https://doi.org/10.31891/csit-2024-2-2
UDC 621.391 160164

Andrii NICHEPORUK, Oleksandr DARIYCHUK, Serhii DANCHUK
Khmelnytskyi National University

MODEL OF PROCESS FOR ENSURING FAULT TOLERANCE IN INTERNET OF THINGS NETWORKS

The Internet of Things is a concept that describes a network of physical objects equipped with embedded technologies, allowing them to collect and exchange data over the Internet. The main idea is to connect various devices and objects around us so that they can collaborate and interact with each other without direct human intervention. However, this carries certain risks: failures in such systems can have serious consequences, including potentially fatal events. Therefore, the reliability of IoT systems becomes critical in many areas, especially where safety is a priority. The problems hindering the resolution of this issue are primarily related to the heterogeneity of the IoT environment, the lack of communication about failures and malfunctions between network elements, and the heterogeneous environment. As a result, the time to detect errors in such networks is quite long. Thus, the aim of this work is to model the process of ensuring fault tolerance to reduce the time to detect malfunctions in IoT networks by designing and implementing a fault tolerance system in IoT networks.

The paper presents a model for providing fault tolerance in the Internet of Things network, describes key concepts, entities and connections, and also defines the main stages and processes that are included in providing fault tolerance. This model is the basis of the functioning of the fault tolerance system. The concept of a fault tolerance system that integrates into existing Internet of Things networks is proposed. The concept of fault tolerance agents is introduced, which make up the basis of the fault tolerance system, and which communicate with each other to ensure the exchange of information about the occurrence of a fault. Two local fault tolerance mechanisms are proposed, which determine the functionality of the agents. To verify the effectiveness of error detection, experimental studies were conducted, including two error detection scenarios using two local fault tolerance mechanisms.

Keywords: faults, Internet of things, event exchange

Андрій НІЧЕПОРУК, Олександр ДАРІЙЧУК, Сергій ДАНЧУК
Хмельницький національний університет

МОДЕЛЬ ПРОЦЕСУ ЗАБЕЗПЕЧЕННЯ ВІДМОВОСТІЙКОСТІ В МЕРЕЖАХ ІНТЕРНЕТУ РЕЧЕЙ

Інтернет речей — це концепція, яка описує мережу фізичних об'єктів, оснащених вбудованими технологіями, які дозволяють їм збирати та обмінюватися даними через Інтернет. Основна ідея полягає в тому, щоб з'єднати різні пристрої та об'єкти, які нас оточують, щоб вони могли співпрацювати та взаємодіяти один з одним без прямого втручання людини. Однак це несе в собі певний ризик: збої в таких системах можуть мати серйозні наслідки, включно з можливими фатальними подіями. Таким чином, надійність систем Інтернету речей стає критичною в багатьох сферах, особливо в тих, де безпека є пріоритетом. Проблеми, які перешкоджає вирішенню цієї задачі, в першу чергу пов'язані із неоднорідністю середовища Інтернету речей, відсутністю комунікації про збої та несправності між елементами мережі, а також гетерогенність середовища. Як наслідок, час виявлення помилок у таких мережах є досить великий. Тому метою роботи є моделювання процесу забезпечення відмовостійкості з метою скорочення часу виявлення несправностей у мережах Інтернету речей шляхом проектування та впровадження системи забезпечення відмовостійкості в мережі Інтернет речей.

У роботі представлено модель забезпечення багаторівневої відмовостійкості у мережі Інтернету речей, яка описує ключові концепції, сутності та зв'язки, а також визначає основні етапи і процеси, які включаються у забезпечення відмовостійкості. Дана модель є основою функціонування системи забезпечення відмовостійкості. Запропоновано концепцію системи забезпечення відмовостійкості, яка інтегрується у існуючі мережі Інтернету речей. Введено поняття агентів забезпечення відмовостійкості, що складають основу системи забезпечення відмовостійкості, і які комунікують між собою для забезпечення обміну інформації про виникнення помилки. Запропоновано два локальні механізми забезпечення відмовостійкості, які визначають функціональність агентів. Для перевірки ефективності виявлення помилок було проведено експериментальні дослідження, які включали два сценарії виявлення помилок із використанням двох локальних механізмів забезпечення відмовостійкості.

Ключові слова: несправність, Інтернет речей, подія.

Introduction

The Internet of Things (IoT) is a concept that describes a network of physical objects equipped with embedded technologies that enable them to collect and exchange data over the Internet. The main idea is to connect various devices and objects around us so that they can cooperate and interact with each other without direct human intervention [1-3]. IoT devices can vary widely, from home smart devices such as thermostats, sockets, lights, and vacuum cleaners, to industrial equipment and transportation systems. They can monitor the environment, control industrial processes, track inventory, send data for analysis, and much more. The Internet of Things enables these smart devices to interact with each other and with other Internet-connected devices, creating a vast network of interconnected devices capable of exchanging data and performing various tasks autonomously. However, this comes with a certain risk: failure in such systems can have serious consequences, jeopardizing their reliability and public trust. Therefore, the reliability of IoT systems has become critical in many cases, especially in highly responsible areas.

Although considerable progress has been made in this direction, this issue remains relevant. The general scheme of ensuring fault tolerance in classical fault-tolerant systems is presented in fig. 1. As can be seen from the diagram, the IoT network is a multilevel architecture, where each level performs its functions. Typically, fault tolerance solutions are developed for individual levels, allowing failures to be detected at a particular level. However, other levels do not have information about errors occurring at higher levels, leading to difficulties in quickly detecting and effectively managing failures at higher levels. This can result in failures spreading throughout the system, affecting its overall reliability and productivity [4]. Therefore, it is important to develop and implement comprehensive fault tolerance solutions that work at all levels of the IoT infrastructure to facilitate quick detection, isolation, and management of failures across the network.

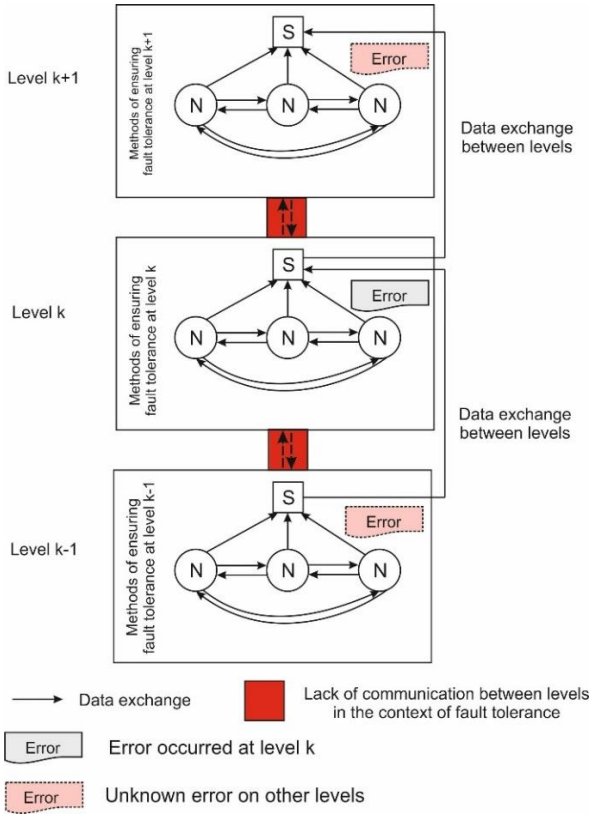


Fig. 1. The general scheme of ensuring fault tolerance of classical fault tolerance systems

To address this problem, a number of solutions have been proposed in the literature. However, existing implementations of fault tolerance mechanisms in IoT systems focus on specific issues: they are designed for specific architectures of IoT systems, they cannot scale beyond small-scale solutions, they provide solutions for specific failures, such as communication channel failures or device failures, and often focus on the same level [5-7]. Additionally, different levels in IoT systems work together, so data and information generated at one level are typically distributed across different levels for further transmission or processing [8, 9]. Therefore, errors at one level can propagate, causing failures at other levels and potentially leading to system-wide failure. Therefore, developing new approaches to ensure fault tolerance in IoT networks is a relevant task. To address this task, a fault tolerance system in IoT networks has been proposed.

Model of process for ensuring fault tolerance in internet of things networks

Developing a model of process for ensuring fault tolerance in an Internet of Things (IoT) network is an important stage in designing the fault tolerance system itself. It allows describing how the system should operate, its essence and relationships, and also determining the main stages and processes involved in ensuring fault tolerance. This model helps identify potential problems and risks that may arise in the IoT network and develop strategies to prevent or minimize them.

To describe the model of the process of ensuring multi-level fault tolerance in an Internet of Things network, a set-theoretic approach will be used.

This model consists of various elements and relationships between them, which together form a system of interaction and functioning aimed at achieving the goal of ensuring a specified level of fault tolerance. The elements of this model are network components, fault response processes, monitoring and control systems, as well as mechanisms for preventing fault propagation. The relationships between these elements determine the ways in which they interact and influence each other in order to optimize the response to faults and ensure the reliability of

the IoT network. The model of the process below is generalized and represents the functioning of the fault tolerance system.

Let's formulate the goal of ensuring fault tolerance as the function f_g of minimizing the time for detecting and correcting faults in network components as follows:

$$f_g: E \times C \rightarrow \min(t_d, t_m), \tag{1}$$

where t_d – time to detect faults in the network; t_m – time to mitigate faults in the network; E – set of faults that may occur in network components *C*.

Let's represent the IoT network model as a set of hierarchical levels:

$$MIOT = \{L\}_{i=1}^{k}, \tag{2}$$

where l_i – network level.

Each level in this MIOT model is represented by a tuple:

$$l_i = \langle N, L, R, P \rangle, \tag{3}$$

where N – nodes, which can be various devices and sensors capable of connecting to each other and exchanging data.

L – links represent connections between nodes, which can be wired or wireless connections, such as Bluetooth, Wi-Fi, Zigbee, etc; R – routes define the paths through which data is forwarded between nodes in the network. This may include routing tables and routing algorithms; P – protocols define the rules and procedures used for communication and data exchange in the network.

Then data transmission in this MIOT network can be defined by the function f_p:

$$f_p = d_{l_{i-1}} \rightarrow d_{l_i} \rightarrow d_{l_{i+1}}, \tag{4}$$

where d_{l_i} – data from level l_i.

Taking into account the function f_p, which defines data transmission in the $MIOT$ network, let's formulate a function that describes possible faults that occur in this process. Since faults can be caused by various factors such as packet loss, data transmission delays, or node failures, we can define a function f_e for this:

$$f_e = e_{l_{i-1}} \vee e_{l_i} \vee e_{l_{i+1}}, \tag{5}$$

where e_{l_i} – fault event that occurred at level l_i.

The logical OR operation is used to denote the logical disjunction operation, which means that the result will be true if at least one of the operands is true. In the context of the function f_e, which describes possible faults in data transmission in the $MIOT$ network, the operands e_{l_i} represent possible fault events on the previous, current, and next data transmission chains, respectively. Using the disjunction operation allows considering any of these fault events as a reason for the occurrence of a common fault in the data transmission system. Thus, if any of these operands becomes true (i.e., if any fault event occurs on any data transmission chain), the result of f_e will be true, indicating the presence of a fault in the system.

Thus, if a fault occurs at level $e_{l_{i-1}}$, level l_{i+1} "is unaware" of this fault and may continue to generate requests to devices at level l_{i-1}. Consequently, the total request time will be the sum of the time taken to traverse all levels, as the levels are unaware of the fault, and thus, the fault will propagate further through the network.

Therefore, designing a fault tolerance system involves reducing the time to detect faults and reducing the propagation path of faults in the network, which corresponds to the goal of ensuring fault tolerance formulated in expression (1).

Thus, to ensure fault tolerance in the Internet of Things network, the involvement of a fault tolerance system (FTS) is proposed.

For this purpose, let's introduce the concept of agents of the fault tolerance system, which will operate at each level l_i, represented as a tuple of functions:

$$A_{l_i} = \langle a_m, a_r, a_e, a_c \rangle, \tag{6}$$

where a_m the node status monitoring function determines the ability of the FTS agent to continuously monitor its nodes at level l_i, detecting any anomalies or faults. This may include checking the availability of sensors and devices, processor load level, response time, and other parameters;

a_r the recovery management function determines the ability of the agent to detect failures or anomalies, and the fault tolerance system agent may initiate the recovery process, such as device reboot or network reconnection;

a_e - the message exchange function determines the ability of the FTS agent to exchange messages with nodes via an MQTT broker to notify about detected faults, transmit recovery status, or send recovery commands;

a_c the action coordination function determines the ability of the agent to coordinate actions with other FTS agents on other nodes in case of failure detection to prevent fault propagation and ensure system stability;

Thus, the operation of FTS agents at each network level involves continuous monitoring of the states of nodes at this level, detection of failures and anomalies, as well as coordination of actions to ensure stability and system recovery in case of problems.

Structurally, each fault tolerance system agent a_{l_i} is defined by the following components:

$$a = \langle InputChanel, EventProc, EventDB, OutputChanel \rangle, \tag{7}$$

where I – input chanel, E – event processing, S – saving event, O – output chanel.

Then we will rewrite the Internet of Things network model (2) taking into account the set of agents:

$$MIOT = \{L, b, A\}_{i=1}^{k}, \tag{8}$$

where $A = \{a_{l_i}\}_{i=1}^{N_l}$ is a set of FTS agent, where N_l – the number of layers of the Internet of Things network; b is an event bus that implements the exchange of messages between agents a_{l_i}.

Let's consider the operation of the fault tolerance system through the functioning of the FTS agents. The functioning of the FTS agent can be represented by the set of its local fault tolerance mechanisms FTm:

$$FTm = \{ft_i\}_{i=1}^{k}, \tag{9}$$

where ft_i is a i-th local fault tolerance mechanisms.

Local fault tolerance mechanisms are strategies or procedures embedded directly into the fault tolerance system (FTS) agent. They are designed to ensure the reliability and stability of the agent's functioning in the face of possible failures or attacks.

It's worth noting that the number of mechanisms may vary for each network level (and thus for each FTS agent). Additionally, this model accommodates extension points, allowing for the addition of new local fault tolerance mechanisms. In this model, the functioning of the FTS agent is represented by two local fault tolerance mechanisms:

$$FTm = \{ft_1, ft_2\}, \tag{10}$$

where $ft_1$is propagation constraints, ft_2 is active monitoring.

The local fault tolerance mechanism implementing propagation constraints is based on intercepting and processing requests from devices addressing an unavailable resource. Let's present the functioning of this mechanism as a chain of functions:

$$ft_1 = \{f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4 \rightarrow f_5 \rightarrow f_6\}, \tag{11}$$

where f_1 is detection of resource unavailability: the fault tolerance agent is notified about the unavailability of the requested resource.

f_2 is interception of new requests: After detecting the unavailability of the resource, all new requests addressed to this device are intercepted;

f_3 is declaration of resource unavailability: Intercepted requests are returned, declaring the device unavailable;

f_4 is activation of verification mechanism: The device verification mechanism is triggered using retries;

f_5 is device verification: The fault tolerance agent periodically checks if the device, which has restored functionality, is working;

f_6 is completion of verification: If it is determined that the resource is operational, the verification process is terminated, and access to the resource is no longer intercepted by the cloud intermediary software.

The local fault tolerance mechanism implementing active monitoring is based on continuously checking the activity status of devices and network levels. Each device on level periodically sends heartbeat signals to confirm their operability. This information is exchanged between devices and levels using the gossip algorithm, which allows for the quick and efficient detection of faults.

The "gossip" algorithm is a method for disseminating information in distributed systems, where each node randomly exchanges information with other nodes. This algorithm is commonly used for data replication, fault

detection, or event dissemination. The main idea is that each node randomly selects another node and transmits a certain amount of information or message to it. Upon receiving the information, the node checks whether it already has this information and updates its state if necessary.

Regarding the local fault tolerance mechanism implementing active monitoring, according to the gossip algorithm, if a device or level fails to send a heartbeat signal within a certain period, they are considered faulty, and this information is propagated throughout the network. This approach allows the system to react immediately to faults and ensure uninterrupted operation in conditions where a specific resource is unavailable.

Let's present the functioning of the active monitoring mechanism as a chain of functions:

$$ft_1 = \{f_1 \to f_2 \to f_3 \to f_4 \to f_5\}, \quad (12)$$

where f_1 is continuous monitoring at all levels: each level of the system and devices periodically send heartbeat signals to confirm their activity and functionality;

f_2 is fault detection using the gossip algorithm: each network level is informed about the activity status of other devices and levels using the gossip algorithm. If a device or level fails to send a heartbeat signal within a certain period, it is considered faulty;

f_3 is dissemination of information about faults: information about faulty devices is propagated throughout the network so that other levels and devices can cease using unavailable resources;

f_4 is interception of requests to unavailable devices: when the cloud middleware receives a request for an unavailable device, it intercepts it and returns it, indicating the resource's unavailability;

f_5 is restoration of operation after device recovery: if the device returns to an active state, the interception of requests is stopped, and access to the resource is restored.

Thus, the developed model of process for ensuring fault tolerance in internet of things networks describes key concepts, entities, and relationships, as well as defines the main stages and processes involved in ensuring fault tolerance. This model forms the basis for the functioning of the fault tolerance assurance system.

Experiments

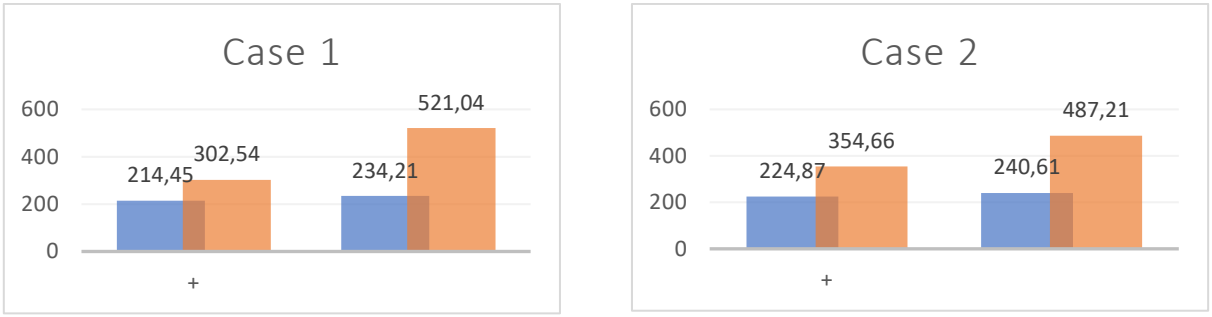
For experimental evaluation of the proposed fault tolerance system, a testbed IoT network system was modeled to include monitoring devices of a “Smart Home” such as temperature, pressure, and humidity sensors. The simulated network consisted of four levels: perception level, communication level, services level, and applications level, simulating the operation of devices (with sensors), gateway, cloud provider, and a mobile application acting as a user interface point. The IoT network was modeled and integrated with the fault tolerance system [10]. Raspberry Pi 3 single-board computer systems were used for the physical nodes connected to the network. Thus, for each level of the IoT network, Raspberry Pi board were used, each with functions corresponding to the respective level (in total, 3 units). It is worth noting that a Mosquitto broker was deployed on the cloud provider node to organize the overall message bus of the fault tolerance system. Software for the operation of the fault tolerance system was implemented in Python.

Experimental investigations were conducted, which included two error detection scenarios using two local fault tolerance mechanisms (equation 10), i.e., the time elapsed from the moment the fault (or error) occurred to the moment it was detected. This metric indicates how quickly the system or device can detect a problem after it occurs. The smaller the fault detection time, the faster one can react to it and take measures to restore normal system operation [11]. Calculation of this metric was performed according to the formula:

$$t_d = t_{dstart} - t_{dend} \quad (13)$$

where t_{dstart} – start time of fault, t_{dend} – fault observation time.

In case 1, a local fault tolerance mechanism ft_1 was applied, which functioned by intercepting and returning all new requests addressed to the faulty device when the cloud intermediary software notified that the requested resource was unavailable, declaring it inaccessible. In scenario 2, another local fault tolerance mechanism ft_2 was used – active monitoring. Unlike case 1, where faults are detected only when a resource is requested, in case 2, continuous monitoring is performed to obtain error-related information at all levels. The gossip algorithm was applied, involving monitoring each level and devices that report whether they are active within a certain period. For each scenario, error data were modeled, differing in the time of introduction into the system and duration. The experiment results are shown in fig. 3. The blue bars in the chart represent the fault detection time with the fault tolerance mechanism, while the red bars represent it without it. The first part of each graph shows the local propagation restriction mechanism ft_1, and the second part shows the active monitoring ft_2. In both scenarios, errors were simulated in the temperature and humidity sensor DHT11.



а) **б)**

Fig. 3. Results of experiments: a) case 1; b) case 2

The longest time to detect an error was recorded in the first scenario, where the fault tolerance assurance system was absent, at 521.04. The difference in case 1 between the time to detect errors with the proposed fault tolerance assurance system and without it was 8%, while in case 2, it was 53.42%. In case 2, the error detection strategy involved spreading information about faults between levels.

Conclusion

Thus, the developed model of process for ensuring fault tolerance in internet of things networks describes key concepts, entities, and relationships, as well as defines the main stages and processes involved in ensuring fault tolerance. This model serves as the foundation for the operation of the fault tolerance assurance system. A concept of fault tolerance assurance system is proposed, which integrates into existing IoT networks. The definition of fault tolerance assurance agents is introduced, forming the basis of the fault tolerance assurance system, and they communicate with each other to exchange information about error occurrences. Two local fault tolerance assurance mechanisms are proposed, defining the functionality of the agents. Experimental studies were conducted to assess the effectiveness of error detection using two scenarios with different local fault tolerance assurance mechanisms. The results of the experiments indicate that the application of the designed fault tolerance assurance system led to a reduction in the time to detect errors in IoT networks. In scenarios where the fault tolerance assurance system was used, a significant decrease in the interval between error occurrence and detection was observed.

References

1. T. K. Gannavaram, U. M. Kandhikonda, R. Bejgam, S. B. Keshipeddi and S. Sunkari, A Brief Review on Internet of Things (IoT), 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2021, pp. 1-6, doi: 10.1109/ICCCI50826.2021.9451163.

2. H.-N. Dai, Z. Zheng and Y. Zhang, Blockchain for Internet of Things: A Survey, IEEE Internet of Things Journal, vol. 6, no. 5, 2019, pp. 8076-8094.

3. P. C. van Oorschot and S. W. Smith, The Internet of Things: Security Challenges, IEEE Security & Privacy, vol. 17, no. 5, 2019, pp. 7-9.

4. M. Stetsiuk, A. Kashtalian The methods of ensuring fault tolerance, survivability and protection of information of specialized information technologies under the influence of malicious software, 2022, Computer Systems and Information Technologies, vol. 1, pp. 36–44. doi:10.31891/CSIT-2022-1-5

5. A. Agrawal, D. Toshniwal Fault Tolerance in IoT: Techniques and Comparative Study, Asian Journal of Convergence in Technology, Vol.7, Issue 2, 2021, pp.46-52

6. P. Vedavalli and C. Deepak, Enhancing Reliability and Fault Tolerance in IoT, 2020 International Conference on Artificial Intelligence and Signal Processing (AISP), Amaravati, India, 2020, pp. 1-6, doi: 10.1109/AISP48273.2020.9073174.

7. M. Davoodi, K. Khorasani, H. Talebi, H. Momeni Distributed Fault Detection and Isolation Filter Design for a Network of Heterogeneous Multi-Agent Systems. IEEE Trans. Control. Syst. Technol. 2014, vol. 22, pp. 1061-1069.

8. M. Melo, G. Aquino Multi-level Fault Tolerance Approach for IoT Systems, Computational Science and Its Applications – ICCSA 2021, 2021 pp 421–436

9. A. Nugraha Tama, H. Kusuma Wardana and S. Nugroho, Gossip Algorithm Implementation for Network Protocol, 2018 International Seminar on Application for Technology of Information and Communication, Semarang, Indonesia, 2018, pp. 299-303, doi: 10.1109/ISEMANTIC.2018.8549774.

10. O. Makieiev, N. Kravets Study of methods of creating service-oriented software systems in azure. Computer Systems and Information Technologies, 2023, vol. 2, pp. 38–47. doi:10.31891/csit-2023-2-5

11. V. Khoroshko, V. Kudinov, M. Kapustian Evaluation of quality indicators of functioning cyber protection management systems of information systems, Computer Systems and Information Technologies, 2022, vol. 2, pp. 47–56. doi: 10.31891/csit-2022-2-6

Andrii NICHEPORUK **Андрій НІЧЕПОРУК**	PhD, Associate Professor of Computer Engineering & Information Systems Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine, e-mail: andrey.nicheporuk.@gmail.com https://orcid.org/0000-0002-7230-9475 Scopus Author ID: 56239856200, ResearcherID: R-9498-2017	кандидат технічних наук, доцент, доцент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет
Oleksandr DARIYCHUK **Олександр ДАРІЙЧУК**	master's degree student, Khmelnytskyi National University, Khmelnytskyi, Ukraine, e-mail: dariychuk@gmail.com	Магістрант кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет
Serhii DANCHUK **Сергій ДАНЧУК**	PhD student, Khmelnytskyi National University, Khmelnytskyi, Ukraine, e-mail: sergey.danchuk.p@gmail.com https://orcid.org/0009-0003-4510-0363	аспірант, Хмельницький національний університет