Post-print of: T. Karla and J. Tarnawski, "Web-based real-time simulation system," 2018 International Interdisciplinary PhD Workshop (IIPhDW), Swinoujście, 2018, pp. 310-315, doi: 10.1109/IIPHDW.2018.8388379.

Web-based Real-Time Simulation System

Tomasz Karla, Jarosław Tarnawski
Gdansk University of Technology, Faculty of Electrical and Control Engineering, Gdańsk, Poland
Email: tomasz.karla@pg.edu.pl, jaroslaw.tarnawski@pg.edu.pl

***Abstract*—The paper presents the development of a simulation system composed of a real-time plant simulator with real-time controller included in the software-in-the-loop structure using web-based communication. The client-server architecture build in a TCP/IP network environment was introduced, where the server is a computing unit for real-time high temporal resolution plant simulation (and optionally also as controllers' platform) and the web browser application is a client for controller purposes and user interface. Such a structure allows to familiarize user with the simulated plant but also to perform simulation with wide variety of control algorithms and control structures. Division of code into functional groups allows easy modifications and application of new plants and controllers for simulation. A communication latency is crucial element and can be critical in case of specific processes to correctly perform the simulation. Simulator uses previously developed mechanism of step adaptation for keeping up with real-time which allows to follow real-time operation at the expense of results quality. Clear and easy interface for signals exchange between the simulated plant and controllers and simulation was introduced. Management console, which has the ability to present data in form or numerical data or dynamically updated graphs was provided. Users have ability to manage simulation parameters and control chosen variables of the simulation through web browser GUI. Some classic control schemes of the nuclear reactor have been chosen as benchmarks and used with simple test scenarios to present capabilities of the proposed structure.**

Keywords*—WebSocket, real-time simulation, web-based simulation

I. INTRODUCTION

The simulation plays significant role in industry, research, education and training activities. It allows to analyze the response of a simulated plant to external stimuli. Simulation applications include acquaintance with the plant and the way it interacts, the ability to evaluate the control applied to the plant. Special version of simulation is real-time simulation. The real-time simulator is expected to deliver simulation results at certain moments in time, according to simulated plant dynamics. Very convenient platform for real-time simulation may be network environment with web browser. The web browser is available on almost every modern hardware and software platform. The browser can use the computational resources i.e. memory and computing power of the hardware on which it runs. As a host of the simulation system using to the standards HTML, WebSocket, WebRTC, there exists a unified way of communication between elements of the simulated control system. Due to the networking nature of the browser's work, it is possible to create distributed simulation systems with high-performance computing servers or just other browsers. Dynamic models of devices as well as entire control systems can be provided for a wide range of users. Results of simulations can be sent to many recipients. The browser also has very large presentation possibilities which can be used to build the user interface - to acquire the operator's commands and present the simulation status and results. Browser is free to use also in the access to development tools and almost everyone can use it. These properties have didactic and popularizing potential. Many services are transferred to the browser to provide broad access and interoperability within one task or project. An example may be office suites like Google Docs and Microsoft Office. The browser also has its limitations in terms of its use for simulation purposes. It is not optimized in terms of computational efficiency and does not allow to enter directly dynamic models as dedicated environments such as Matlab/Simulink, Octave or SciLab.

Real-time simulation means that the results of the simulation are given in accordance (not later and not before) with the time of the simulated element or system. This allows to familiarize user with the dynamics of the simulated plant, time dependencies between the simulated quantities. It also allows to check if the prepared control systems are sufficiently computationally efficient to work with a given model or device. Interacting with the real-time model allows to analyze the answers of the simulated plant on an ongoing basis. Other methods of simulation require the preparation of *a priori* controls (before starting the simulation) and observation of the *a posteriori* results (after simulation completion) or in a completely asynchronous (not in pace with the dynamics of the plant) manner. On digital machines, the values of simulated quantities are updated at discrete time with a certain calculation step. The calculation of new values takes time, and the results may be available ahead of time, on time or after the time for which they were calculated. Therefore, in a real-time simulation, there must be a mechanism to coordinate the delivery of results, ensuring that the results are reported at the right time. In the case of calculations completed before the time, this mechanism will stop the submission of results and calculations of the next step. In case of too long calculations, this mechanism will give results immediately after the calculation is completed and depending on the solution it will try to give next results on time (increasing the simulation step) or accepts the loss of synchronization of the results with the simulated time. Definition of the Hard Real Time HRT system appears where requirements that in all simulation steps the results are must be given on time without any violations. Soft Real Time SRT means that incidental overruns of calculation

time relative to the simulation time are acceptable. The simulation step should be selected for the needs of appropriate representation of the simulated model, the stability of the selected numerical method and the capabilities of the equipment. This article discusses the SRT simulation system with adaptive simulation step.

II. RELATED WORK

The idea of the virtual laboratory based on of simulated/emulated plants is well presented in [1]. The reference [2] indicates many examples of web based simulation applications in the field of automation. By developing concepts described in [3], [4], [5] the authors designed a framework for soft-real-time simulations based on network client-server model with web browser as simulation supervisor and user interface. Real-time issues in a network environment are current problems for engineers and scientists [6]. A real technical problem is to develop an application running under the regime of real-time, with a high temporal resolution, in an environment which is not a native real-time [7]. The adaptive multistep algorithm for catching up real-time was applied to meet real-time requirements. The real-time simulator allows to familiarize the user with the dynamics of the processes, accustom the user with sense of real-time relationships between the simulated values [8], [9]. Control experiments can be performed, and their results observed. Many examples of web-simulated plants can be found in literature. The number of publications concerning real-time simulation based on the network environment is considerably smaller. The book [10] classifies the types of simulations according to so-called rapid prototyping of the control system. The simulation method described in this article can be classified as 'Algorithm in the loop' or 'Software in the loop' simulations, i.e. situations in which a controller is attached to the plant model in the form of a control algorithm written in any language or target software form. In the described structure, it is possible (but not considered) to develop software that will provide communication with the target hardware control device (DCS, PLC) and thus provide the most advanced 'Hardware in the loop' simulation structure.
Modern SCADA systems, for example InTouch Anywhere [11] enable presenting and sharing simulated or real industrial data from a plant in a browser window. They are rather oriented to presenting synoptic screens as a website, but not as simulation environments with real-time work control.

III. SIMULATION STRUCTURES

The article presents two structures for simulation. The first one is distributed, shown in Fig. 1, requires at least two separate units of computation, in which one plays the role of the plant simulator and the other the control system and the user interface. The second structure is centralized. One computing unit is the plant simulator and control system. The simulation and user interface management system are located on the second unit. The distribution of roles in a distributed structure enables to build a library of plants and place their simulation models on very efficient machines, eg corporate or university servers. Access to simulated plants and the possibility of control is via client browsers. Communication delays appears due to this structure which should be taken into account in the simulation. The usefulness of this structure is preserved when the transport delays together with the calculation time on the server do not exceed the expected simulation step. In this structure, the user's commands are read in accordance with the current simulation status and the result is presented on an ongoing basis. In a centralized structure, there is no problem of transport delays between the plant and the control system, but the user is notified of simulation results with a delay and his commands are also applied to the plant with a delay. Only the distributed structure will be presented in detail.

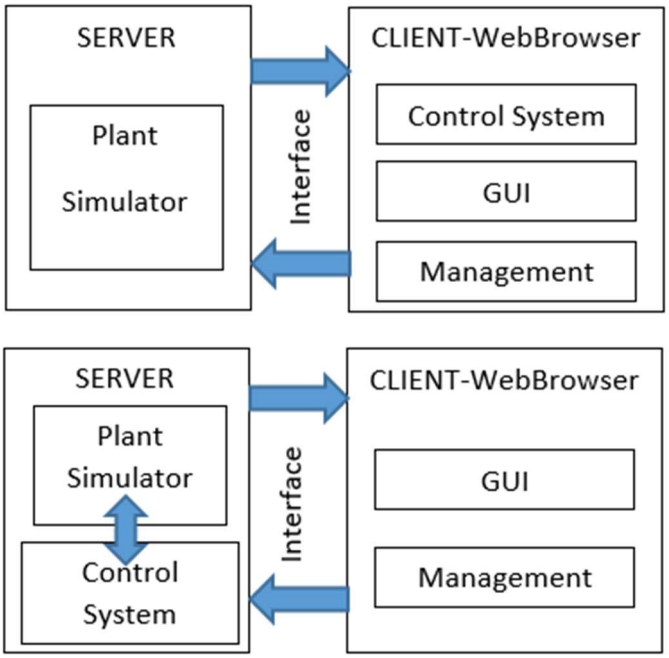

Fig. 1. General structures of simulations.

IV. A RELATION BETWEEN THE QUALITY OF THE SIMULATION AND THE LENGTH OF THE SIMULATION STEP AND THE SIMULATION TIME

The main parameter of the real-time simulator is the simulation step. The described distributed structure causes the simulator code to be executed on efficient digital machines, but communication delays should be taken into account. The presented simulator is supposed to operate according to the idea of SRT, i.e. there may be occasional exceedances of the simulation time, but in the case of such a transgression, the simulation monitoring mechanism is supposed to "catch up time" by increasing the simulation step which may affect the simulator's quality performance. Thus, the issue of the relationship between the quality of simulation and the step of simulation appears. As a measure of the quality of the simulation, its compliance with the model simulation implemented without the real-time regime was accepted.

Another important issue is the numerical stability of the simulation and the inability to infinitely increase the step of simulation. In this case, the maximum allowable simulation step is entered to ensure correct calculation. The idea of adaptive simulation step length is presented on fig. 2 where: t_b - time of basic calculations, $t_{o,k}$ - calculation time in step k which exceeds the base calculation time, $t_{+,k}$ - estimated time needed to perform calculations, $t_{nb,k}$ - adaptively selected calculation step, $t_{no,k+1}$ - new calculation time enabling synchronization in the future with the step of basic calculations. Detailed information can be found in paper [9].

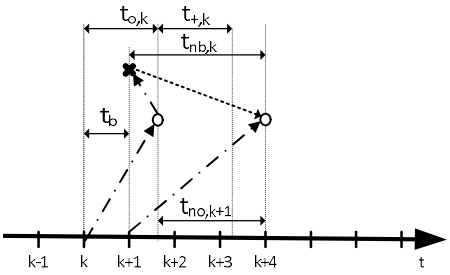

Fig. 2. Mechanism of catching up real time during simulation

V. SIMULATION SYSTEM

The most widely available technologies, allowing for real-time data exchange using the browser are WebRTC and WebSocket. The first one is P2P communication type, which allows for a significant dispersion of communication and direct communication between clients. It also natively takes into account aspects of real-time communication, but it is not supported by all browsers and incompatibility problems between different browsers are reported. In addition, despite the character of P2P, an additional server is most often used to set up the connection in order to simplify the entire structure. WebSocket, in turn, is more of a classic client-server architecture, where it is easier to control two-way communication, is easy to implement and have a wealth of documented usage. From the scientific and engineering point of view, it would be interesting to compare WebRTC and WebSocket closely to the possibilities, ease of use and results of communication delays. The authors plan such work in the future. This article uses the WebSocket communication technology paired with node.JS to prepare software environment for the system. The developed system allows for easy implementation of many simulation plants in JavaScript and the extension with new functionalities depending on the needs. Simulation calculations for selected plants are carried out on an efficient computing unit that acts as a server. By connecting to the simulation server directly using a web browser, it is possible to use a prepared GUI that can be used by users without advanced technical knowledge. Simulations can be carried out in two modes. The first one is based on a centralized structure, the second one with a distributed structure. The basis for the operation of the distributed system structure is the communication interface developed for the purposes of the article with the use of socket.io events. Individual events are distinguished into several groups related to: connection set-up and disconnection, simulation initialization, main simulation loop and user requests. After connecting to the server, the user can obtain information about available events and parameters of simulation plants, thanks to which he can prepare his own interface and algorithms. Table I presents the most important events of the developed interface and fig. 3 presents general hardware-software structure of the system.

TABLE I. MAIN SOCKET.IO EVENTS USED IN DEVELOPED INTERFACE.

Type	Client	Direction	Server
Initialization of connection	Connecting to server: socket.connect("http://...")	→	On connection event: socket.on("connection" , function(data){…})
	Display welcoming message from: socket.on("welcome" , function(data){…})	←	send welcoming message: socket.emit("welcome" , {data})
	Acquire information about available plants: socket.on("objects_info" , function(data){…})	←	and information about available plants and their properties: socket.emit("objects_info" , {data})
Initialization of simulation	Prepare simulation, set required parameters (like plant number, step-time etc.): socket.emit("set" , {data})	→	Calculate initial state of plant, set default values of parameters from: socket.on("set" , function(data){…})
	Prepare necessary input for step of simulation (control values, step time adaptation etc.) based on initial data: socket.on("set_done" , function(data){…}) or previous step data	←	and send information of initial state when all necessary operations are done: socket.emit("set_done" , {data})
Simulation Main loop	and send processed data to calculate next simulation step: socket.emit("calculate" , {data})	→	Calculate step of simulation with received data: socket.on("calculate" , function(data){…})
	Process and store results from: socket.on("calculate" , function(data){…}) in temporal buffer	←	and send results: socket.emit("results" , {data})
Conditional	Send data from buffer for archiving purposes: socket.emit("save" , {data})	→	Write to file acquired data from: socket.on("save" , function(data){…})
On user request	Send command to prepare archived data for download: socket.emit("download" , {data})	→	Receive command to prepare for for download: socket.on("download" , function(data){…})
	Acquire link to file: socket.on("download_ready" , function(data){…})	←	save leftover data, compress file send link to prepared file: socket.emit("download_ready", {data})
Disconnection	Disconect from server: socket.emit("disconnect" , {data})	→	Disconnecting information: socket.on("disconnect" , function(data){…}), close connection, remove unnecessary data and files, free memory

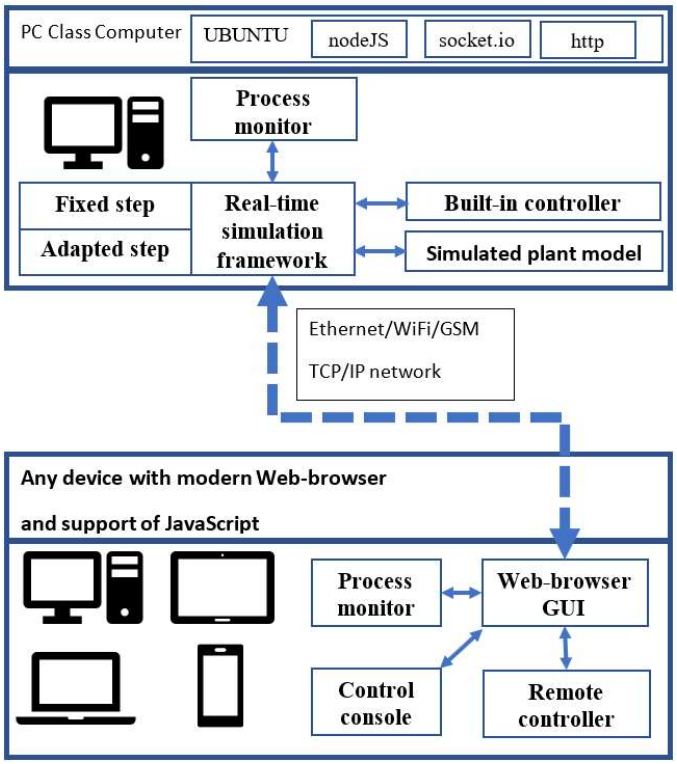

Fig. 3. Hardware-software implementation of distributed simulation stracture

VI. EXAMPLE SIMULATIONS IN SIMULATION SYSTEM

For the presentation purposes authors implemented two plants in simulation framework and PID controller with AntiWindup and DeadZones capabilities. The first considered plant is a simple first order linear system (inertia). Second plant is much more complex, represents basic principles of nuclear reactor [12] and includes processes with varying dynamics from very fast one (like neutron kinetics) to very slow (like burning up nuclear fuel). Developed model [8][9] of simulation plant is composed of 18 differential equations and over 100 linear equations. Differential equations were resolved using Runge-Kutta 4 order method. Simulator server hardware-software platform was composed of: PC Class Computer with i7 4790k CPU, 32 GB RAM and fast SSD disk. As for the client units, there were 2 test units, one smartphone, which was used only for monitoring simulations, and one laptop, on which later were stored results of simulation. Both unit were within the same local network but accessed with different medium (laptop was using Ethernet cable, smartphone Wi-Fi AC).

Tests included simulating both plants in control system loops with PID. In case of reactor plant, from all the processes, thermal power was chosen as controlled value. Both plants were tested with prepared testing scenarios which were composed of step changes in set points at specific times.

Simulations were performed with the use of prepared WebBrowser GUI. It allowed for full control over simulation process. All necessary simulation and controller parameters were possible to change in form of inputting numerical values in specific textboxes or using knobs. Prepared GUI allowed for presenting real time trends of chosen simulated processes values as well as simulation control values (delays, communication times, step time etc.). Fig. 4 presents developed GUI for operators, where is it possible to adjust parameters of simulations and parameters of controllers.

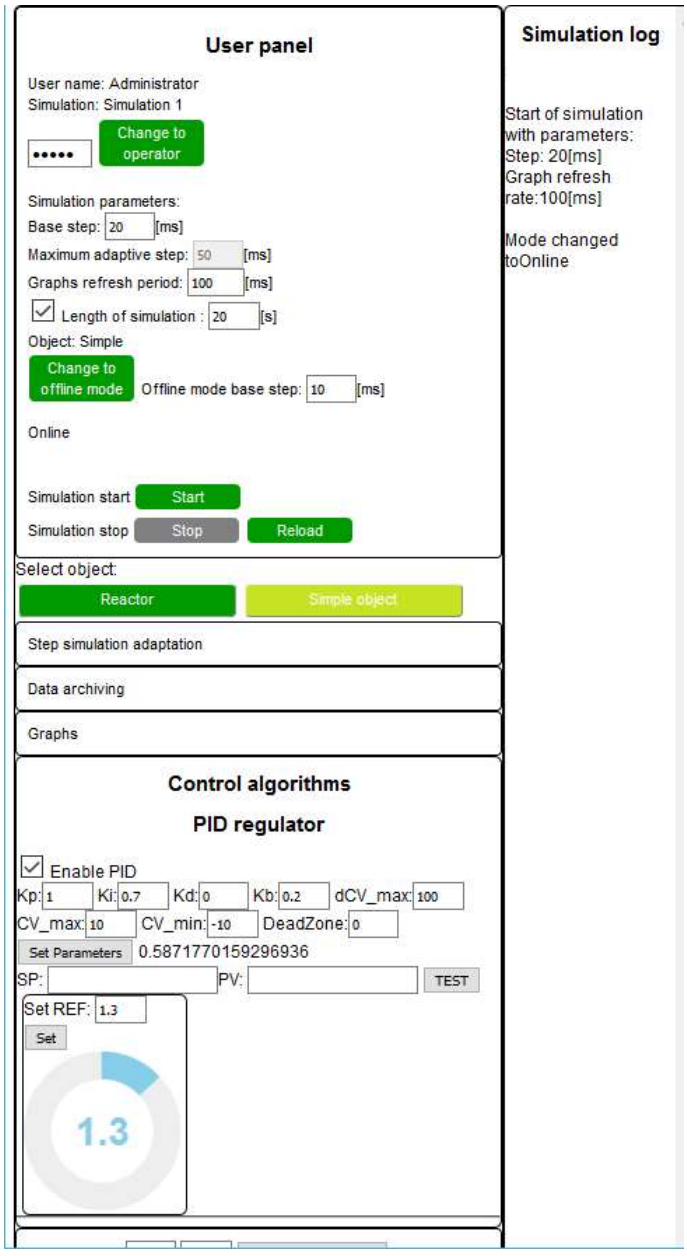

Fig. 4. Developed GUI – control panel.

The values of selected process variables can be displayed for the user in two different forms. One is simple display of instantaneous values of the process variables, other one is displaying registered values of the processes through simulation in form of a trend. Fig. 5 presents an example of time trends for relation between real and simulation time (deviation from real time) and the response of simple plant (inertia).

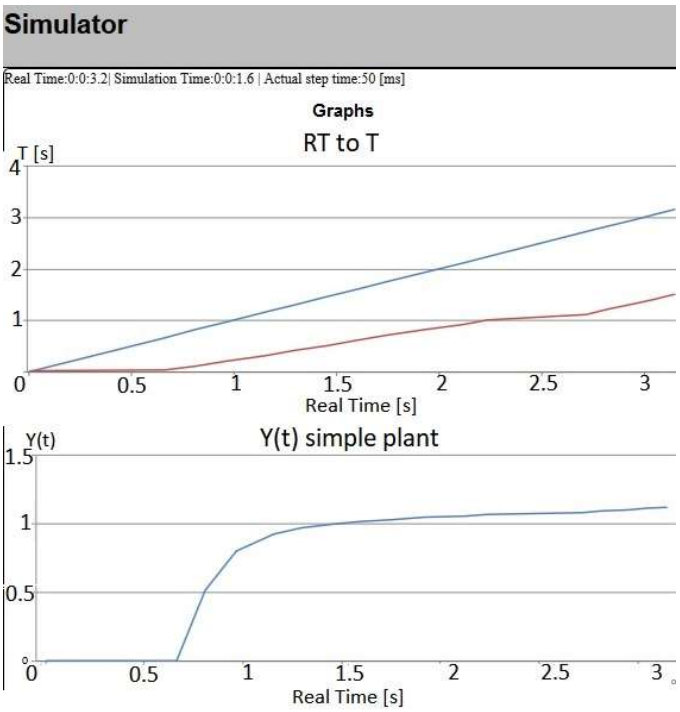

Fig. 5. Developed GUI – graphs.

Tests of both plants were conducted in distributed structure mode with adaptive step length of simulation for real-time operation with base step 10 ms. Both simulation results were compared with reference data obtained from non real-time simulation with fixed step 10 ms (for simple plant output was simply Y(t) of first order plant, for reactor plant simulation results present change of thermal power). Fig. 6 and 7 show results of both simulations, simple plant and reactor plant respectively. In each pair of trends top trend is showing results of simulation at calculated simulation times, bottom one obtained simulation results in real time.

Like mentioned before, difference in quality of results between reference simulator and simulation system were expected. Main reason for differences are occurring varying delays in communications, which require compensation by step adaptation (fig.8 presents observed communication delays in one of performed simulations and their cumulative distribution function). In case of very simple and fast plant the difference is easily visible and can be quite big on short periods of time, but average error of whole simulation in comparison to reference data is small. In this case there wasn't much of difference in real-time execution and calculation time of reference data.

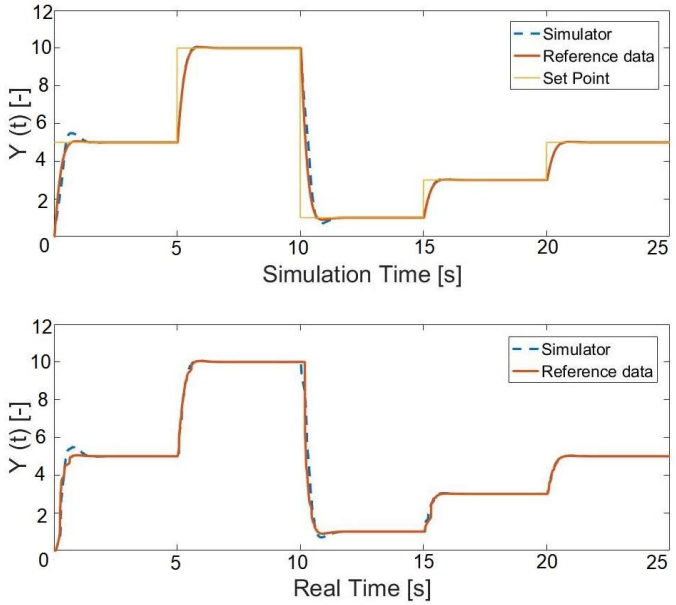

Fig. 6. Simple plant simulation results.

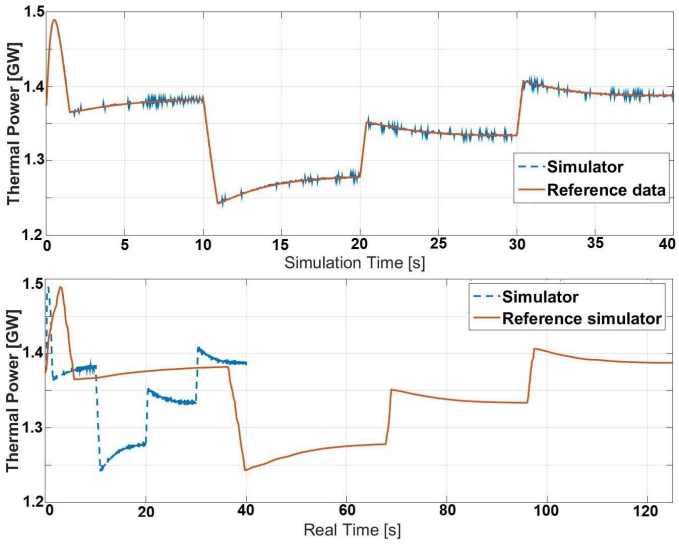

Fig. 7. Reactor plant simulation results.

In case of much more complex reactor plant results also showed differences, but the value of maximal observed error was below 0.5% in comparison to reference value which is acceptable. But what is clearly noticed is difference in computation time. The time to calculate reference data took more than three times more time than their simulation period (clearly visible by comparing top and down trends in each figure). Fig. 9 presents time differences observed in reactor plant simulation between calculated simulation time and real time. Table II presents calculated errors of simulations.

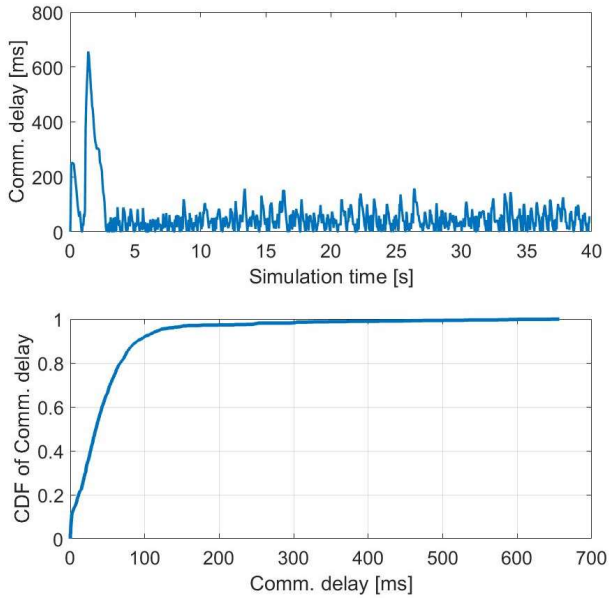

Fig. 8. Reactor plant simulation varying delays of communication.

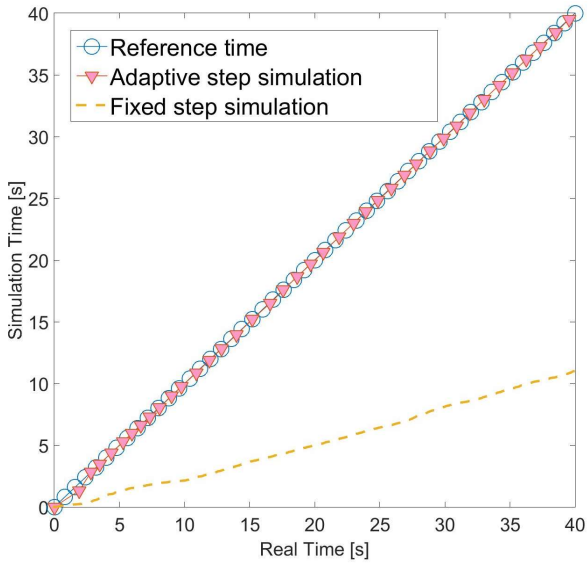

Fig. 9. Simulation times compared to real-time for different types of simulations

TABLE II. OBSERVED ERRORS OF SIMULATIONS IN TEST SCENARIOS

	Simple plant	**Reactor plant**
Maximal instantaneous error	57%	0.42%
Average of relative errors	1.79%	0.07%

VII. CONCLUSIONS

The real-time simulation system based on network solutions and a web browser open up new possibilities for presentation, education and popularization. The clear, functional interface between simulated plant and controller was introduced. The software-in-the-loop scheme allows to set up almost any combination of real-time control system with a simulated in the server plant model. The simulated schema presented in the paper has many advantages: accessibility across the Internet, availability on any hardware and an operating system, free software, open programming, simulation flexibility, scalability and the ability to build an arbitrary a user interface. A functional example of the application of this structure has been demonstrated in the article. The example of an extremely time-sensitive plant that is a nuclear reactor was introduced. Control effects are graphed in browser windows similar to those from simulation packages or HMI applications. The obtained results were consistent with those obtained in the classic simulation software which confirms the correctness of the implementation. WebRTC technology with the possibility of managing communication delays will certainly be the subject of future work in this field.

ACKNOWLEDGEMENTS

The research work was done under grant Polish MNiSW 8902/E-359/M/2016: Young Researcher Support Program. The authors wish to express their thanks for support.

REFERENCES

[1] G.C. Goodwin, A.M. Medioli, W. Sher, L.B. Vlacic, and J.S. Welsh, “Emulation-Based Virtual Laboratories: A Low-Cost Alternative to Physical Experiments in Control Engineering Education.”, IEEE Transactions on Education, vol. 54, no. 1, pp. 48 - 55, February 2011

[2] Ch. Dufour, C. Andrade, J. Bélanger, “Real-Time Simulation Technologies in Education: a Link to Modern Engineering Methods and Practices”, Proceedings of the 11th International Conference on Engineering and Technology Education INTERTECH 2010 Ilhéus, Bahia, Brazil, 2010

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

[4] J. Byrne, C. Heavey, P.J. Byrne, “A review of Web-based simulation and supporting tools”, Simulation Modelling Practice and Theory, vol. 18, no. 3, pp. 253-276, Elsevier, March 2010

[5] S.M. Kuo, B.H. Lee, W. Tian, Real-Time Digital Signal Processing: Implementations and Applications, John Wiley & Sons, Ltd, 2006

[6] G. Carlucci , L. De Cicco , S. Holmer , S. Mascolo, “Congestion Control for Web Real-Time Communication”, IEEE/ACM Transactions on Networking, vol. 25, no. 5, pp. 2629-2642, June 2017

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

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

[9] D. Juszczuk, J. Tarnawski, T. Karla, K. Duzinkiewicz, “Real-Time Basic Principles Nuclear Reactor Simulator Based on Client-Server Network Architecture with WebBrowser as User Interface”, Trends in Advanced Intelligent Control, Optimization and Automation, pp. 344-353, Springer, 2017

[10] J. Eickhoff, Simulating Spacecraft Systems, Springer, 2009

[11] Wonderware InTouch Access Anywhere User Guide, access 25.02.2018 https://www.logic-control.com/datasheets/1/InTouch/ITAA_UserManual.pdf

[12] Y. Oka, K. Suzuki, Nuclear Reactor Kinetics and Plant Control, Springer, 2013