The results reported here represent the first stage in the development of a full-featured laboratory system aimed at studying machine learning algorithms. The relevance of the current work is predetermined by the lack of network small-size mobile robots and appropriate control software that would make it possible to conduct field experiments in real time. This paper reports the selection of network data transmission technology for managing mobile robots in real time. Based on the chosen data transmission protocol, a complete stack of technologies of the network model of a multi-agent system of mobile robots has been proposed. This has made it possible to build a network model of the system that visualizes and investigates machine learning algorithms. In accordance with the requirements set by the OSI network model for constructing such systems, the model includes the following levels:

1) the lower level of data collection and controlling elements – mobile robots;

2) the top level of the model includes a user interface server and a business logic support server.

Based on the built diagram of the protocol stack and the network model, the software and hardware implementation of the obtained results has been carried out. This paper employed the JavaScript library React with a SPA technology (Single Page Application), a Virtual DOM technology (Document Object Model), stored in the device's RAM and synchronized with the actual DOM. That has made it possible to simplify the process of control over the clients and reduce network traffic.

The model provides the opportunity to:

1) manage the prototypes of robot clients in real time;

2) reduce the use of network traffic, compared to other data transmission technologies;

3) reduce the load on the CPU processors of robots and servers;

4) virtually simulate an experiment;

5) investigate the implementation of machine learning algorithms

Keywords: multi-agent systems, mobile robots, machine learning, network model, WEB interface, WebSocket

UDC 004.75
DOI: 10.15587/1729-4061.2020.213989

BUILDING A MODEL OF NETWORK INTERACTION BETWEEN THE COMPONENTS OF A MULTIAGENT SYSTEM OF MOBILE ROBOTS

V. Diduk
PhD*
E-mail: inokc@i.ua
V. Hrytsenko
Doctor of Pedagogical Sciences*
E-mail: grytsenko@vu.cdu.edu.ua
A. Yeromenko*
E-mail: eromenko.andrey@gmail.com
*Department of Automation and Computer-Integrated Technologies
The Bohdan Khmelnytsky National University of Cherkasy
Shevchenka blvd., 81, Cherkasy, Ukraine, 18031

Received date 25.08.2020
Accepted date 09.10.2020
Published date 22.10.2020

1. Introduction

Finding optimal ways to solve the tasks of big data analysis puts forward high requirements for pre-training of the programmer, mathematician, and other specialists of related professions. If these specialists lack the necessary knowledge and experience, they must spend extra time studying, or refuse to further participate in research or development projects. The area of machine learning, which has gained traction recently, makes it possible to resolve the above-mentioned issues. By using the theoretical provisions of machine learning, programmers may skip developing bulky software products that could include all possible solutions. Instead, it becomes possible to apply one of the generally accepted algorithms that would independently find the optimal solution through the comprehensive analysis of existing data. The result from the analysis could make it possible to predict, draw conclusions, make decisions, etc.

The development of microprocessor technology and modern advances in robotics have opened a new field of using the machine learning algorithms. At a modern rate of increasing the number of robots, it is a relevant task to develop algorithms for data analysis and autonomous decision-making about a robot's behavior. Examples of such robotic systems include cars, unmanned drones, robot couriers, etc. Implementing a closed system within one technical unit makes it possible to minimize the number of perturbations; the implementation of an autonomous control system over a robot is not difficult. When employing a significant number of autonomous mobile robots, the process of managing each individually becomes resource-intensive and difficult to implement [1]. In such systems, it is also optimal to use machine learning algorithms.

Machine learning algorithms are typically tested using a computer by analyzing a significant amount of data; the analysis result is information presented in the text form. At present, this approach is not optimal when real robots are involved, especially in multi-agent systems, because it does not allow the visualization of solution results and conducting a field experiment in dynamics. Existing models of laborato-

ry robots do not provide the capability to change a program during the experiment, or have excessive dimensions and high cost. All this makes it impossible to merge them into multi-agent networks to investigate machine learning algorithms.

Thus, it is a relevant task to develop systems for investigating the multi-agent management of collective mobile robots that can function autonomously and make decisions independently.

2. Literature review and problem statement

The body of research into multi-agent robotic systems increases every year, as evidenced by a significant number of publications, specifically works [2–13]. The least studied are the possibilities of simultaneous control over all components of such a system, or their autonomous interaction. Such tasks include group behavior management: monitoring the motion trajectory, controlling robot collisions, tactical maneuvers, distributing the tasks of monitoring open or closed areas, allocating data processing tasks, etc. Up to now, the use of machine learning in the multi-agent mobile robot systems has been examined in studies [2, 5, 6, 8–10, 12, 13], and others. Thus, papers [2, 4, 10] reported only the analysis of the following:

1) the possibilities of scaling multi-agent systems of mobile robots;

2) the limit number of agents and the size of the work field;

3) the main areas of research that are necessary for the development of such systems without specific solutions for their construction.

In [3], attention is paid to the development of synchronous operator control systems for a single robot, as well as synchronous multi-operator multi-robot systems. The limitation of the work is the lack of the system's capability to work without the participation of an operator, the absence of the possibility to collect data from robot operation and to verify machine learning and control algorithms. Paper [5] reports the results of developing a system of two robots and testing the Dyna algorithm. The disadvantage of the system is the lack of feedback from robots to the central computer, with their movement only parallel to the rectangular coordinate system. Works [6, 11] describe approbation results for the algorithms of canonical variational analysis by La-Remor and GraWoLF, respectively. Studies [8, 9, 13] report the results from the self-learning of the system with a limited number of robots. A common limitation of the cited studies is the need to program each robot separately and the lack of the possibility to collect information at the server. All judgments on the effectiveness of the algorithm operation, as well as the system in general, are passed by researchers indirectly, based on the results of observations. Work [7] is mainly a review analyzing the maximum number of a system's agents within a single work field. Paper [12] describes a significant number of machine learning algorithms, as well as the models of building physical systems of mobile robots. A common feature of all available studies is the use of closed systems configured to perform a pre-programmed single algorithm of agent operation (robot). When changing the tasks of research, developers are forced to program each robot separately. Accordingly, the time spent to change the program increases in proportion to the number of agents involved in the system. The effectiveness of an algorithm is estimated by the researchers mainly based on the results of visual observation. The disadvantages of such systems include:

1. The difficulty of changing the activity program of each robot.

2. No ability to accumulate data at the server.

3. The lack of an interaction interface between a user, who is not a controller programming specialist, and the system's agents.

Our analysis gives reason to assume that, at present, there are quite powerful mathematical models for building multi-agent systems and the interaction among their components. However, there is no any effective toolset to investigate their operation. It may prove optimal to design a WEB-oriented control system, within which the entire algorithm of data processing is to be implemented at the server while control over each agent should be centralized. Such an approach could make it possible to:

– use any robot in a system to implement arbitrary algorithms without changing its software;

– accumulate the collected data at the server for further analysis or application as the source data for subsequent experiments;

– increase the number of potential researchers by designing a simplified interface and providing a remote access to the system.

3. The aim and objectives of the study

The aim of this work is to build a model of network interaction of the multi-agent system of mobile robots, which can be used in real time by researchers of any level of training for the implementation of machine learning algorithms.

To accomplish the aim, the following tasks have been set:

– to choose a network data transmission protocol to manage remote microprocessor tools in real time;

– to select a stack of technologies and define the role and relationship of components of the model of a multi-agent system of mobile robots;

– to verify the theoretical provisions proposed in this study using modern software and hardware development tools.

4. Choosing a real-time network protocol for managing remote microprocessor tools

Modern Internet connection and Internet of Things (IoT) devices are partly based on HTTP (Hypertext Transfer Protocol), Websocket, and MQTT protocols. If a real-time response is not essential, one typically uses HTTP; however, the protocol cannot be used to organize the client-server connection required for this study tasks.

One way to solve the problem is to apply the principle of Polling, to send AJAX (Asynchronous JavaScript and XML) requests at short intervals [14]. As soon as the server receives the request, the response is immediately sent, the connection is broken, and the server is ready to process the next request (Fig. 1).

The disadvantages of this method are data oversaturation (since Header is sent in all queries, the traffic is high) and query determination.

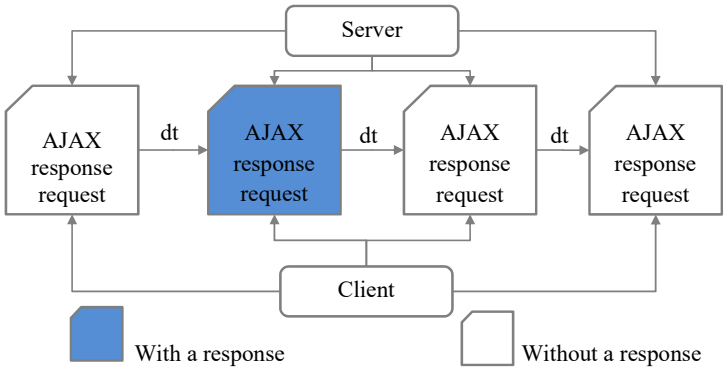

Fig. 1. Data transfer diagram using the Polling method

The issue of query determination is partially resolved by Long Polling, a method under which the client sends an HTTP request but the server can delay the response and send it to the client at the right time. If the response is not sent within a certain period, the client disconnects from the server and creates a new request (Fig. 2).

Decreasing the discreteness of requests increases the network traffic, which is not acceptable for real-time control systems.

A modification of the Long Polling method is Streaming, whereby the server does not constantly maintain a connection and sends the user information in small portions – packets (Fig. 3).

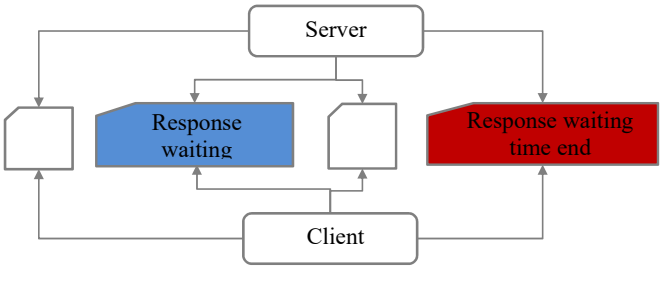

Fig. 2. Data transfer diagram using the Long Polling method

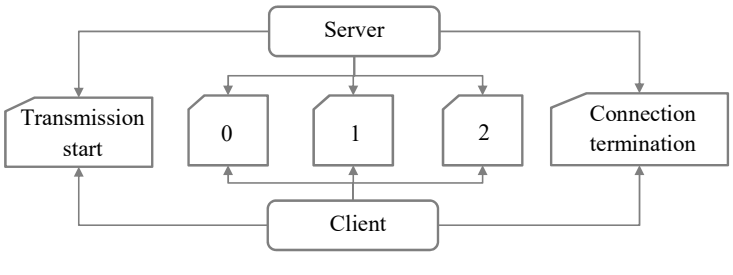

Fig. 3. Data transfer diagram using the Streaming method

The advantage of a given method is the absence of delays associated with the constant creation of requests. A header of the request is sent only when connected to the server, which reduces data traffic.

The disadvantage of the method is one-way communication, which is not suitable for modeling machine learning algorithms.

The WebSocket data transfer protocol combines the advantages of all the above-mentioned methods by implementing asynchronous, client-server "request-response" interaction. The server and the client under this approach are equal participants in the exchange of data. All interaction between the server and the client takes place using the WebSocket API [14].

Fig. 4 shows the process of establishing a connection, receiving, sending messages, and breaking the connection between the client and the server.

Thus, the most appropriate is the development of a model of a multi-agent system of mobile robots with real-time control based on the WebSocket protocol. In addition, the use of a given protocol helps reduce the use of Internet traffic, and reduce the CPU usage of the client and server.

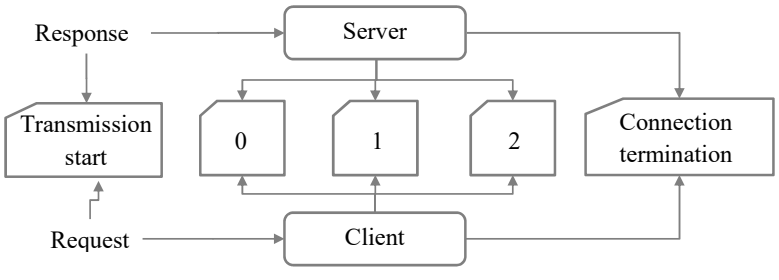

Fig. 4. Data transfer diagram using the WebSocket protocol

5. Selecting a technology stack and constructing a model of a multi-agent system of mobile robots

Since it was determined that for the convenience and ease of access to the control subsystem the entire system is WEB-oriented, the user interacts with the system using a WEB-application. As a projection on the selected technologies, the network model of a multi-agent system of mobile robots aimed at studying the performance of artificial intelligence algorithms is shown in Fig. 5. The main advantages of choosing this approach are its cross-platform nature, as well as the remote access for all participants of the process, etc.

A modern pattern in the WEB development industry is the separation of back-end and front-end servers. This approach makes it possible to reduce the load on a BLL (Business Logic Layer) server, which handles user requests and sends orders to the robots-clients.

UIL (User Interface Layer) comprises all resources (media files, images, JSX templates, CSS and JavaScript files) that are required to build the page of the application that the user sees. They are stored at a separate front-end server that sends them to the client when the application is downloaded.

To build UIL – a WEB-application for the user, the JavaScript library React is used in this work (Fig. 6). It provided an opportunity to use an SPA (Single Page Application) technology implementing Client Side Render.

The BLL is based on Node JS and the Express framework. In the development of the software part of the model, React was used to provide an additional plugin that makes it possible to apply TypeScript. The disadvantage of JavaScript is the lack of strict typification, which impairs both the stability of operation and the security of applications. TypeScript adds the possibility to strictly typify variables.

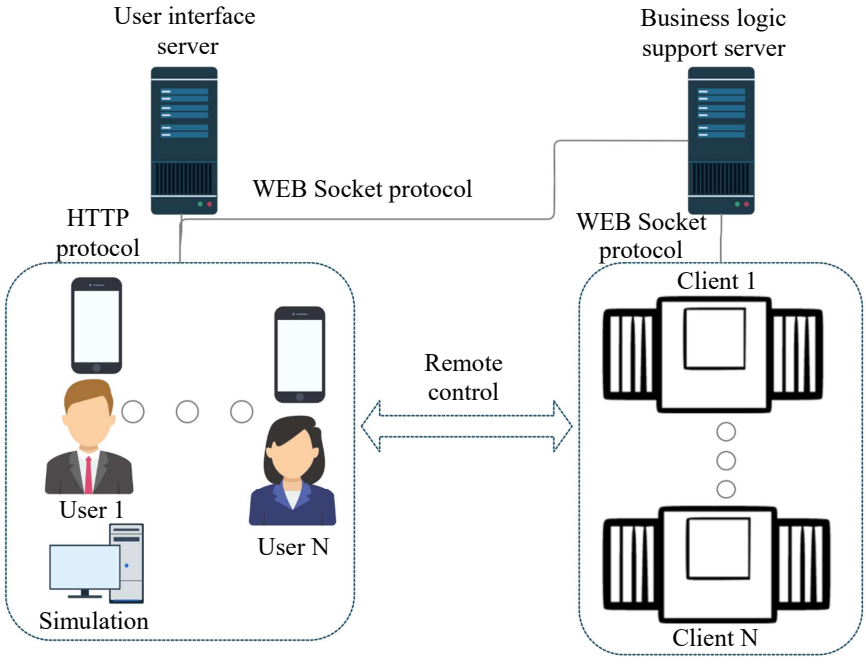

Fig. 5. Model of network interaction between the components of a multi-agent system of mobile robots

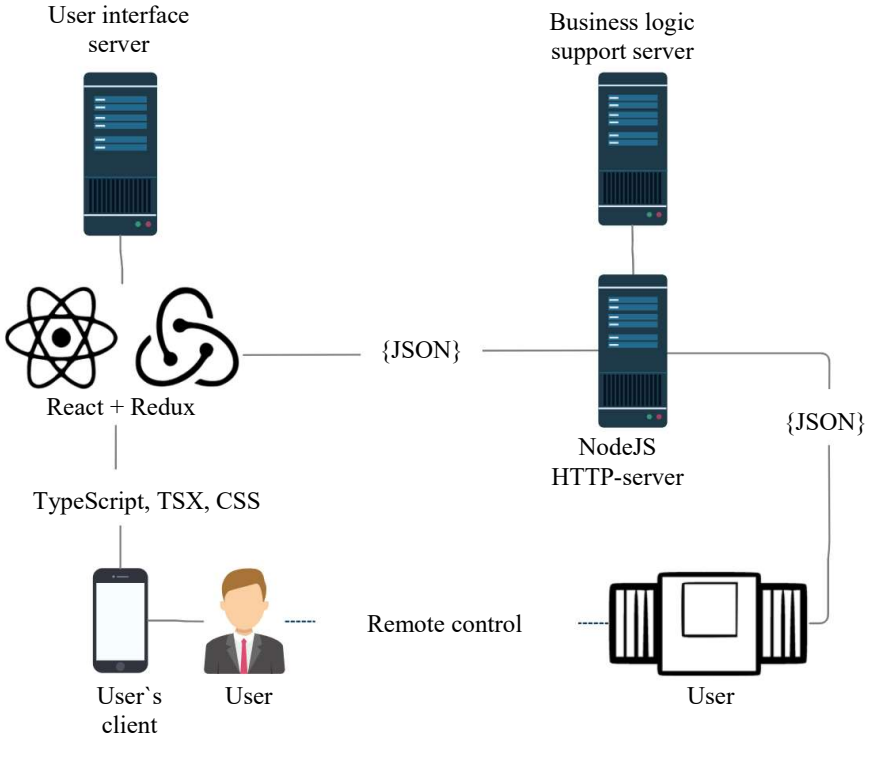

Fig. 6. The diagram of technologies used

For the convenience of storing data and building the React components of the user's application, the Redux library has been used, through which data sent from the server through the API create events (actions), interact, and combine with the data already stored in the application, then enter the global storage where they can be used in any part of the application.

The JavaScript library React has been used in our study to implement the SPA paradigm. The first time one loads a page, React sends only one initially unfilled index.html template to the client. Along with it, the user receives JavaScript files that implement JSX templates, that is the templates containing the React components based on which a virtual DOM (Document Object Model) is built, which is an idealized state of the user application interface. Virtual DOM is stored in the RAM of the device and synchronizes with the actual DOM by the methods of the React library.

When new information arrives from the back-end server, the state of the components in Virtual DOM changes. To

this end, WebSocket was applied, although one can use RestAPI and AJAX. The React library compares the software branches in VDOM and DOM. When a mismatch is detected, the page is rerendered (Fig. 7).

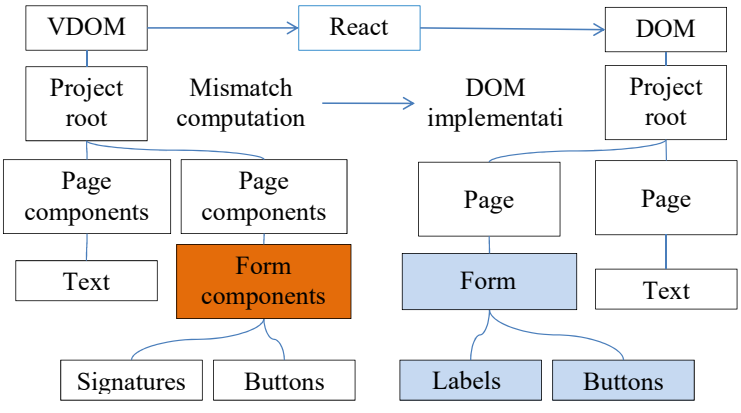

Fig. 7. The structure of DOM based on VDOM

The latest version of React has been used in our study to provide access to hooks, the features that make it possible to employ the local state of a component and other React functionality without using classes [15].

When using a functional style, the React components are declared as functions. The Arrow Function Declaration method was applied in the software implementation of the models when creating functions. All functions are anonymous and are assigned to the constant that is required to call them. To avoid failures in managing a significant number of agents, one needs to adhere to strict data typification and store their signatures. When declaring the component, clear types of input arguments and the result of the function operation should be specified. All React functional components return the FC (Functional Component) type, which is JSX markup (Fig. 8), based on which the VDOM is built.

6. Verifying the model of a multi-agent system of mobile robots aimed at studying machine learning algorithms

To check the results of our theoretic and practical work, the possibility of software simulation of the multi-agent system of mobile robots was implemented in the study. The simulation of robot behavior was implemented using a canvas technology, which is an element of HTML5. The Canvas makes it possible to include elements of the raster 2D graphics in the web page, as well as to construct hardware-accelerated 3D graphics based on WebGL.

The application consists of two sections: Control and Simulation; in Fig. 8, the left and right sides, respectively.

To access the control page, the user must select a username and connect to the virtual client, "Cube", by filling out the form fields in the pop-up window (Fig. 9).

The user name and ID fill forms are validated at the server side; if an error occurs, the user receives a message with the appropriate text.

By filling in the field "Set cube" with the robot identification number, not connected to another user, the user establishes a connection to the robot and accesses the control section.

The user can interact with the robot's hardware, including the robot LED matrix used to generate the individual "Cube" robot label, as well as move control panel of a separate robot (Fig. 8).

The "Simulation" tab contains virtual models of robots "Cube", and makes it possible to perform both virtual simulation of the system operation and visualize the actual behavior of the system of mobile robots (Fig. 10).

A *lazy loading* technology has been used to optimize the performance of the application. Since the user's application consists of two large sections: a control joystick and a simulation window, only one part of the VDOM tree with the required section is loaded into RAM at the same time (Fig. 10).

At the time of development of the system, only two experimental hardware units of the "Cube" robots were built. Robots represent a two-wheeled platform with a central axle and a zero clearance. Given such a solution, the robots have a small size of the central unit, 50×50×50 mm. An LED matrix was installed into the cover of the robot to generate a graphics label, which would make it possible to distinguish between each individual client in the work field. To test the control interface and a load on the back-end server, a simulation of the system behavior was created. This simulation is a virtual environment hosting the models of the "Cube" robots. The models can be both virtual and match the commands received by the experimental samples attached to the system.

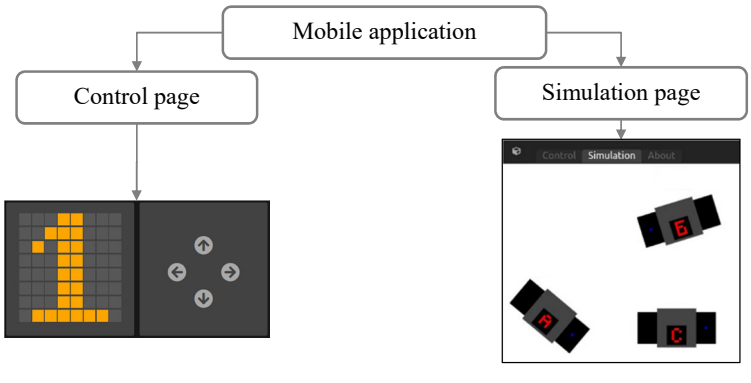

Fig. 8. A multiagent system's control and simulation pages

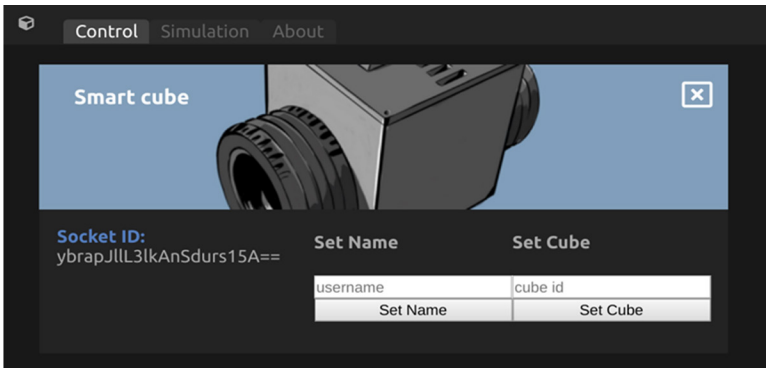

Fig. 9. "Cube" robot connection form interface

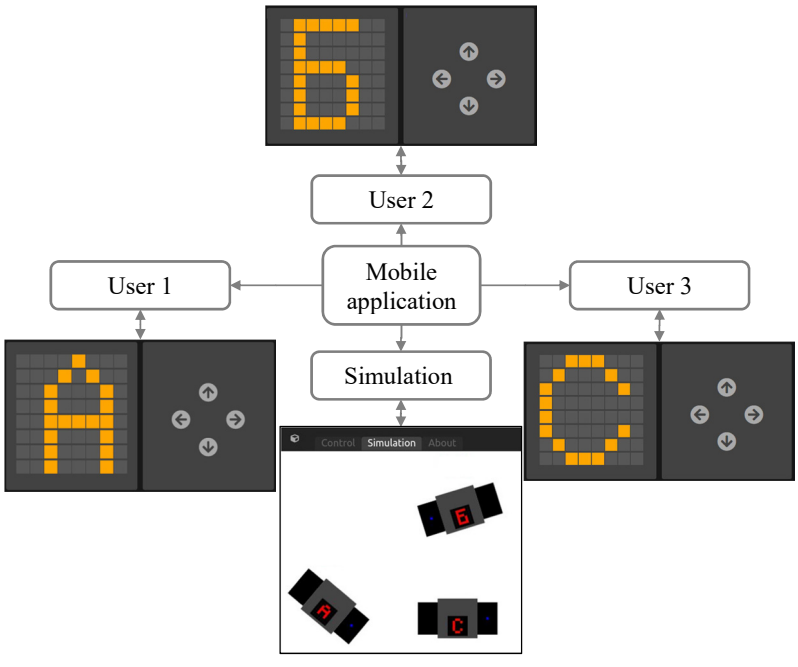

Fig. 10. The diagram of models' interaction during simulation and the system's users

Together with the commands to the engine, the server sends updated data about the position of the rotation point, the angle of rotation, and where exactly the rotation point of the robot is located. Thus, the theoretical results of our study have been implemented using a virtual model and partly with the use of hardware, which allowed us to check the feasibility of the developed theoretical provisions and outline the prospects for further scaling of the system.

7. Discussion of results of building a model of the network interaction between the components of the multi-agent system of mobile robots

The main task for modern engineers- developers working with artificial intelligence algorithms is to verify existing and build new algorithms. At the same time, an important issue is the possibility of checking the results of their work using actual models. A typical solution is the development and approbation of algorithms for a specific task. This approach narrows the range of potential researchers, since not every research institution can allow the purchase of costly robots or the participation in commercial developments. Instead, many research universities are staffed with mathematicians engaged in theoretical development. Sometimes their work does not receive appropriate feedback in the scientific environment due to the lack of tested results involving actual samples. The proposed model is a versatile toolkit that could minimize hardware costs and would, through a simplified WEB-oriented user interface, ensure accessibility for researchers at all levels of training. At present, within the framework of our work, two experimental physical samples have been implemented – the "Cube" robots whose hardware implementation is described in [16]. The WEB interface contains a manual control page (Fig. 8–10), which will be further supplemented with the functionality of the command designer to construct the algorithm of the automated robot activity control program during the experiment. According to the built models (Fig. 5, 6), the software component of the robots' hardware is proposed to be implemented in the form of clients executing address commands from the server. Under this approach, it is not necessary, when changing a task, to change the firmware of each robot separately. Control is executed centrally from the server. The control system provided the possibility to check the operation of the virtual simulation system and control over robots in real time. In this case, to test the system's limits regarding control time, the business logic support server was located in Singapore and the user interface server was located in the United States. At the same time, the two users were also territorially removed from each other and from the actual models of robots at a distance of about 30 km. In this case, users and robots worked in the networks of different providers. The measurements during the experiment showed a delay between the control signal and the robot's response equal to about 0.9–1.2 s. When connecting users and clients to the same network using a local server, the latency of the system decreased to 6 ms.

Video broadcast of robots' movement for users was implemented in the first stage by Zoom Cloud Meetings. The results are fully satisfying to laboratory needs and are suitable for use in the further development of the system to study machine learning algorithms. When moving to building actual systems or increasing the number of clients above 200, the model needs to be modified. It is necessary to move to a multilevel architecture using additional protocols such as progressive realizations of HTTP and MQTT. It is also worth foreseeing a location in the model for an additional local server. Its functions would include loading customer management subprograms, collecting data, and sending statistics to a remote server.

Further development of this study is as follows:

1) to develop improved mobile robots;

2) to improve the functionality of the robots and implement full-fledged feedback to their server for data acquisition from sensors;

3) to implement an image recognition system to analyze the work field and determine the coordinates of each robot;

4) to design a conceptual user interface with a simple configurator of robot behavior;

5) to improve the network model of robots for the ability to control a significant number of robots at a minimal time delay.

8. Conclusions

1. Based on the results of search and approbation of existing data transmission technologies, it has been found out that for the initial stage of the experiment it is advisable to develop a model of a multi-agent system of mobile robots based on the WebSocket protocol. Among the available technologies, this model has the greatest performance speed, loads the CPU of the client and server the least, and minimizes the cost of Internet traffic.

2. We have determined the components of the model of the multi-agent system of mobile robots, their role, and relationships. Based on the selected WebSocket protocol and the structure of the built model, a basic stack of technologies has been defined for the software implementation of the study results. The demonstrated structures (Fig. 5, 6) correspond to the reference network model OSI (The Open Systems Interconnection model) and include the distribution of tasks to the level of user interface and the level of support for business logic. This has made it possible to manage the prototypes of robot clients in real time, to model and investigate the execution of machine learning algorithms.

3. The hardware and software implementation of the developed theoretical provisions have been carried out. The JavaScript library React with a SPA (Single Page Application) technology was used in this work to implement Client Side Render. Virtual DOM (Document Object Model) technology stored in the device's RAM and synchronized with the real DOM was applied to optimize the performance of the system.

The results of our study are suitable for building laboratory models and provide an opportunity to ensure compliance with educational and research goals.

References

1. Stepanov, P. P. (2019). Application of group control and machine learning algorithms on the example of the "Battlecode" game. Cybernetics and programming, 1, 75–82. doi: https://doi.org/10.25136/2306-4196.2019.1.23527
2. Yang, E., Gu, D. (2004). Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey. Available at: https://www.researchgate.net/profile/Dongbing_Gu/publication/2948830_Multiagent_Reinforcement_Learning_for_Multi-Robot_Systems_A_Survey/links/53f5ac820cf2fceacc6f4f1a.pdf
3. Elhajj, I. H., Goradia, A., Xi, N., Kit, C. M., Liu, Y. H., Fukuda, T. (2003). Design and analysis of internet-based tele-coordinated multi-robot systems. Autonomous Robots, 15, 237–254. doi: http://doi.org/10.1023/A:1026266703684
4. Cao, Y. U., Fukunaga, A. S., Kahng, A. B. (1997). Cooperative mobile robotics: antecedents and directions. Autonomous Robots, 4, 7–27. doi: http://doi.org/10.1023/A:1008855018923
5. Van der Zwaan, S., Moreira, J. A. A., Lima, P. U. (2000). Cooperative learning and planning for multiple robots. Proceedings of the 2000 IEEE International Symposium on Intelligent Control. Held Jointly with the 8th IEEE Mediterranean Conference on Control and Automation (Cat. No.00CH37147). doi: https://doi.org/10.1109/isic.2000.882949
6. Asada, M., Uchibe, E., Hosoda, K. (1999). Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. Artificial Intelligence, 110 (2), 275–292. doi: https://doi.org/10.1016/s0004-3702(99)00026-0
7. Touzet, C. F. (2000). Robot awareness in cooperative mobile robot learning. Autonomous Robots, 8, 87–97. doi: https://doi.org/10.1023/A:1008945119734
8. Mataric, M. J. (1997). Reinforcement learning in the multi-robot domain. Autonomous Robots, 4, 73–83. doi: https://doi.org/10.1023/A:1008819414322
9. Michaud, F., Mataric, M. J. (1998). Learning from history for behavior-based mobile robots in non-stationary conditions. Machine Learning, 31, 141–167. doi: https://doi.org/10.1023/A:1007496725428
10. Fernandez, F., Parker, L. E. (2001). Learning in large cooperative multi-robot domains. International Journal of Robotics and Automation, 16 (4), 217–226.
11. Bowling, M., Veloso, M. (2003). Simultaneous adversarial multi-robot learning. IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence, 699–704. Available at: http://www.cs.cmu.edu/~mmv/papers/03ijcai-grawolf.pdf
12. Liu, J., Wu, J. (2001). Multiagent Robotic Systems. CRC Press, 328. doi: https://doi.org/10.1201/9781315220406
13. Matarić, M. J. (2001). Learning in behavior-based multi-robot systems: policies, models, and other agents. Cognitive Systems Research, 2 (1), 81–93. doi: https://doi.org/10.1016/s1389-0417(01)00017-1
14. Srinivasan, L., Scharnagl, J., Schilling, K. (2013). Analysis of WebSockets as the New Age Protocol for Remote Robot Teleoperation. IFAC Proceedings Volumes, 46 (29), 83–88. doi: https://doi.org/10.3182/20131111-3-kr-2043.00032
15. Introducing Hooks (2020). React. Available at: https://en.reactjs.org/docs/hooks-intro.html
16. Diduk, V. A., Savchenko, B. S. (2020). Robototekhnichna systema z viddalenym keruvanniam. Vseukrainska naukovo-praktychna Internet-konferentsiya "Avtomatyzatsiya ta kompiuterno-intehrovani tekhnolohiyi u vyrobnytstvi ta osviti: stan, dosiahnennia, perspektyvy rozvytku". Cherkasy, 46–49. Available at: https://conference.ikto.net/pub/akit_2020_16-22march.pdf