УДК 681.3 https://doi.org/10.15407/pp2026.02.058

Д.В. Рагозін, В.Є. Смірнов

SIMULATION AND ANALYSIS OF PEER-TO-PEER ROBOT SWARM NETWORK

In the paper we describe the full cycle development of TDMA-based communication protocol for a swarm of robots, including simulation and hard-ware implementation. The protocol is targeted to have a robust, interference-immune transport in the network. First, we developed a simulator, based on SimPy simulation package, which helps us to run thousands of simulations for proving the concept of TDMA-based communications for robotic swarm. Second, using the simulator we developed a bunch of techniques for the TDMA transport to improve network robustness and simulations allowed to gather statistics and choose the better algorithm. Third, we employed so-called AI tools to implement parts of simulator and a helper technique to convert simulation code to embedded code, approaching "digital twin" paradigm. Finally, the simulated protocols are successfully ported to hardware, which supports LoRa protocol, but not limited to LoRa physical layer. The resulting embedded code works accordingly to simulation results and gathered statistics. The developed simulation environment and modern so-called AI tools allowed to shorten dramatically the embedded software development cycle and evaluate algorithm efficiency information from the simulation results before applying on real hardware.
Keywords: swarm simulation, wireless network, digital twin, TDMA-based communication

D.V. Rahozin, V.Ye. Smirnov

МОДЕЛЮВАННЯ І АНАЛІЗ РІВНОПРАВНИХ РОЇВ РОБОТІВ

У статті розглядається повний цикл розробки комунікаційного протоколу на основі TDMA для рою роботів, включно з моделюванням та апаратною реалізацією. Протокол розроблений для забезпечення надійного, захищеного від перешкод передавання даних у мережі. 1) Було розроблено симулятор на основі пакета моделювання SimPy, який дозволяє проводити тисячі симуляцій для підтвердження концепції комунікацій на основі TDMA для рою роботів. 2) За допомогою симулятора розроблено низку методів для передавання даних в рамках фреймів TDMA з метою покращення стійкості мережі. Ці симуляції дозволили зібрати статистику, проаналізувати та вибрати кращий алгоритм. 3) Використано інструменти так званого генеративного штучного інтелекту для створення частин симулятора та додаткові техніки для перетворення коду моделювання у вбудований код з метою наближення до парадигми «цифрового двійника». 4) Змодельовані протоколи успішно перенесені на обладнання, яке підтримує протокол LoRa, але не обмежується фізичним рівнем LoRa. Отриманий вбудований код працює відповідно до результатів моделювання та зібраної статистики на основі моделі. Розроблене середовище моделювання та сучасні інструменти штучного інтелекту дозволили значно скоротити цикл розробки вбудованого програмного забезпечення та оцінити інформацію про ефективність алгоритму з результатів моделювання перед застосуванням на реальному обладнанні.
Ключові слова: моделювання рою, бездротова мережа, цифровий двійник, зв'язок на основі технології TDMA

1. Introduction

Today the swarm of robots or drones is an in-demand technology for field application, as it provides control of multiple robotic devices using only one control center or even allow them to act autonomously following some scenario. For various industrial purposes the basic scenario, when an operator controls only one drone, looks obsolete, as this case requires a dedicated operator for each active drone. The multiple drones use may improve the execution efficiency of many process types, but it requires: 1) big enough number of operators; 2) dedicated control channels which should be separated one from another; 3) maybe the strongest issue - synchronization and controlling operators to execute meaningful tasks over some area without interfering one another. Anyway, the human operator use now is the simplest and the cheapest solution for many cases, as the operator job can be done remotely for

low price. But this benefit cannot be projected for near future. The main goal of our research is to discover the possibilities of building semi-autonomous robot swarms, which act over defined area in peer-to-peer network and may exchange roles under changing environmental conditions and under limitation of robotic resources until the mission is completed. One of the main limitations is the definition of communication protocol, which further should support efficient swarm control protocol.

There are many aspects that restrict the data exchange paradigm in a robotic network, but the first step is the building of inter-robotics communication. We are not going to write down a long list or a large classification of factors that affect the network protocol, but we highlight the aspects most important for our case. One aspect is the limited bandwidth, at least for the autonomous drone scenario usually. We have no requirement for continuous video streaming delivered for human operator, so usually the geographical coordinates, velocity and RSSI are required data to exchange. Another aspect is the maintaining robot network integrity in case of obstacles, jamming and noise in communication channel. The data bandwidth aspect should be considered for establishing tradeoff between being narrow enough against the increased number of drones in the swarm, up to hundreds. All the aspects restrict the tradeoffs in definition of efficient control protocol, which allows the robotic network to reach mission goals with or without operator control. Possibly, some attention should be paid for limiting power consumption – to extend the life of battery-operated drones in "suspend mode" or temporary inactive robot mode. This enables long missions; even weekly mission becomes possible. The proper communication protocol gives a good basis for building a network - with wide variety of underlaying physical and transport layers. Robust protocol is a good base for implementing particular algorithms which automatize robot mission planning and execution.

In chapter 2 we are defining the task and goals for the robotic network concept and discussing its most important properties. In chapter 3 we researching the possible ways to build a model, and describe the model of communication protocol. In chapter 4 we describe simulation results, including gathered metrics for model efficiency and corresponding hardware implementation.

2. Robotic Network Concept

2.1. Protocol concept

Our goal is to define the communication model of robotic network with respect to modern hardware we use further for its implementation. The model is used to evaluate important metrics: 1) swarm recovery time during mission execution in case of jamming/bad link cases; 2) drone swarm reaction time for rebuilding a swarm control software for a new mission; 3) minimum required bandwidth for tasks; 4) much simpler characteristics of the swarm behavior in case of different physical radio transceivers and power supply characteristics. On the next step the developed model allows to evaluate high-level swarm mission scenarios.

It should be noted, that our research echoes the method applicable for ad-hoc networks [1], however the modern robots have less limitations – more energy, less operating time, less restrictions for transmission channels and speeds, less number of operating devices. Still, the robots can move across the network area, compared to practically non-moving sensor. And the main difference – if the earlier sensors devices have quite simple onboard sensors, modern robots have RGB and thermal cameras, radars, lidars, ultrasonic sensors and many other devices. Such device set give the overwhelming information about environment, but still the scenarios of the robot swarm use are quite basic, usually restricting useful scenarios for basic operations, for example, in agricultural sector. We are hopeful that introducing even the basic swarm usage scenarios into industry will help to employ more and more use cases over near time.

For the definition of the protocol concept, we are reviewing several most often used scenarios for the swarm: 1) surveillance scenario, where the swarm is constantly looking for some anomalies over an area; 2) continuous execution of basic tasks for robots – e. g. spraying some agriculture; 3) delivering packages over routes – in case of emergency situation – with possible mission changes "on-the-fly".

All enlisted tasks require communication protocol for swarm orchestration, when all swarm devices share the same quite narrow bandwidth re-source; and a protocol for mission control. For this paper we concentrate on communication protocol: for OSI network layers, we are considering levels 2-3 – Dala Link and Network layers, and partially level 4 – Transport layer. The physical layer for practical evaluation is fixed and the most valuable for today are LoRa standard physical layer.

Practical considerations for scenarios show us, that the communication layer should concentrate at least on the following tasks: 1) providing stable communication between the operator and the swarm; 2) stable communication in case if operator is off-line and the network is autonomous; 3) robustness in case if some nodes fall off-line for short or moderate time; 4) adding new nodes for swarm; 5) joining two separate swarms. Looking back to previously elaborated metrics: coverage, fault tolerance, response time, scalability, throughput [2], robustness, task completion rate [3], precision, success rate, adaptability to scale [4] and so on – we are setting narrower but more complex metrics and goal to simulate, as our scenario set and selected physical link layer mark several metrics as more important. So, we define a set of application-level metrics of our interest, which reflect the components of swarm mission success.

2.2. Physical layer

The most common off-the-shelf communication solution is based on LoRa [5], which looks to be well-known low-power solution for environmental sensors, designed for the long range – up to 15 km distance – applications. On the other side the communication speed looks to be quite low, 0.3 – 50 kBit/sec, but this does not look as a big issue. Earlier we have discussed, that low automation degree of drones requires high bandwidth, as the human operator requires good quality video stream for effective operation, for example current FPV drones` infrastructure is built exactly such way. If we drop video stream, we also drop high bandwidth requirements, moving exact object recognition operations to drone side. This allows to fit communication requirements into strict LoRa band-width, limiting communication to simple exchange of sensor values. The types of sensor samples are: current GPS position, velocity vector, power level, payload weights or value. Basically, the required sensor value types depend on control protocols, which are defined on higher protocol levels, and this requires some iterative process of defining over-the-network control algorithms. We are going to cover several control protocols samples in the next articles, now we concentrate on employing off-the-shelf communication solutions into our swarm.

2.3. Common considerations for communication

Generally, the control protocol does not depend on particular physical layer, but the practical considerations usually point to the cheapest "industry standard" radio transceivers available on market. For the time of writing this article different LoRa devices and modules, working in 433MHz non-licensed range, are the most suitable type of devices available on market. This does not mean that TDMA-based protocol requires LoRa, it can be implemented on any type of radio transmitting devices, where the user can directly control the transmission speed and operating modes of transmitter. For communication protocol planning we should know basic timing delays for switching the transmitter between generally idle mode, receiver mode and transmitter mode. The maximum time necessary for changing transmitter operating mode (e.g. from receive mode to transmission) specifies the time gap between TDMA slots. Other point is the accuracy of local clock, which also affects time gaps between TDMA slots in protocol. The theoretical and practical considerations for time synchronization between connected devices in our network were earlier made in [6], including practical results and hardware implementation. The useful feature is RSSI, which allows to evaluate the received signal strength, so we are able to evaluate the distance between devices and possible device movement.

It should be noted, that robotic communication networks are developed having in mind the target structure of the network and use scenarios. The initial use scenarios for swarm define the complexity of the network

physical layer protocol. Basically, we can separate the network types into the following large groups: 1) permanent configuration, where robots practically are not moved; 2) permanent configuration where robots can move within the limits of their defined areas, but leaving the communications between neighbors practically unchanged – with minor distance change; 3) configuration where robots can freely move across area. The corner cases of the last type is the subdivision of the swarm into several swarm with communication configuration rebuild or joining several swarms into one swarm. All the network types also are challenged the problem of network nodes, that can temporary be offline, so leaving the network for short time and joining it after the leave. The effective solution of corner cases greatly improves the overall performance of the network, and one of goals of our study is the modeling of these corner cases scenario.

For our study we chose the scenario, where the robots located in the geographical center of the swarm can communicate to all the robots in the swarm. For moderate number of robots (80-200) we are employing TDMA-based techniques, where we can effectively divide time resource into the defined number of time slots and give fixed amount of outcoming traffic for each robot per time slot. The first useful work, describing TDMA protocol for mobile devices was described in [7], where multi-hop mobile devices network concept for low-speed communications was described. Also, it was proved [6] that off-the-shelf and low cost quartz resonators can provide time-synchronization for robotic network. These techniques enable to design various types (multi-hop, one-hop) of networks, based on TDMA communication layer techniques which can employ LoRa physical layer.

2.4. LoRa communications considerations

We start from considerations, that in order to simplify TDMA network, we can employ peer-to-peer network concept over a comparatively large geographical area, as maximum distance for LoRa communication is up to 15 km. One of the algorithmic improvements we can employ for this case – the selection of central zone (fig. 1), where the devices can reach all the devices in the network, but other devices possibly cannot communicate directly to peers located too far from them. Fig. 1 shows the robots located on the semi-rectangular field, the central zone devices (black squares) can communicate to all the devices in the network and, at least, one of them has a kind of bridge to Internet network for reporting swarm statistics and receiving control commands. Fig. 1 shows additionally 2 devices, marked with 1 and 2, and border limits – dashed lines, 1-limit and 2-limit, which shows the communication range limits for devices, marked 1 and 2. So, comparing to TDMA-based peer-to-peer communication networks, the central zone controls the TDMA protocol and slots. Other devices, competing for slots in TDMA network, does not "hear" all the devices.

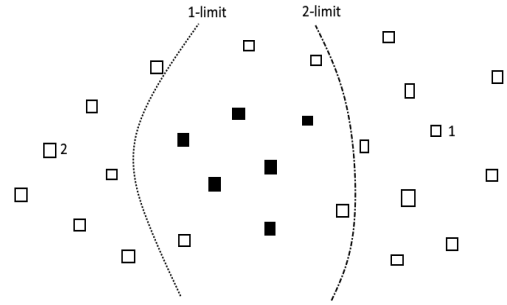

Fig. 1. Two-zones robotic network structure

Consider the TDMA frame structure for IoT devices, shown at fig. 2. The reader may refer to [5] for comparing to simpler implementation.

Slot X	Slot T_0	…	Slot T_K	Slot 0	Slot 1	…
	Slot N-1	Slot N	Slot R_0	…	Slot R_M	

Fig. 2. TDMA frame structure

The total length T_{frame} of the TMDA frame at fig. 2 is fixed, and this time may vary from 0.5 sec to 15 sec, which depends on necessary communication characteristics of the network. T_{frame} shows the typical time of spreading information in the network; for the

network with central zone at fig. 1 this time grows to 2*T_{frame}. For peer-to-peer network this time looks very flexible in terms of control protocol.

The TDMA frame may include: start slot X, which is employed for adding an additional robot or device to the network. Slots T_0-T_K may be used for optimizing of the adding new devices into the network – they extend slot X. Slots 0-*N* are used for communicating between devices, so basically each device owns one slot, optimized scenarios employ multiple slots for one device to increase bandwidth. For our case the value of N starts from 30-40 and finishes near value 200. Some scenarios introduce device priorities, so that high-prioritized devices use several slots for communication. Slots R_0-R_M are used for additional traffic optimizations.

As LoRa transmitters supports RSSI value, which renders approximate distance between receiver and transmitter, the robot approximately can have information about distance between devices, GPS information allows to track device movement and control the so-called density of the devices over a field. This also can be used for specifying communication speed into separate slots and optimizing the traffic inside network. For the simplest applications, only slot X and slots 0-N are used. Slots Ti and Ri can be allocated inside slots 0-N, employing different optimization techniques, which are out of discussion scope now.

3. Swarm simulation concept

3.1. Metrics discussion

The proposed TDMA technique looks basic, but there are several algorithm parts, which are considered quite complex and need to be observed and proven inside simulation. The usual communication procedure looks simple, but the most interesting issues are forming the network and reentering the network after communication signal jams. Communication jams is a short-range or long-range signal obstacles which prohibits communication for the part of the network. In case if the jam is quite long in time, there is a possibility that after such a jam the network need to build the network from scratch. The scenarios of joining two swarms into one or separating one swarm are also targets for our simulation.

3.2. Simulation toolkit

The main goals of the simulation not only for our study are at least 1) to have an environment suitable for observing and analyzing the algorithms; 2) provide a cheap alternative for direct use of hardware; 3) provide a bridge between simulated environment and scenarios – the code in usual case – and the real hardware. Additional point is to have ability to switch between hardware platforms without conceptual coder changes. It should be noted that we are not considering computing hardware of the robotic system, as communication protocol usually requires less than 1% of overall system computing power.

The first common choice for such kind of simulator is NS-3[8], or OM-NeT++ [9]. These simulators and a bunch of other simulators, functionally close to it, are event-driven simulators, which have a rich number of extensions, able to simulate practically every communication protocols. Still the older our work [6] clearly shows, that for our study much lighter tools can be involved. Our final choice was LoRaSim[10] from Lancaster University. Its functionality looks quite basic, but our study showed us that this choice was right: the simulator is lightweight and cheap in term of resources necessary to learn it and use it. As we use LoRa hardware as a tool, we do not need to track much of LoRa protocol internals. All TDMA protocols can be based on transmitters, which are able to receive a packet of defined length and provide a packet of defined length. For robustness we need the ability to change underlaying protocol without redoing the system. So, LoRa simulator [10] is used as an interchangeable component and a physical layer, which can be changed to another physical layer implementation, simply connecting channel layer in OSI model to the physical environment.

3.3. SimPy implementation benefits

The protocol simulation is based on the SimPy [11] – process-based discrete-event simulation framework based on standard Python. This choice is based on its simplicity, as

1) standard Python infrastructure is used; 2) simulation is supported by shared objects and synchronization component libraries in SimPy; 3) the simulation code flow in SimPy is a Python process without specific definitions and infrastructure, which makes the SimPy use also cheap and simple in learning how to use and apply.

The use of common programming language as Python showed us many benefits for our case. First, the simulated process algorithmics, which implement various TDMA-network support parts, is implemented as a well-structured programming code. We even may utilize the concept of "digital twins", directly translating the simulation code into the robotic code. Despite the fact, that Python interpreter makes the Python code slower 10-20x times that C++ code, the modern AI tools – such as commonly used ChatGPT or Gemini tools - allow seamlessly convert Python code to C++ code. The underlying protocol concept employs statically allocated components – data buffers, arrays, variables and so one, as the simulated process is targeted for embedded platforms. The AI tools use allows to overcome the barrier, which was introduced long time ago by using different programming languages for different kind of simulations – as all fast simulations employ C++ optimized code, for example ROS-2 [12]. Its simulation concept is an exact "digital twin" system, versus more easily written but more slow systems in Python, such as SimPy. Now AI tools allow to rewrite still with some limitations the simulation code from one language to another, simplifying and extending the "digital twin" concept use. Sure, that AI tool now cannot deal with optimized driver code for LoRa, but the underlying low level code base is not the simulator part. It is separated from our simulated algorithms with abstraction layer, so we are able to convert simply between languages even using AI tools. For developers, who are still worried with automated code conversions – it is quite possible to get to mature Python-based model, convert it to C/C++, verify differences between original and converted code flow, and in case of success – move further to embedded code.

Another important point for the SimPy use is that the Python-based development and process-based simulator helps to use AI code generation tools efficiently, as it can provide the whole algorithmic base for Python effectively even if the developer of the simulation algorithm is not a proficient Python programmer. Although the "digital twin" concept cannot be used directly for our model, the common code structure for object (robot) behavior is a common control code with time-synchronization primitives and yield instructions, which allows to synchronize the object behavior with model time. During the conversion of the model code into the hardware-side code, only these synchronization primitives need to be changed into hardware-related code. However, the underlying real-time support library can provide the corresponding compatibility layer with SimPy synchronization primitives. All the other control code can be built using AI tools and we have widely practiced to use AI tools while prototyping the simulation code.

4. Simulation details and results

We have implemented our simulation ecosystem on the top of LoRaSim [10] and SimPy [11] software packages, and implemented our TDMA algorithmics using this simulation engine. Our implementation is well aligned with concept used in ROS and ROS-2 [12] simulation, where the robotic control software is build using "digital twin" concept. This concept suppose that the simulation process is identical to control program for the real robot or drone, so the ROS simulator provides all necessary functionality as system libraries, including alternate versions of flight controller.

To be complete with moving the simulation results to the real hardware, Raspberry Pi boards with LoRa transceiver E32-433T30D was used, which is connected by UART link. As expected, real hardware elaborates simulated algorithms as expected.

4.1. Simulation engine

Despite of increased computational complexity - as proper physics simulation for drone engine should be computed during simulation cycle – the "digital twin" approach works well for the modern computing hardware. Our solution provides real time simulation for dozens of drones, including the render-

ing of drone operations using 3D-engine via Unity, and this even emulates video stream for opera-tor reference. We successfully approached the "digital twin" concept, except we have the main loop implemented in Python. The structure of SimPy-based code reflects the structure of real-time control program for drones, so the dropped "digital twin" paradigm component – the same programming language – is covered with AI tools which make translation between Python and C language. We introduce some internal limitations for simulation code – such as static memory allocation and simpler code structure – and current AI tools work well enough during program mapping to C language. We are not going to discuss exact over-head introduced by language translation, this may be a separate investigation mainly in the area of program code quality and code base management.

Due to nature of the model, we have clear layers of the code, which simplify object synchronization in model and coding practices. The bottom layer includes LoRaSim simulator, which is used mainly for physical layer of data transmission, so the practical synchronization of robot's work is done by LoRaSim layer. As discussed in previous chapter, we can change LoRaSim layer to any other wireless simulation layer, as basically we need the following functionality: 1) packet receive; 2) packet transmission; 3) RSSI value as a part of packet receive; 4) setting speed and transmission power values. Any wireless protocol simulator that provides this functionality can be used instead LoRaSim. TDMA frame forming is based on SimPy timer functionality, which can be easily mapped on any hardware. The TDMA frame formation principle is inspired by ideas from old work [6], lowering the complexity of multi-hop system into two-hop network or star-based network. Additionally, LoRaSim is slightly modified to provide tranmission jamming and simulation programmable packet loss.

4.2. Simulation goals and metrics

On the top of TDMA frame formation we simulate a control program, which includes the following main parts:

1. Initial forming of TDMA network, when the robotic swarm should define priorities, example metric – the number of visible devices or RSSI of root node which has link to operator. Operator link is low-speed and transfers basic swarm statistics, goal reaching results and passes operator commands to swarm devices. We are not considering radio transmitter power consumption as it usually less than 1% of overall power consumption on any flying drone. One of modeling goals was to observe different scenarios of network forming, allowing to improve the speed of robot joining the swarm by extending the functionality of slot X (fig. 2). Our model allows to improve speed of network forming 4x-8x times, depending on slot X length and configuration.

2. Link loss correction, which has two cases: short time link loss for several frames, where the robot should not leave the network and save his slot(s) active; and long-time link loss, when the algorithm is similar to initial network forming. This is the most interesting part, as the metric of time, necessary for forming the TDMA-based network, is the main metric for cases, when swarm works around concrete buildings which shield the signal from central drones, so the drone may accomplish some mission in offline, join the network again and send gathered data to robots, which have connection to main network.

The most interesting metric we have analyzed was the time of the network reconstruction, when the network structure is collapsed after a long signal jam. We have applied different optimization techniques for minimization of the number of TMDA frames necessary for network reconstruction, which are not the topic of our paper, and our simulation engine is able to run thousands of network reconstruction scenarios with the results in table 1.

Table 1.

Network reconstruction times for basic and optimized TDMA protocol.

Number of nodes in network	Construction frames for the basic period	Construction frames for the optimized period
5	7	3
12	16	6
20	25	7
40	45	13
80	87	25

So, the numbers in table 1 clearly shows that our simulator allows us to construct and debug algorithms quickly, as the brute force approach to develop some algorithms just on hardware simply not working. Also we are not using any techniques of formal verification due to its complexity, so we need the big number of simulations with randomizing conditions in start and randomizing jamming for each network configuration. Our simulation allows us to have thousands of runs for the network with some constant number of nodes.

4.3. Hardware platform notes

Sure, that the first move of the simulation to hardware platform introduces some problems, related to first improper assumptions on timings for radio transceiver. However, after initial timings correction, the simulation code runs at the platform as we expected. We use frequency band 433MHz, available for radio amateurs, and the limit to 0.5W output power allows to debug the protocol without specific restrictions.

For hard debugging cases we may use a "sniffer" hardware, which is a unit which always receives data from our wireless network, and logs the TDMA frame structure, frame/packets timestamps and data, so it may be compared with simulation log.

It should be noted, that the simulation of protocol and its different optimized version is the only way to make an embedded system in appropriate time. Even if hardware platform Raspberry Pi 4 looks developer friendly, the protocol debugging is a nutshell, as requires elaboration of results and its analysis after each run. Also, it is practically impossible to provide algorithmic debugging via multiple runs, as the communication speed in LoRa protocol makes scenarios run time extremely long. Also, it is quite hard to provide jamming for LoRa in real life.

We are not mention real hardware tests at large distances. Our previous experience with transceivers shows that in real life the transmission distances, signal power and speeds closely resemble the hardware manuals, so the operational range of our transceivers reaches the maximum, described in manuals. Our simulation engine uses RSSI information and introduces random errors due to communication range, so the simulation results are similar to the real-life scenarios.

5. Conclusion

Our research clearly shows that quite simple simulation system can greatly speed up the development of robotic networks with wireless communication abilities. We developed a SimPy-based simulation, which allows to develop and debug TDMA-based protocol for self-organized robotic swarm based on LoRa wireless hardware, and the protocol has advanced abilities to reconstruct network in case of signal jams. The developed algorithms can be comparatively easily transformed, sometimes with helps of current AI tools, into embedded computers, equipped with LoRa hardware, but not limited to LoRa. We have not noticed important differences in functioning of the simulator-based system and its hardware counterpart, so we are close to name these systems as digital twins. The most important is the shortening the development cycle of communication system several times, up to our experience in embedded systems development.

References

1. Agrawal R., Faujdar N., Romero C.A.T., Sharma O., Abdulsahib G.M., Khalaf O.I, Mansoor R.F., Ghoneim O.A. Classification and comparison of ad hoc networks: A review. // Egyptian Informatics Journal, Vol. 24, Issue 1, 2023, p. 1-25, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2022.10.004.
2. Kimon P. Valavanis, George J. Vachtsevanos. Handbook of Unmanned Aerial Vehicles. Springer, 2014. 3022 p. https://doi.org/10.1007/978-90-481-9707-1
3. Mohsan, S.A.H., Othman, N.Q.H., Li, Y. et al. Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends. Intel Serv Robotics 16, 109–137 (2023). https://doi.org/10.1007/s11370-022-00452-4
4. Mueller, M., Smith, N., Ghanem, B. (2016). A Benchmark and Simulator for UAV Tracking. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. LNCS, vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_27

5. Paredes W.D., Kaushal H., Vakilinia I.; Prodanoff Z. LoRa Technology in Flying Ad Hoc Networks: A Survey of Challenges and Open Issues. Sensors 2023, 23, 2403. https://doi.org/10.3390/s23052403
6. Rahozin D. (2008) Modeling synchronised sensor networks // Problems in Programming, special issue, 2008, #2-3. P. 721-729. https://nasplib.isofts.kiev.ua/handle/123456789/2155
7. Kanzaki A., Uemukai T., Hara T., Nishio S.. Dynamic TDMA Slot Assignment in Ad Hoc Networks // In Proc. 17th Int. Conf. on Advanced Information Networking and Applications (AINA'03), 2003. – P. 330–336.
8. Campanile L., Gribaudo M., Iacono M., Marulli F., Mastroianni, M. Computer Network Simulation with ns-3: A Systematic Literature Review. Electronics, 9(2), 2020, p. 272. https://doi.org/10.3390/electronics9020272
9. Virdis A., Kirsche M. Recent Advances in Network Simulation The OMNeT++ Environment and its Ecosystem: The OMNeT++ Environment and its Ecosystem, 2019. ISBN 978-3-030-12841-8. https://doi.org/10.1007/978-3-030-12842-5.
10. Voigt T., Bor M., Roedig U. Alonso, J. Mitigating Inter-Network Interference in LoRa Networks. In EWSN '17 Procs. of the 2017 Int. Conf. on Embedded Wireless Systems and Networks (pp. 323-328). ACM Press. http://dl.acm.org/citation.cfm?id=3105395
11. SimPy homepage, https://simpy.readthedocs.io/en/latest/, last accessed 2025/08/02.
12. Haridevan A., Kang J., Yuan M., Shan J. ROS2-Gazebo Simulator for Drone Applications. In Proc. of 2024 Intl. Conf. on Unmanned Aircraft Systems (ICUAS), p. 1232-1238. https://doi.org/10.1109/ICUAS60882.2024.10556903.

Дата першого надходження до видання: 16.04.2026
Внутрішня рецензія отримана:02.05.2026
Зовнішня рецензія отримана:08.05.2026
Дата рекомендації до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Рагозін Дмитро Васильович,
кандидат технічних наук,
старший науковий співробітник
Ragozin Dmytro,
Ph.D. (technical sciences),
senior researcher
http://orcid.org/0000-0002-8445-9921.

Смірнов Валентин Євгенович,
аспірант
Smirnov Valentyn,
post-graduate student
http://orcid.org/0009-0006-4022-5951.

Місце роботи авторів:

Інститут програмних систем
НАН України
Institute of Software Systems of the
National Academy of Sciences of Ukraine
тел. +38-044-522-62-42
E-mail: ukrprog@isofts.kiev.ua
www.iss.nas.gov.ua