laboratory network, where all devices were located. WAN means connecting to the Internet network outside the laboratory network. Figure 3 presents schemes of tested scenarios. The table 3 contains the results of the carried out tests.

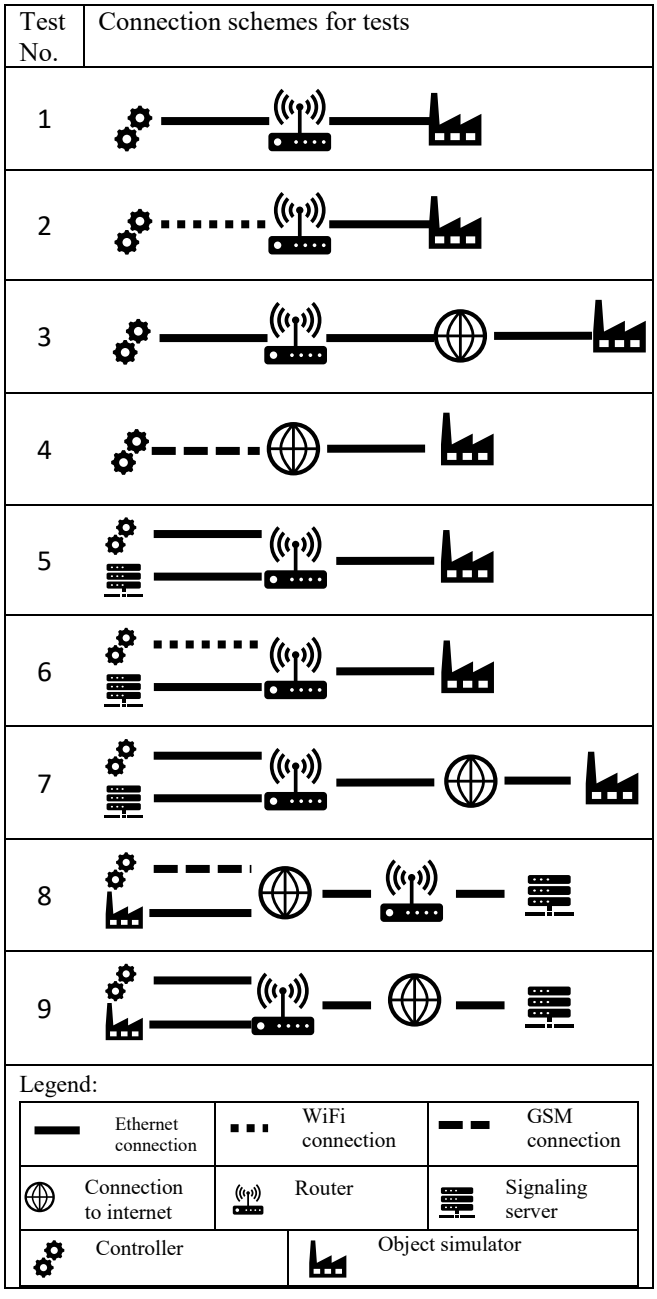

Fig. 3. Connection schemes for tests.

TABLE I. WEBSOCKET SCENARIOS

WebSocket		
Test No.	Controller	Object simulator
	Network type -> Connection to internet	
1	LAN -> Ethernet	LAN -> Ethernet
2	LAN -> WiFi	LAN -> Ethernet
3	LAN -> Ethernet	WAN -> Ethernet
4	WAN -> Modem GSM	WAN -> Ethernet

TABLE II. WEBRTC SCENARIOS

WebRTC			
Test No.	Controller	Object simulator	Signaling server
	Network type -> Connection to internet		
5	LAN -> Ethernet	LAN -> Ethernet	LAN -> Ethernet
6	LAN -> WiFi	LAN -> Ethernet	LAN -> Ethernet
7	LAN -> Ethernet	WAN -> Ethernet	LAN -> Ethernet
8	WAN -> Modem GSM	WAN -> Ethernet	LAN -> Ethernet
9	LAN -> Ethernet	LAN -> Ethernet	WAN -> Ethernet

TABLE III. RESULTS OF TESTS

	Test No.	Max latency [ms]	Average Latency [ms]	Packet loss [%]
WebSocket	1	3	2.5	0
	2	5	3.7	0
	3	33	15.9	0
	4	61	22.5	0
WebRTC	5	3	2.3	0
	6	5	2.9	0
	7	35	13.7	0
	8	41	19.8	0
	9	30	2.3	0

The results of the tests showed that in both technologies the data control mechanisms work correctly, no lost packet was observed. The communication delays themselves are similar to each other in similar scenarios between the two technologies, however in most cases WebRTC obtains slightly lower delays. An unquestionable advantage of WebRTC technology is the ability to host a set of applications on an external server that can be made available to multiple devices on another network. Thanks to this, devices can load selected programs from a centralized remote server and the simulation itself can be performed locally between clients. This situation can be seen in scenario No. 9, where, despite communication with the WWW server via the Internet, the delays were almost identical to when all devices were in the local network.

VI. WEB BROWSER AND WEB COMMUNICATION IN CONTROL LOOPS

There are various possibilities of using an internet browser and network communication in automation applications. The authors conducted work on the construction of real-time simulators of high time resolution where the browser acted as simulator of basic phenomena occurring in a nuclear reactor [5]. Since it is possible to execute code that performs such computationally demanding tasks and RT communication protocols are available, it is possible to construct entire control loops by placing object model and controllers on different computers. This is very interesting in the issues of multidimensional and decentralized control with many independent control loops.

VII. DELAYS HANDLING

The control theory provides several ways to account for delays in control loops. The simplest solution is to use the Smith predictor [9, 10, 11], which causes the controller to receive feedback turned up in such a way that the delay in the system was not present. The idea of Smith predictor is presented in fig. 4.

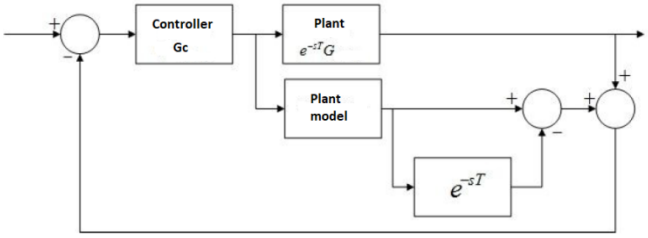

Fig. 4. Smith predictor scheme

However, this approach requires an accurate object model and latency knowledge. The latter is often estimated with round trip time (RTT). This approach is not practical and effective in a situation where the delay is variable over time. Another method is to replace a model with unknown delay value $d(k)$ in (1) with a discrete model with unknown parameters $a_i(k)$ (2). With this version of the model methods of adaptive control based on parameters identification can be used.

$$y(k) = a(k)u\big(k - d(k)\big) + e_1(k) \tag{1}$$

$$y(k) = \sum_{i=\underline{d}}^{\overline{d}} a_i(k)u\big(k-i\big) + e(k) = a_{\underline{d}}(k)u\big(k-\underline{d}\big) + \ldots + a_{\overline{d}}(k)u\big(k-\overline{d}\big) + e(k) \tag{2}$$

The most effective method of including delays, including these variables over time, is predictive control e.g MAC, GPC, DMC [8]. With proper selection of prediction horizons, the influence of delays can be limited.

VIII. PID CONTROL EXAMPLE

One test scenario consisted of control loop with simple first order inertia with small delay as simulated object and simple PID controller. The delayed nature of object together with communication delays posed problem for simple PID controller. Fig. 5 presents results of simple simulation showing reference trajectory and obtained results. Fig. 6 presents observed delays of communication through simulation.

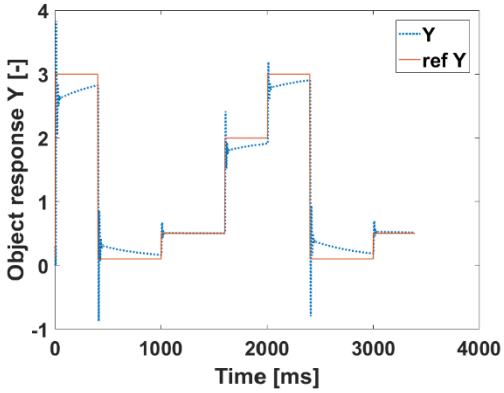

Fig. 5. Simple inertia and PID control loop simulation results.

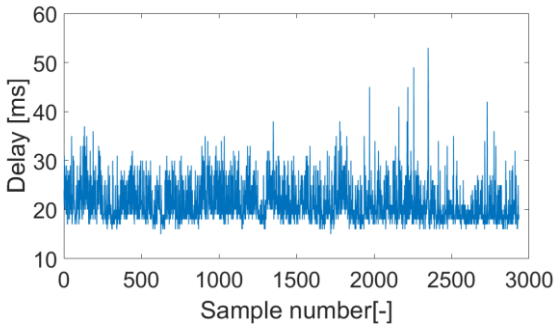

Fig. 6. WebRTC communication delays in PID control scheme.

IX. MULTIDIMENSIONAL DMC CONTROL EXAMPLE

Based on delay data obtained in studies with the PID controller, an analysis of the control capabilities of a multidimensional 2x2 facility was carried out in MATLAB / Simulink. The communication loop between the controller and the object, as well as between the object and the controller, has been simulated with previously recorded communication delays for the WebRTC protocol. Variable Transport Delay blocks presented on fig. 7 were used to include delays in control system.

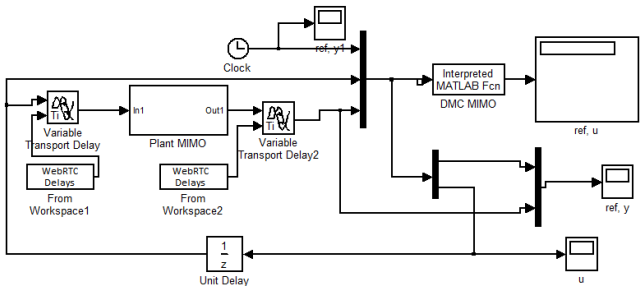

Fig. 7. MIMO DMC control scheme with communication delays.

The object is given presented in detail on fig. 8 by the following transmittances

G11k = 0.7; G11T = 0.100 s; G11tau = 0.067 s;
G12k = 0.1; G12T = 0.120 s; G12tau = 0.015 s;
G21k = 0.15; G21T = 0.080 s; G21tau = 0.010 s;
G22k = 0.8; G22T = 0.020 s; G22tau = 0.030 s;
The simulation step was taken for 1 ms.

G11k is gain from first input to first output, G21 is time constant of the control object from second input to first output (Y1), G12tau is object's time delay from first input to second output (Y2).

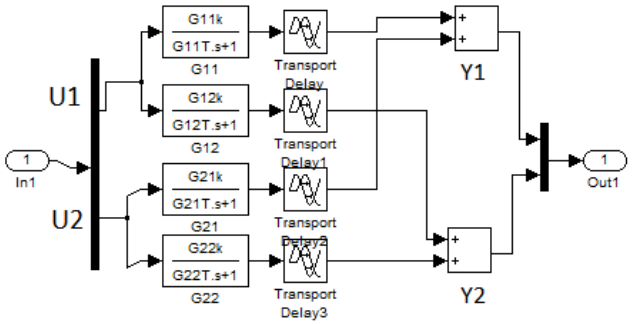

Fig. 8. MIMO plant configuration.

Multivariable DMC controller was designed based on algorithms from [7, 8]. Results of MIMO DMC are presented on fig. 9., 10. Time span of presented referenced, controlled and control signal values is 6 s.

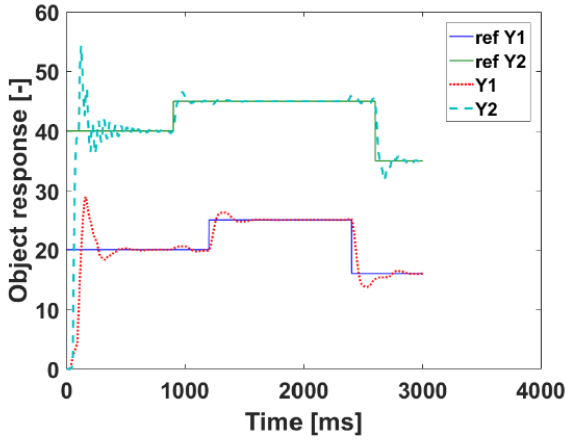

Fig. 9. The referenced and controlled values of MIMO DMC

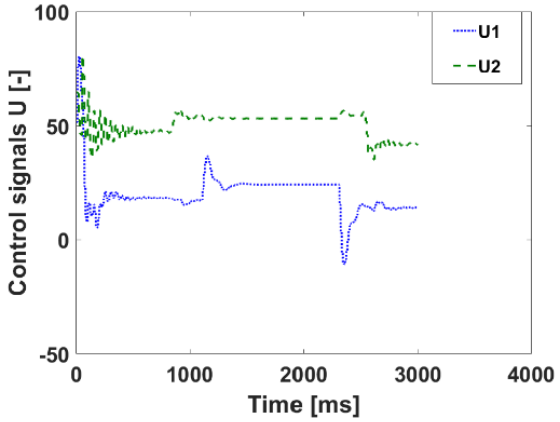

Fig. 10. Control signals of MIMO DMC

Predictive controllers usually cope with delays, including variable delays in time - when they are equipped with a mechanism for updating knowledge about the object, however, this applies to objects with slow-changing delays, not random variables. The delay values used in the test matched the time constant of the considered object. The control took place correctly.

X. CONCLUSIONS

Authors analyzed performance of popular technologies used in web browser for real time communications. The WebSockets and WebRTC protocols were characterized and their advantages were indicated. The applicability of these protocols as elements of the control loop was analyzed. Detailed research was conducted on the introduced communication delays in various networks like LAN / WAN / WiFi / GSM, etc. The classic control loop was implemented with the object and the PID controller using the WebRTC protocol. The control results were satisfactory. The registered delays were also used to simulate the operation in the predictive multidimensional control loop. Despite communication delays comparable with delays in the object and time constants of the object, the control was correct. There are already many reported uses of a web browser as a human machine interface (HMI) and even a SCADA system. The article presents that WebSocket and WebRTC network techniques can be used not only as elements of the control structure but can also be used to build a complete control loop: object, controller, HMI.

References

[1] Mozilla Web Sockets API https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[2] Mozilla WebRTC API https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

[3] Popovici K., Mosterman P. J. (Eds.), Real-time simulation technologies: principles, methodologies, and applications., CRC Press, 2012

[4] Tarnawski J., Karla T., Real-time simulation in non real-time environment, 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR), 2016, pp. 577-582

[5] Juszczuk D., Tarnawski J., Karla T., Duzinkiewicz K., Real-Time Basic Principles Nuclear Reactor Simulator Based on Client-Server Network Architecture with WebBrowser as User Interface, Polish Control Conference 2017, Cracow, Poland, 18-21 June 2017

[6] Karla T., Tarnawski J., Duzinkiewicz K., Cross-Platform Real-Time Nuclear Reactor Basic Principle Simulator, 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje 24-27 Aug. 2015, pp.1074-1079. IEEE

[7] E. F. Camacho, C. Bordons Alba, Model Predictive Control, Springer-Verlag London, 2007

[8] C. E. Garciaa, M. Morshedi, Quadratic Programming Solution of Dynamic Matrix Control (QDMC), Chemical Engineering Communications, 1986

[9] Astrom K., Wittenmark B.,Computer Controlled Systems 3rd Edition, Prentice Hall, 1997

[10] K. Warwick and D. Rees, *Industrial Digital Control Systems*, IET, 1988

[11] Normey-Rico J. E., Camacho E. F., Dead-time compensators: a survey. Control Engineering Practice, 2008, 16(4), 407–428.

[12] Vatanski N., Georges J.-P., Aubrun Ch., Rondeau E., and Jämsä-Jounela S.-L., Networked control with delay measurement and estimation. Control Engineering Practice, 2009, 17(2), 231–244.

[13] Morawski M. and Ignaciuk P., Reducing impact of network induced perturbations in remote control systems. Control Engineering Practice, 2016, 55(10), 127–138.