This study investigates unmanned aerial vehicle (UAV) networks operating under the influence of destructive and hostile factors. The study considers, among destructive factors, changes in signal power at the receiver side caused by distance and battery charge limitations. Among the hostile factors, cyber-physical threats have been examined, including those caused by electronic countermeasure (ECM) systems and malware-based attacks.

Algorithmic and software solutions have been developed to simulate the behavior of UAVs under such constraints. A special feature of the proposed model, in contrast to existing approaches, is that it integrates factors such as signal degradation caused by ECM systems and the dynamics of malware propagation within the network.

Scenarios include UAV behavior under jamming, the probabilistic spread of malware, and the switching of operational modes in response to threat exposure. The results were achieved by integrating a wide range of parameters, including device identifier, signal power, transmitter radius, transmission frequency, geolocation, task type, malware sensitivity, message handling queue, propagation delay, as well as movement speed. These features enable the model to realistically reproduce system behavior in uncertain and hostile environments, allowing both defensive and offensive security strategies to be explored. Devising appropriate operational scenarios is also possible.

The proposed software solution is characterized by the high level of detail in the simulation and the use of the Rust programming language, which ensures performance, modularity, and future extensibility. The solution reported here supports visualization of the behavior of up to 100 UAVs and more through both images and animations. It can be used for analyzing attack scenarios, designing robust UAV architectures, and prototyping offensive security tools. The source code is publicly available at GitHub repository, supporting practical usage and further research applications

Keywords: UAV network simulation, cyber-physical security, malware, electronic countermeasures

UDC 629.8:62–71:681.5:004
DOI: 10.15587/1729-4061.2025.340918

DESIGN OF A SIMULATION TOOL FOR PLANNING UAV MISSION SUCCESS UNDER COMBAT CONSTRAINTS

Anton Tyshchenko*
Iryna Stopochkina
Corresponding author
PhD, Associate Professor*
E-mail: i.stopochkina@kpi.ua
*Department of Information Security
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Beresteyskyi ave., 37, Kyiv, Ukraine, 03056

Received 14.07.2025
Received in revised form 18.09.2025
Accepted date 29.09.2025
Published date 28.10.2025

How to Cite: *Tyshchenko, A., Stopochkina, I. (2025). Design of a simulation tool for planning UAV mission success under combat constraints. Eastern-European Journal of Enterprise Technologies, 5 (9 (137)), 14–26. https://doi.org/10.15587/1729-4061.2025.340918*

1. Introduction

The use of unmanned aerial vehicles (UAVs) and related technologies have become a critical component of both civilian and modern military operations. A prime example is the ongoing war in Ukraine, when UAV technologies are widely deployed on the battlefield, providing a significant strategic advantage. In modern warfare, UAV networks are exposed to a wide range of threats, the most serious of which are communication disruptions caused by electronic countermeasures (ECM) systems and the spread of malware.

Most existing UAV network modeling systems are adapted to civilian environments, such as industrial, agricultural, or logistics applications in peacetime, and usually do not take into account the complexities of combat environments. These include electronic warfare interference, cyberattacks, unreliable communication channels, as well as the risk of losing control over UAVs. As a result, their applicability to military simulations and operational planning is limited.

Research into this area is important because ensuring the reliability, resilience, and security of UAV networks is a key factor for the successful execution of missions under combat conditions. Construction of models that integrate mesh and star topologies, take into account electronic countermeasures, and reflect the dynamics of cyber threats allows for a deeper understanding of system vulnerabilities and supports mission planning and risk assessment. Such solutions have practical applications for improving UAV control systems, simulating battlefield scenarios, and training operators.

Given these considerations, research focused on modeling UAV networks under cyber-physical threats is relevant.

2. Literature review and problem statement

Malware poses a significant threat to the operational integrity of UAVs. It can be used to hijack control systems, interfere with firmware analysis, disrupt mission execution, or organize covert surveillance [1].

In [2], various categories of malware targeting UAV networks were investigated, which provide a meaningful understanding of the principles of UAV malware operation. It was shown that it is possible to categorize and explain the principles of malware operation. However, modeling of threats and their spread was not carried out. The reason for this is that the study focused only on the analytical part without involving network dynamics models.

In [3], documented cases of compromised UAV systems were reviewed; paper [4] considers malware in relation to specific types of UAVs. In [5], the role of data manipulation and DoS attacks on UAV networks is shown. In [6], examples of countering these threats are given, and in [7] a case is described where attackers were able to gain control over a system from several devices. These studies show that attacks using malware are real and highly effective. However, models of malware propagation in complex network topologies taking into account the principles of device operation have not been built. The reason is the lack of available tools for

reproducing scenarios taking into account combat conditions. To overcome the difficulties, it is necessary to develop software solutions aimed at modeling the spread of malware in UAV networks close to actual conditions.

In [8], a formal model for UAV trajectory planning using three-dimensional cellular automata was proposed. A similar approach was used in [9] for obstacle-aware path planning. Cellular automata have also been used to coordinate robot swarms [10]. These discrete rule-based methods are suitable for modeling the movement of mobile agents, including UAV swarms. Further studies [11, 12] also show how cellular automata can be used for UAV behavior modeling tasks, in particular, for shortest route planning and device path planning. However, cellular automata have limitations when modeling battlefield dynamics: they assume synchrony of transitions and neglect stochastic delays, time, and functional characteristics of devices. This is due to the simplified discrete nature of the model. An option to overcome these difficulties may be the use of stochastic or hybrid simulation models that can take into account temporal and spatial factors, as well as adverse environmental factors.

In [13], a genetic algorithm-based approach was proposed for optimizing flight trajectories under energy constraints. However, the work did not take into account factors such as the spread of malware and the presence of electronic warfare (EW). The reason is the focus of the work on civilian applications of UAVs, which is also typical of previous studies. The way to overcome these gaps is to provide solutions that also take into account the aspect of operation in environments complicated by military operations. Such solutions should not contain information with limited access but should be based exclusively on open data, which distinguishes them from closed military technological advancements.

An alternative approach based on Petri nets is reported in [14–17]. In [14], a model for monitoring tasks is proposed, taking into account the battery charge of devices. In [15], a model is built that shows the interactions of a UAV network, however, without a control center. Paper [16] considers technical security issues that can be taken into account in the context of a model based on Petri nets. Work [17] is aimed at modeling UAV clusters, taking into account the mission of the corresponding cluster. It is shown that these models reproduce parallel processes and resource allocation well. However, they are not intended for spatiotemporal dynamics or highly detailed behavior. The reason is the conceptual limitations of the Petri nets apparatus. The solution can be to combine Petri nets with other simulation tools, or to use other approaches.

More general-purpose simulators, such as NS-3 [18], OMNeT++ [19], and the GAMA platform [20], have shown that they can be used to simulate standard attacks on networks. However, their architectures are focused on predefined scenarios and do not take into account the specificity of combat conditions, they are not designed to develop scalable, platform-independent applications. The reason is the initial purpose of these tools for civilian tasks and scientific research tasks, which is overcome by taking into account battlefield conditions, in particular cyber-physical attacks on UAVs.

In [21], the fragmentation of existing simulators with a video interface and their orientation mainly to the simulation of flight physics, sensor systems, and navigation tasks is emphasized. At the same time, there is a lack of consideration of specific battlefield conditions, which is explained by the fact that these simulators are intended mainly for operator training tasks, and not for planning missions with many devices. In addition, the problems of scaling existing simulators to UAV swarms are noted, which are overcome by the implementation of modular and fast software solutions. The reason is the use of languages, algorithms, and platforms that are not always focused on scalability and performance. The solution can be the development of software solutions using languages and algorithms that meet the relevant requirements.

The above gives grounds to argue that it is advisable to conduct research into designing a simulator for UAV mission planning. The model should be built taking into account the influence of obstacles [22], the spread of malicious software [6], and realistic logic for coordinating devices in topologies relevant to actual situations in military operations [23].

3. The aim and objectives of the study

The aim of our research is to build a scalable and high-performance simulation model for UAV networks operating under cyber-physical threats. The proposed solution would make it possible to integrate real-world operational constraints and scenarios into the simulation and serve as a flexible tool for both researchers and practitioners in the field of UAV security.

To achieve the goal, the following tasks were set:

– to define the relationship between UAVs, enemy devices, and the control center;

– to determine cyber-physical attack scenarios, including the impact of electronic countermeasures and malware, reflecting operational conditions arising under modern battlefield conditions;

– to implement a software simulation model and verify it for UAV networks.

4. The study materials and methods

The object of our study is unmanned aerial vehicle (UAV) networks operating under destructive and hostile factors. Destructive factors include changes in signal power on the receiver side because of distance and battery limitations.

The hypothesis of the study assumes that a simulation rule-based modeling approach could provide more realistic and operationally relevant prediction of UAV network behavior than existing universal modeling frameworks.

The following assumptions were adopted when constructing the simulation model:

1. Network topologies: UAVs operate using either mesh or star communication topologies.

2. Communication structure: Each UAV communicates with the control center, either directly or indirectly through repeaters.

3. Electronic countermeasure interference: Electronic countermeasure systems are capable of jamming communication signals, with the effect being determined by the signal power and distance to the transmitter/receiver.

4. Autonomous behavior: UAVs are capable of autonomously responding to loss of communication based on a set of predefined behavior scenarios.

Simplifications accepted in the study:

1. Environmental noise and terrain effects are not explicitly modeled but are indirectly represented through signal degradation functions.

2. UAV power consumption and battery limitations are considered in a simplified form: each device has a charge, the level of which is indicated in conditional units. When the charge is discharged, the device fails.

3. Malware implementation involves deterministic activation after successful connection to an adversary network node.

4. UAVs were considered as points in three-dimensional space.

Modeling approach and software implementation.

The modeling software was developed in the Rust programming language to ensure high performance, modularity, and scalability. The source codes are publicly available at the GitHub repository. The application was implemented with a flexible architecture consisting of a backend and a frontend.

The modeling platform supports modeling up to 100 UAV nodes and provides functions for defining scenarios, importing/exporting data, and visualizing UAV motion and network states. Communication scenarios were defined to reflect the interaction between transmitters and receivers and the impact of control messages on UAV behavior.

Network topologies.

Modeling supports both mesh and star topologies. In a mesh topology, UAVs communicate with each other and with a control center to coordinate mission tasks. In star topology, one or more relays act as intermediaries between the UAV group and the control center. This topology is particularly relevant for real-world battlefield scenarios where the number of UAVs is limited, as is the case in the current Ukrainian context. Relay devices are considered vulnerable to electronic countermeasures, and signal degradation or loss is modeled dynamically.

Electronic countermeasures simulation.

The following effects of ECM systems were implemented in the model:

1. Powerful jamming, leading to complete signal loss and disconnection from the control center.

2. Low-power jamming, which weakens the signal, leading to poor communication and reduced data transmission quality (e.g., photos or videos).

3. GPS (Global Positioning System) data spoofing, which replaces the original navigation data, potentially redirecting the UAV or causing a mission failure. This mechanism can also serve as a vector for introducing malware.

The simulated effects of electronic countermeasures take into account the signal frequency, jamming radius, and distance-dependent signal attenuation. Signal loss is determined by comparing the power of the control signal with the power of the jamming signal.

Real-world EW tactics observed in Ukraine.

The behavior of electronic countermeasure systems, as part of the EW used by the Russian military, as reported in open sources [23–25], was used to define realistic jamming scenarios. These include stationary powerful jamming devices deployed to protect key positions, as well as mobile electronic countermeasure installations.

Empirical data from Ukrainian UAV operators indicate that in the event of signal loss, many UAVs try to autonomously restore the signal by climbing until the control connection is restored. This maneuver increases the probability of signal restoration and prevents forced landing in electronic countermeasure zones. In some cases, UAVs can follow programmed trajectories after disconnection, although they have a lower probability of restoring the connection, being in an area under the influence of electronic countermeasures.

Data sources.

Numerical data for the software were obtained from open descriptions of radio electronic warfare equipment and empirical data on the current use of UAVs in Ukraine, including [23–25].

Software environment used in the development of the software:

– rustc 1.88.0 – Rust programming language code compiler;

– vim 9.1.1215 – text editor for writing the main application code in the Rust programming language and the auxiliary code in Python;

– git 2.51.0 – an application for managing versions of the developed software;

– Mozilla Firefox 142.0.1 – a web browser for viewing generated visualizations and managing the repository on GitHub;

– docker 28.4.0 – an application for containerizing the Rust programming language toolkit;

– cargo 1.88.0 – a utility that makes it possible to manage Rust packages, compile and test the written code of the developed software;

– Python 3.13.7 – an execution environment for the Python programming language code.

5. Results of building a model of a network of unmanned aerial vehicles taking into account interfering factors

5. *1.* Definition of communication between interacting devices

The system modeled in this study consists of three types of entities:

1. UAV network is a system of interconnected unmanned aerial vehicles that receive and process messages from a control center. The developed solution was tested with a number of devices of 100 units or more.

2. Control center is a stationary unit that transmits signals and messages to a controlled network of UAVs. The model includes a single control center that corresponds to actual deployment conditions.

3. Malicious devices are entities that disrupt the operation of the UAV network and the control center, generating interference that impairs communication between transmitters and receivers, similar to the functioning of electronic countermeasure systems. They also carry out GPS spoofing attacks, sending falsified GPS data to UAVs and spreading malware throughout the network.

4. Electronic warfare (EW) is a set of tactical and technical measures aimed at detecting, complicating, or neutralizing the electromagnetic capabilities of the enemy. Technical means that implement the functions of jamming or replacing navigation signals are referred to in our paper as ECM systems.

The principles of interaction between UAVs and enemy devices are illustrated in Fig. 1.

A special data structure called Signal was designed to implement communication principles. This structure includes the transmitter and receiver identifiers, an additional data payload, the transmission frequency, and the signal power, which depends on the transmitter signal power at the receiver location.

The Signal structure provides the ability to model various data in the program (e.g., GPS data, control commands) and interference signals (e.g., electronic countermeasure noise), depending on the role of the sender.

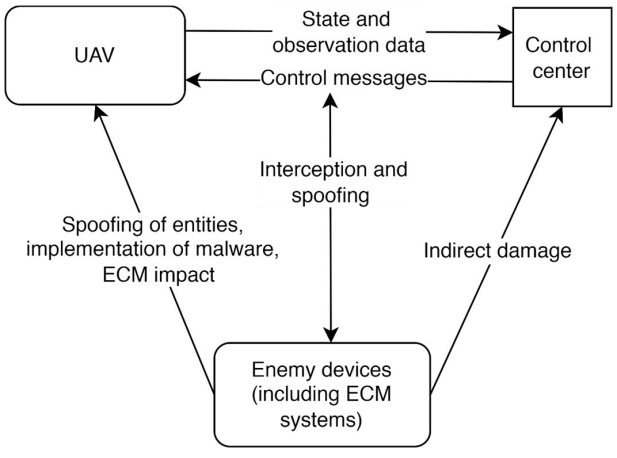

Fig. 1. The impact of enemy devices on unmanned aerial vehicles and the control center

For a signal to be successfully received, the following conditions must be met:

1. The receiving module (RX) in the software must analyze the frequency of the incoming signal.

2. The current signal supported by the RX module must have a lower level than the new incoming signal.

3. The level of the received signal must not exceed the maximum threshold that the RX module can process; otherwise, the signal will be perceived as unreadable noise, and its data will not be extracted.

This feature allows for a more realistic simulation of ECM means modeled as transmitters that send empty but powerful signals to all devices within their effective range. The power of these signals is determined by the characteristics of the ECM transmitter module (TX).

The communication between devices is described based on two introduced concepts: signal power and signal level. Both metrics serve to quantify the quality of the connection between devices. The key difference between them is their nature: power is a continuous value, while level is discrete. In particular, the following signal levels are distinguished: L_{green}, L_{yellow}, L_{red}, L_{black} (high, medium, and low quality, respectively).

The calculation of the transmitter signal power at a given distance is based on the free space path loss formula, assuming that the distance between devices is significantly greater than the wavelength of the transmitted signal

$$\frac{S_r}{S_t} = k\left(\frac{\lambda}{d}\right)^2, \tag{1}$$

where S_r is the signal power at the receiver side, S_t is the signal power at the transmitter, k is a constant, λ is a constant, d is the distance between the devices.

Calculate the receiver signal level L_r at a certain distance d from the transmitter at level L_t

$$L_r = \begin{cases} L_t, d \geq R_{green}, \\ lower\left(L_t\right),\ R_{green} < d \leq R_{yellow}, \\ lower\left(lower\left(L_t\right)\right),\ R_{yellow} < d \leq R_{red}, \\ L_{black},\ d > R_{red}\ . \end{cases} \tag{2}$$

The calculation of the receiver signal level $\hat{L}_r$ under the influence of ECM system at level L_{EW} is performed according to the following rules:

$$\text{if } L_{EW} = L_{black} : \hat{L}_r = L_{red}, \tag{3}$$

$$\hat{L}_r = L_{yellow},\ \text{if } (L_{EW} = L_{red}) \text{ and } (L_r = L_{green}), \tag{4}$$

$$\hat{L}_r = L_{red},\ \text{if } ((L_{EW} = L_{yellow}) \text{ and } (L_r = L_{green}))$$

or

$$((L_{EW} = L_{red}) \text{ and } (L_r = L_{yellow})). \tag{5}$$

The signal reception occurs with a certain probability, which depends on the level and takes values P_{green}, P_{yellow}, P_{red}, P_{black}.

It is assumed that the positions of electronic countermeasure means are known in advance and can be obtained using radio reconnaissance. Electronic countermeasure means are located in such a way that their zones of activity do not overlap. Scenarios of the impact of electronic countermeasures on UAVs are considered in the chapter "Cyber-physical attack scenarios".

5. 2. Cyber-physical attack scenarios

Scenario 1. Impact of ECM, loss and recovery of communication with UAV.

ECM devices are transmitters that broadcast a high level/power signal, but not data, to all devices in range.

The model takes into account the negative impact on the frequencies of the UAV control networks and GPS.

Let D represent the UAV, C denote the control center, J is the ECM jamming device. Quantity r_J is the effective interference radius of J, $d(D,J)$ is the Euclidean distance between D and J, f_c is the control signal frequency, S_C is the signal power from the control center, S_J is the interference signal power at frequency f_C, t is the current time. Let A_D be a set of predefined actions of the UAV in case of loss of connection, state $D(t)$ is the current operating state of aircraft D, $Conn(D,C) \in \{1,0\}$ is the connection state (1 means "connected", 0 means "connection lost"):

Step 1. Checking the interference zone.

If $d(D,J) < r_J$ then D is inside the interference zone.

Step 2. Signal and frequency conflict.

If $freq(J) = f_C$ *ma* $S_J > S_C$ then $Conn(D, C) := 0$.

Step 3. Reaction to loss of communication.

If $Conn(D, C) = 0$, then the UAV switches to autonomous behavior

$$stateD\left(t\right) \cdot select\left(A_D\right),$$

where

$$A_D = \left\{Hover, Ascend, ReturnToPoint, ContinueTask, FailSafe\right\}.$$

The choice of actions is made according to

$$select\left(A_D\right) = f\left(DroneType, FailSafeSettings\right).$$

Step 4. Reconnection.

If at time $t' > t$

$$S_C\left(t'\right) > S_J\left(t'\right) \Rightarrow Conn\left(D,C\right) := 1,$$

then

$$ReceiveNewTask\left(D\right); stateD\left(t'\right) := ExecuteTask.$$

Scenario 2. Impact of ECM on GPS communication.

Let f_{GPS} be the frequency of the GPS signal, S_{GPS} be the GPS signal power at the position of UAV D, S_J be the interference signal at frequency f_{GPS}. The value t is the current time, $dir_{D(t)}$ is the direction of UAV at time t, $pos_{D(t)}$ is the current position of UAV D, $Conn(D, GPS) \in \{1,0\}$ is the status of GPS connection (1 is connected, 0 is lost):

Step 1. Checking the interference zone.

If $d(D, J) < r_J$, then D is inside the GPS radio suppression zone.

Step 2. GPS signal conflict.

If $freq(J) = f_C$ *ma* $S_J > S_{GPS}$, then $Conn(D, GPS) := 0$.

Step 3. Reaction to loss of GPS connection.

If $Conn(D, GPS) = 0$, then the UAV continues horizontal movement in the current direction:

$dir_D(t') := dir_D(t)$ *for* $t' > t$;

$pos_D(t') := pos_D(t) + \Delta t\ dir_D(t)\ v_D$, where v_D is the UAV speed.

Step 4. Restore GPS connection.

If at time $t'' > t'$

$$S_{GPS}(t'') > S_J(t'') \Rightarrow Conn(D, GPS) := 1,$$

The UAV requests the current location $pos_D(t'')$ via GPS and updates its navigation path accordingly.

$$UpdateNavigation(pos_D(t'')).$$

Scenario 3. *Propagation of UAV malware under the influence of an attacker.*

Let M be the malicious transmitter (attacker device), r_M be the effective radius of propagation of malware from M, $d(D, M)$ be the Euclidean distance between UAV D and the malicious device M, $State(D) \in \{V,I,P\}$ be the vulnerability state of D, where V be vulnerable, I be infected, P be protected, T_{infect} be the time delay between receiving a signal with malware and infection, T_{spread} be the time delay between infection and transmission of malware W to others, *PayloadType* be the classification of malware impact (e.g., data leakage, control seizure, denial of service).

Malware spreads through signals transmitted by infected UAVs or attacker devices.

Rules for changing states.

Initial states. $State(D) = V$ for uninfected, unprotected devices. $State(D) = P$ for devices resistant to malware infection. $State(D) = I$ – absorbing since infected devices cannot be disinfected.

Permissible transitions:

– $V \rightarrow I$ after delay T_{infect}.

– I, P – final states.

Step 1. Checking the area of infection with malware.

If $d(D, M) < r_M$, then UAV D is inside the infection area of device M.

Step 2. Receiving the signal with malware.

The malicious device M sends a signal containing malware to UAV D. If $State(D) = V$, then the UAV enters the latent infection state

At time $t + T_{infect} \Rightarrow State(D) := I$.

Step 3. Spreading the malware to infected UAVs.

If $State(D) = I$ *ma* $t' > t + T_{infect} + T_{spread}$, then D starts broadcasting malware signals to all UAVs D' within communication range.

Step 4. Recursive infection.

For each UAV D', receiving a malware signal from D

If $State(D') = V$, then at time $t'' := t' + T_{infect} \Rightarrow State(D') := I$.

Scenario 4. GPS spoofing.

This attack is based on the fact that GPS signal sources do not perform authentication. As a result, any device can spoof GPS data and transmit it to other devices. If the delay of signal transmission from the attacker's device is less than that of legitimate GPS signals, and the signal quality is higher, the victim UAV will receive and use the fake signals. The attacker's device itself functions as a standard transmitter.

Let J be the GPS spoofing device, r_J be the effective range J, $d(D, J)$ be the Euclidean distance between D and J, S_{GPS} be the power of the real GPS signal, S_J be the power of the spoofed GPS signal from J, $Conn(D, GPS) \in \{1,0\}$ be the GPS connection status. GPS_{true} are the real GPS coordinates, GPS_{spoof} are the spoofed coordinates from the attacker:

Step 1. Checking the spoofing zone.

If $d(D, j) < r_J$, then UAV D is within the range of malicious J.

Step 2. Receiving the spoofed signal.

Malicious J broadcasts the spoofed GPS data GPS_{spoof} to UAV D.

Step 3. Evaluating the spoofing condition.

If $S_J > S_{GPS}$, then GPS_{spoof} is accepted by D and $Conn(D, GPS) := 1$ (spoofed).

Step 4. Incorrect navigation.

UAV D updates its navigation system using the fake coordinates

$Navigation(D) := GPS_{spoof}$ and continues its movement based on the spoofed position and trajectory data.

Step 5. Spoofing termination condition.

If at some time $t' > t$, $d(D, J) \geq r_J$, then $Conn(D, J) := 0$ (loss of the substituted signal) and the UAV resumes receiving the GPS signal from legitimate satellites and recalculates the true location:

$$GPS_{true} := Acquire();$$

$$Navigation(D) := GPS_{true}.$$

The above scenarios make it possible to algorithmize the behavior of UAVs under adverse conditions and implement a computer model that will simulate the behavior of the UAV network under various conditions. Among such conditions are the impact of ECM means, malware distribution devices, which leads to falsification of GPS data and unavailability of UAVs.

5. 3. Software package implementation

The implementation was carried out in the Rust programming language, which outperforms Python and C/C++ in several critical aspects:

1. Performance. As a compiled language, Rust is generally faster than interpreted languages such as Python. While it may be slightly less performant than lower-level languages such as C and C++, Rust provides a favorable balance between speed and safety.

2. Memory safety. Unlike C and C++, which require manual memory management, Rust provides memory safety

through a borrowing-checking mechanism that prevents data conflicts.

3. Graph processing libraries. Graph visualization was implemented using the petgraph and rustworkx-core libraries. These libraries support not only the creation of static illustrations but also the generation of videos, which makes the simulation results more visually understandable and dynamic.

The proposed solution is characterized by low requirements for computing resources while maintaining the ability to perform operations on graphs up to 100 nodes and above. The solution is able to provide visualization of computational results in real time without compromising performance. The software package is designed to work in Windows or Linux operating systems with x86-64 processor architecture as a cross-platform application that does not require specific hardware.

A comparison of the implemented software package with known platforms and modeling tools suitable for modeling UAV networks is given in Table 1. This comparison evaluates key characteristics related to the modeling of cyber-physical threats, such as support for modeling the effects of electronic countermeasures, malware propagation mechanisms, flexibility in determining network topology, and visualization capabilities.

When developing the software, it was possible to replace the implemented variable values with those that correspond to the characteristics of real aircraft, ensuring the ability to adapt the simulator to the user's needs. In particular, the speed, flight altitude, flight range, and behavior scenarios when entering the zone of the electronic countermeasure system were taken into account. The initial data and types of behavior used in the simulation best correspond to multirotor UAVs. By replacing the values of speed, flight range, flight altitude, the software package can be reconfigured for specific UAV models. The communication topologies correspond to the mesh model (interaction of each with each) and the star model (on repeaters).

The developed simulator software package consists of software code in the Rust language and is provided in open access on [26].

A series of experiments were conducted to demonstrate the capabilities of the software implementation of the model and to compare networks that use signal power (S-networks) and networks that use level (L-networks). The parameters for each model used in the experiments are given in Table 2. Formulas (1) (S-networks), (2) to (5) (L-networks) and the scenarios from the previous chapter were used during the simulation.

In the plots (Fig. 2, 3), the colored dots represent devices, mainly UAVs. The point surrounded by a green circle indicates the control center. The red circle represents the approximate area of influence of the device that disrupts the GPS connection or spreads malware (Fig. 5, 6). The orange circle indicates the approximate area of influence of the enemy device that performs a GPS spoofing attack (Fig. 7, 8).

Experiment 1. Star topology in the study of connection quality.

Fig. 2, *b* shows that the connection quality in the L-network is unstable, which is explained by the non-determinism of the signal level calculation.

The influence of topology on establishing a connection in a network with the same parameters was considered in experiment 2.

Experiment 2. *Network topology in the study of connection quality.*Thus, the change in network topology had a minor impact on the quality of the S-network connection (Fig. 3, *a*). Changes in the L-network (Fig. 3, *b*) are due to its non-determinism associated with the influence of electronic countermeasures, which is considered probabilistically in this model. Additionally, an experiment was conducted with the reaction of devices to the loss of communication with the control center. The following behavior options were implemented: continued movement to the target in a straight line (and self-destruction), movement to restore the connection, free flight in search of restoration of the connection, movement to the control center, failure. Testing showed the correct reaction to the programmed scenarios.

Table 1

Comparing the proposed solution with existing ones

Features	Our Simulator	NS-3	OMNeT++	GAMA Platform	CPN Tools/IDE
Electronic countermeasures simulation (jamming, GPS data spoofing)	Detailed electronic countermeasures (GPS signal jamming, radio frequency noise, spoofing)	Limited (requires special models)	Limited (add-ons available)	Not included in platform tools	Not applicable
Malware propagation simulation	Integrated state transitions under the influence of malware (V/I/P)	Not directly supported	Possibly through extensions	Possibly via scripts	Formally possible, but lacks physical simulation
The difference between signal power and signal level	Both are simulated, configurable for each device	Signal power modeling	RSPI (Received Signal Power Indicator) is supported	Abstract vision of the agent	Not used
Configuring device behavior under the influence of ECM/malware	Different options supported (user defined)	Manual topology adjustment	Built-in topology modules	Agent-based layout	Can configure token routing
Time-step based movement + communication	Unified space-time modeling	Modeling of discrete events	Modeling of discrete events	Continuous agent movement	Focus on event logic, not movements
Real-time terrain/RF model support	Planned (Digital Elevation Model integration)	Through external relief models	Extensions through the INET framework	GIS ready	Not supported
Visual frontend (visualization)	3D visualization and video are integrated	Basic via NetAnim	Progressive graphical user interface	Real-time 2D, 3D	Visualize only token shuffling
Orientation to cyber-physical systems	A sustained focus on cyber-physical systems under attack	Focus on networks	Focus on networks	Focus on socio-ecological issues	Formal simulation of cyber-physical systems
Openness and extensibility	Open (release on GitHub)	Yes	Yes	Yes	Yes

Table 2

Experiment parameters

Parameter	Value	Experiment number
Number of UAVs	100 and 1 control center	1, 2, 3, 4, 5, 6
Network topology	star	1, 3, 4, 6
	mesh	2, 5
Approximate range of the control center	200 m	1, 2, 5
	1000 m	3
	300 m	4, 6
Approximate UAV coverage radius	30 m	1, 2
	50 m	3, 4, 6
	25 m	5
Approximate GPS signal range	1000 m	1, 2, 3, 4, 5
Approximate ECM zone	100 m	4
	50 m	5
Approximate range of devices for the spread of malware	200 m	6
GPS device position	(0, 0, 200)	6
Falsified geolocation	(–200, –100, –200)	6
UAV target	not defined	1, 2, 3, 5
	target (0,0,0)	4, 6
Data latency multiplier	0	1, 2, 4, 5
	1500000	3, 6
Delay in the spread of malware	500 ms	5

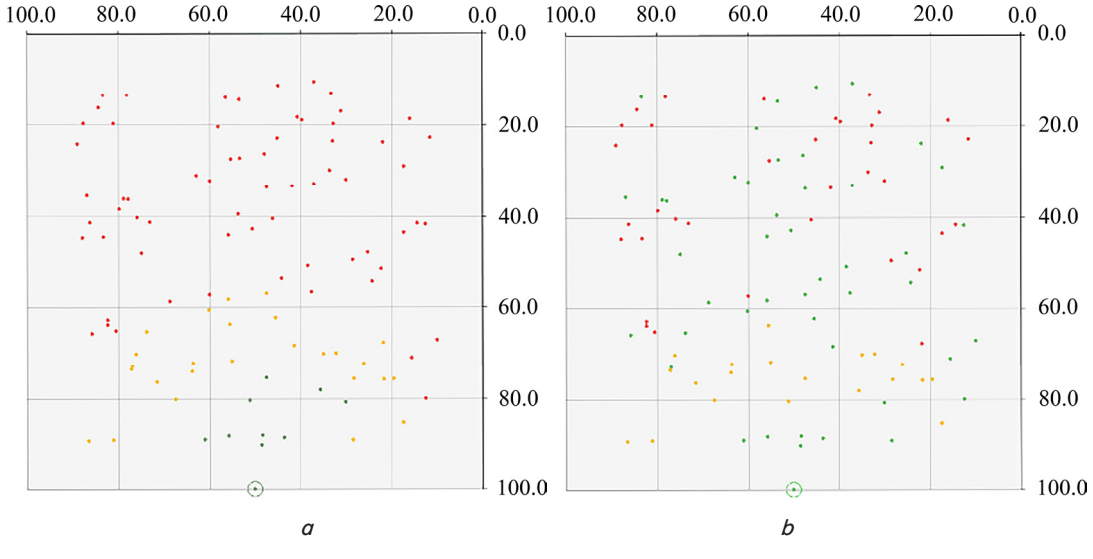

Fig. 2. Connection quality according to the star topology: *a* – S-network; *b* – L-network

Experiment 3. Data transmission delays.

The experiment analyzed the influence of delays with which UAVs receive signals with new targets from the control center. To compare the operation of networks with and without time delays, two networks were shown. In the main network, signals arrive with a certain time delay (the UAVs of which are marked in black). In the auxiliary network, signals arrive instantly (the devices of which are marked in red) (Fig. 4). All other parameters of these networks are identical. The behavior of networks with a "star" and "mesh" topology was analyzed, but in terms of network response speed, they look almost equivalent.

Our experiment confirmed that UAVs located closer to the control center receive signals and change their target earlier.

The difference between S- and L-networks is insignificant, which can be observed in Fig. 4, *a*, *b*, respectively.

Experiment 4. The effect of ECM, which degrades GPS communication.

The behavior of the UAV under conditions of loss of GPS communication was also tested (Fig. 5). Since the GPS device broadcasts signals directly to all devices within range, the choice of network topology does not affect this scenario.

Before entering the effective area of influence of electronic countermeasures, UAVs move along a straight line towards the target. After reaching the obstacle zone, they completely lose connection with GPS. Accordingly, they continue to navigate along a straight line, maintaining the previous course until the signal is restored.

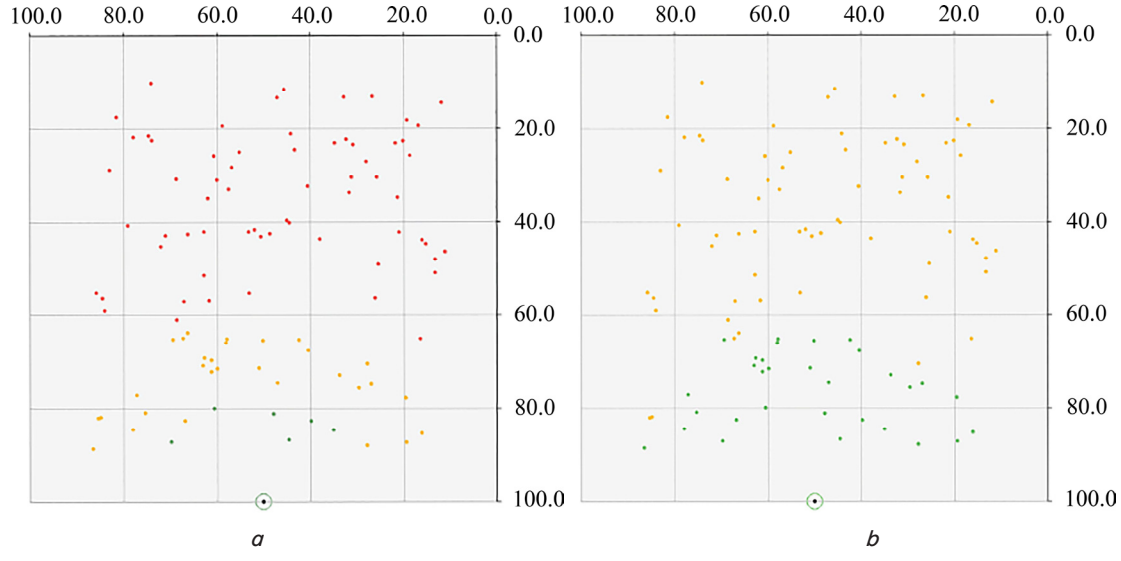

Fig. 3. Connection quality according to the "mesh" topology: *a* – S-network, *b* – L-network

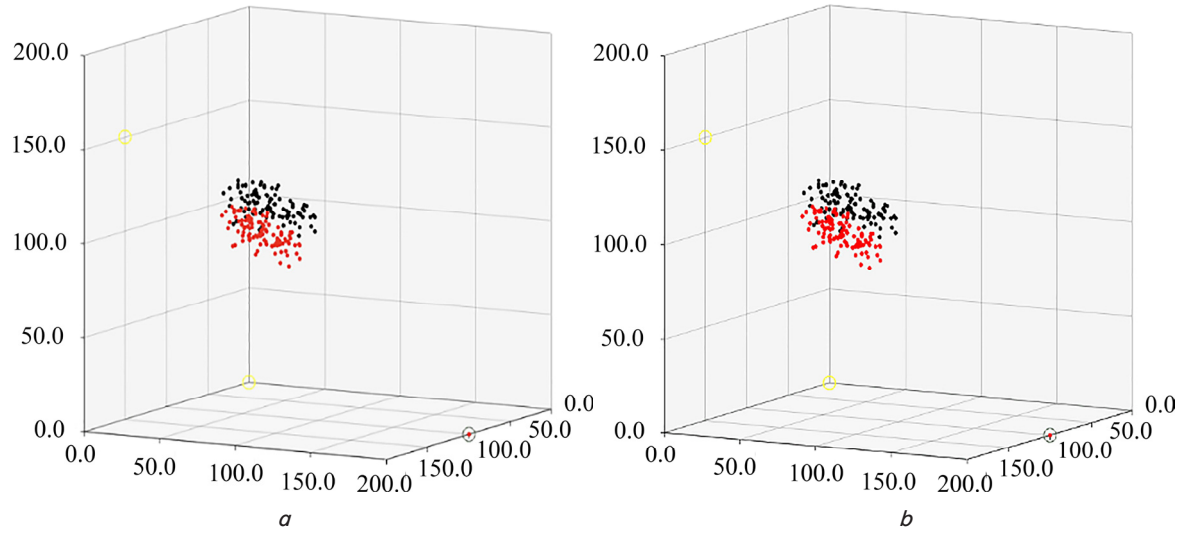

Fig. 4. Delays of changing the approach to the target of the UAV network according to the "star" topology: *a* – S-network, *b* – L-network

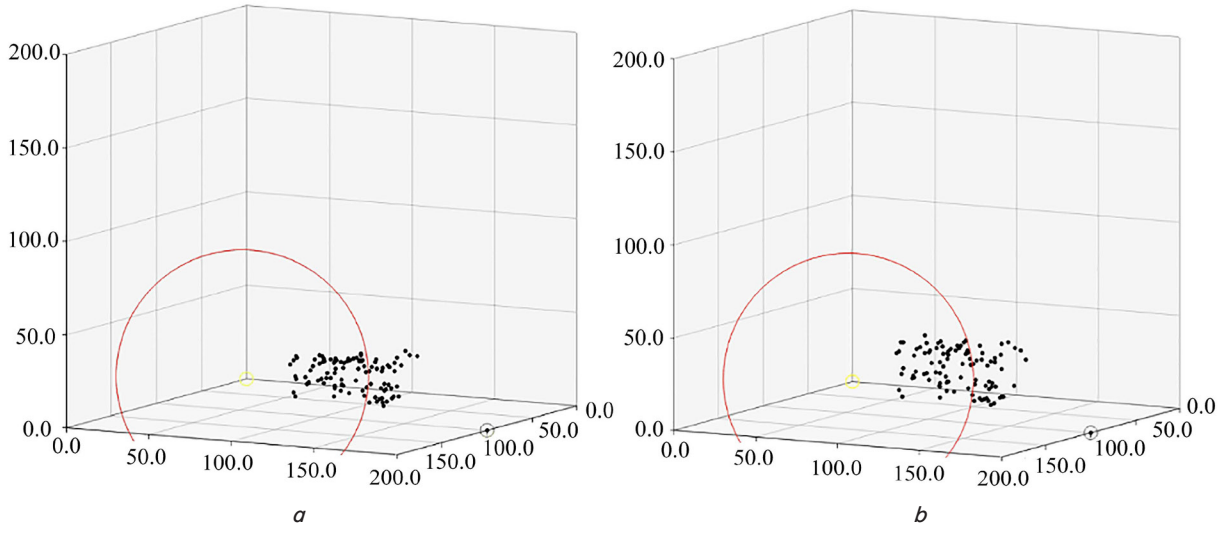

Fig. 5. Unmanned aerial vehicles in the effective range of electronic countermeasures: *a* – S-network; *b* – L-network

Experiment 5. Malware propagation in the network.

In networks with a "star" topology, malware does not spread since UAVs do not have sufficiently powerful transmitters to send malicious data back to the control center. Instead, the mesh topology makes it possible to demonstrate the spread of DoS malware, as shown in Fig. 6.

In this scenario, all devices are considered potentially vulnerable, although this can be controlled using the software parameter "vulnerability probability". The simulation illustrates the sequential disappearance of devices from the network as they become inaccessible due to infection and disruption caused by an attack. The simulation demonstrates how the network degrades over time. This experiment confirms that the mesh topology enables recursive transmission of malware.

Experiment 6. GPS spoofing.

In this scenario, UAVs move towards a target, which is depicted as a small orange circle enclosed within a larger orange circle representing the area of influence of the GPS spoofing signal (Fig. 7).

However, as soon as the quality of their connection to the attacker's device exceeds that of the real GPS device, they start receiving falsified geodata. This causes them to change their direction of movement (Fig. 8).

Metrics calculated using the developed software.

To illustrate the functionality and performance of the developed software, a set of quantitative indicators was calculated and analyzed.

One of these indicators is the packet delivery ratio (PDR), which is defined as the ratio of the number of data packets successfully received by the UAV from the control center to the total number of packets transmitted by the control center during the simulation.

PDR serves as a key indicator of the reliability of data transmission in the UAV network. The results of PDR calculation are shown in Fig. 9.

The success threshold (ST) indicates the minimum percentage of UAVs that must reach the target for the mission to be categorized as successful.

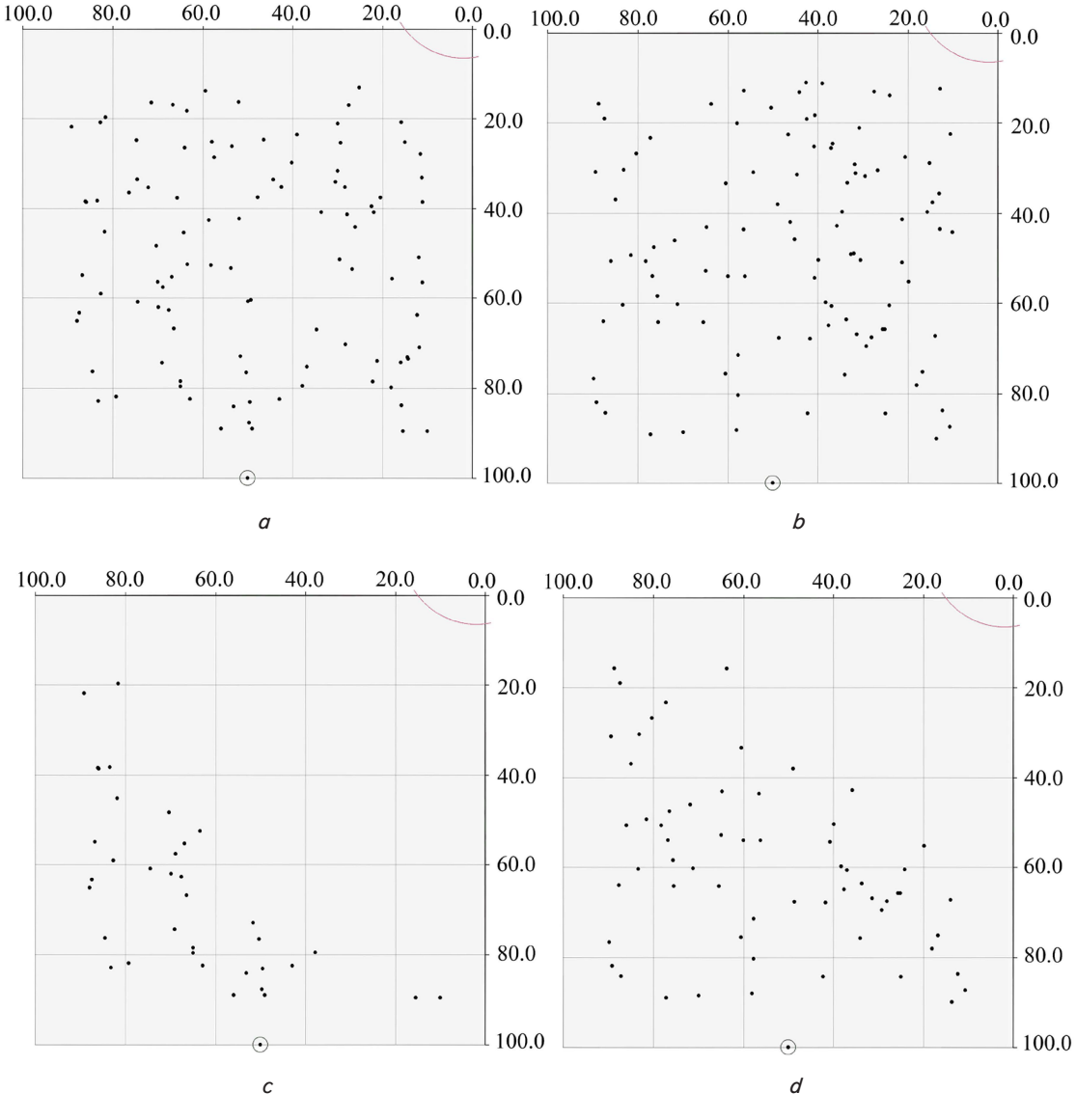

Fig. 6. Malware propagation, Mesh topology: *a* — S-network, step 1; *b* — L-network, step 1; *c* — S-network, step 50; *d* — L-network, step 50

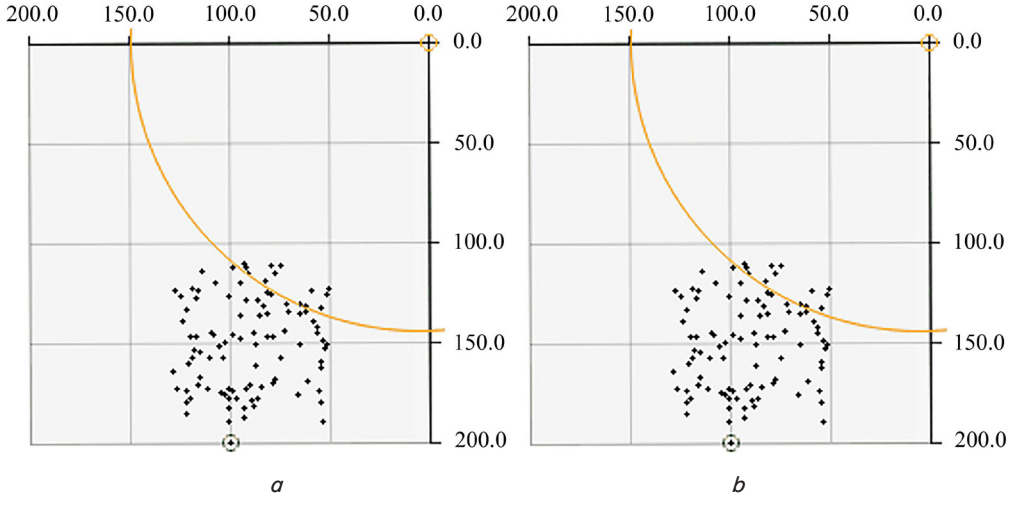

Fig. 7. UAV movement to the target within the GPS substitution device zone: *a* — S-network, *b* — L-network

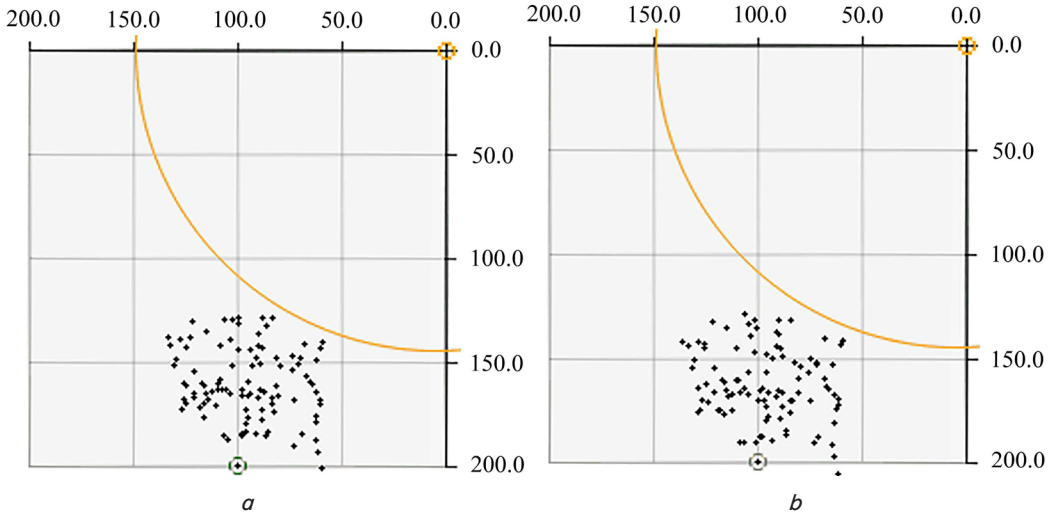

Fig. 8. Movement of unmanned aerial vehicles with falsified geodata in the opposite direction: *a* — S-network; *b* — L-network

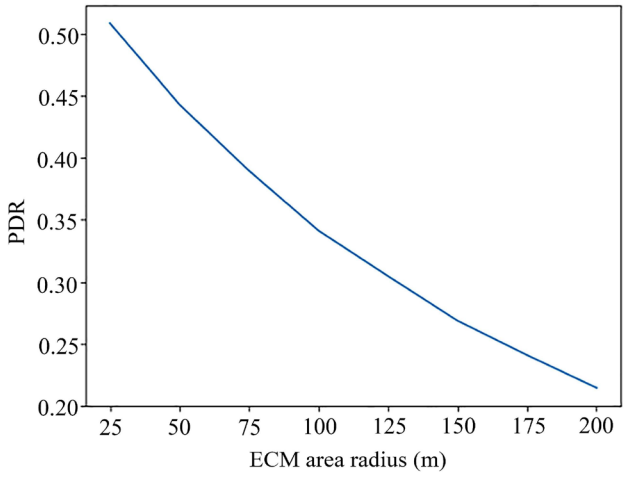

Fig. 9. Dependence of the packet delivery coefficient on the range of the electronic countermeasure device

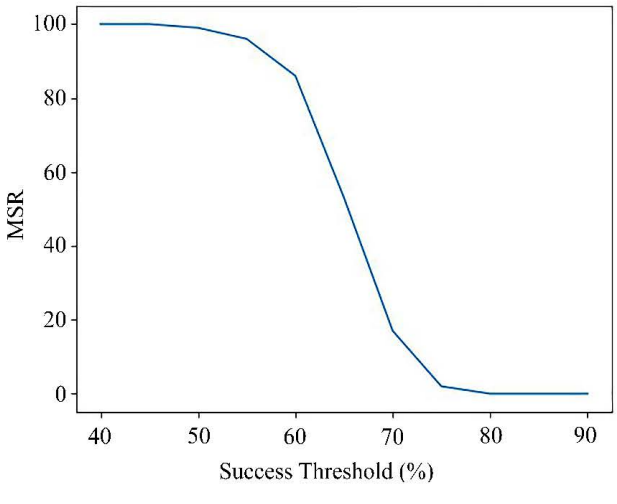

Fig. 10. Dependence of the average mission success rate on the success threshold value

The mean success ratio (MSR) reflects the average proportion of missions that met or exceeded the success threshold.

Additional modeling parameters used in this scenario included the following receival probabilities: $P_{green} = 0.95$, $P_{yellow} = 0.75$, $P_{red} = 0.5$, $P_{black} = 0.1$.

6. Characteristics of the proposed software model of the UAV network: discussion

Existing platforms [19–21] focus mainly on conventional network scenarios, such as routing and traffic modeling, and rarely take into account hostile battlefield conditions. In contrast, the proposed solution integrates the effects of electronic countermeasures (jamming and GPS data spoofing), recursive malware propagation, and device-specific behavior (relations (1) to (5), scenarios 1–4). This provides a higher level of relevance to the tasks of assessing resilience and tactical planning under modern military conditions.

The PDR plot (Fig. 9) demonstrates the impact of ECM tools on the success of communications in the UAV network. Sensitivity analysis of MSR with respect to different success thresholds (Fig. 10) revealed a typical nonlinear decline. Our work differs from cellular automata and Petri nets [9–11, 14–17], in which modeling is limited to discrete transitions without taking into account stochastic signal degradation, the influence of ECM means. Our result clearly connects the influence of signal attenuation and the spread of malicious software with the behavior of UAV. This allows for a more realistic representation of critical zones (60–70% of the threshold), where the success of the mission is sharply reduced.

The results reported here confirm the possibility of analyzing the availability of the UAV network using the developed software under normal operation conditions (experiments 1, 2), data transmission delays (experiment 3), and cyber-physical threats (experiments 4–6). They also demonstrate the ability of the model to reflect changes in UAV behavior strategies in response to various environmental and operational factors, in contrast to works [9–13, *15*–17].

Papers [18, *19]* that describe the NS-3 and OMNeT++ environments demonstrate the effectiveness of these tools for modeling standard network attacks. However, unlike them, our simulator reproduces not only loss of communication but also signal attenuation, delays, coordinate substitution, and recursive propagation of malware at the device level. This becomes possible due to formalized scenarios that take into account real tactical conditions of the battlefield.

The technical implementation also differs in that most academic prototypes are implemented in Python, which limits their performance in complex scenarios, or in C++ [22], which potentially poses security issues. In contrast, the proposed implementation in Rust provides safe memory management and high-speed graph operations for networks of up to 100 nodes and above. This makes it possible to simulate UAV movement, communication, and enemy interference simultaneously without overloading resources.

The resulting software package combines the effects of ECM and malware propagation with realistic UAV behavior logic, while maintaining computational efficiency. Unlike isolated models, it offers an integrated environment for analyzing the resilience of UAV networks. Its open source code creates conditions for further expansion, including terrain-aware routing scenarios and hybrid attacks.

Experimental tests have confirmed the performance and suitability of the framework for tactical planning. In contrast to models that only assess the controllability of devices [14, 17], or studies that only describe the effects of cyber-physical attacks [2, 5], our work recreates a simulated scenario of GPS signal loss. Additionally, signal restoration maneuvers and malware propagation (≈83 infected UAVs out of 100 in the mesh topology) are recreated. Using quantitative indicators (PDR, MSR, number of infected devices), the reproducibility of the results and the suitability of the model for assessing the success of planned missions in combat conditions have been shown. This, at the same time, can be considered as a step towards introducing indicators for assessing the resilience of UAV networks and tactical mission planning, which responds to one of the challenges identified in [22].

There are certain limitations of our study: the simulation does not take into account the effect of terrain on the movement and communication of devices, the movement is uniform, the devices are not able to collide with each other and are represented by points in space.

The study's results provide a basis for further work aimed at devising optimized UAV response strategies adapted to specific mission objectives and threat scenarios, in order to increase the overall network resilience and mission reliability. Future research should focus on integrating terrain- and energy-sensitive modeling into the modeling system to more accurately reflect operational conditions in real military scenarios. In parallel, the modeling of the battery charge level of each device considered in this solution in some conditional units, which is affected by the distance traveled, could additionally take into account the signal transmission activity and external interference, which would facilitate the evaluation of energy-efficient strategies.

7. Conclusions

1. We have defined relationships between UAVs, malware propagation devices, ECM tools, and the control center. Unlike cellular automata or Petri net models, stochastic signal degradation and adversary dynamics were taken into account, while omitting low-impact environmental details. That made it possible to build a mathematically understandable structure that could be scaled to battlefield scenarios. A feature of our result is the explicit linking of signal attenuation and malware propagation to UAV behavior, which provides an advantage over purely abstract swarm models.

2. The formalized scenarios devised describe jamming, GPS spoofing, and malware propagation, as well as device signal behavior. Compared to NS-3 or OMNeT++ environments, this solution makes it possible to model loss of communication under the influence of ECM tools, signal attenuation, coordinate substitution, and recursive malware propagation at the device level.

3. The developed software package improves the simulation of the UAV network by combining the effects of electronic countermeasures and malware with realistic behavioral logic, while maintaining computational efficiency. The open source release of the software ensures reusability; its modular design supports future extensions such as terrain-aware routing and hybrid attack scenarios. The software provides high performance for graph operations at the level of up to 100 nodes and more. Tests have shown that the response speed exceeds Python prototypes, while maintaining modularity and extensibility, which distinguishes the framework from academic prototypes with low performance and ex-

plains its ability to simultaneously simulate UAV movement, communication, and enemy interference without overloading resources.

Experimental verification of the software confirmed the operability and compliance of the framework with the tasks set. Our results included demonstrations of unstable connectivity in L-networks due to probabilistic reception, predicted GPS signal loss and recovery maneuvers, as well as high infection rates (≈83 out of 100 UAVs) under the condition of spreading malware with a mesh topology. Quantitative metrics such as packet delivery rate and mission success rate (up to 100% at deterministic connectivity, reduced by electronic countermeasures) confirmed that the model provides measurable, reproducible results. These findings confirm the suitability of the framework for resilience assessment and tactical planning.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

The associated data are available in a GitHub repository [23].

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies in creating the presented work.

Acknowledgments

This work was inspired in part by discussions and research questions arising from the project "Towards Networked Airborne Computing in Uncertain Airspace: A Control and Networking Facilitated Distributed Computing Framework." The authors declare that the publication costs were covered by their own funds.

References

1. McNabb, M. (2025). Ukraine's Trojan Horse Drones: A New Frontier in Cyber Warfare. DroneLife. Available at: https://dronelife.com/2025/04/10/ukraines-trojan-horse-drones-a-new-frontier-in-cyber-warfare/
2. Yaacoub, J.-P., Noura, H., Salman, O., Chehab, A. (2020). Security analysis of drones systems: Attacks, limitations, and recommendations. Internet of Things, 11, 100218. https://doi.org/10.1016/j.iot.2020.100218
3. Hartmann, K., Giles, K. (2016). UAV Exploitation: A New Domain for Cyber Power. 6 8th International Conference on Cyber Conflict. Tallinn, 205–221. Available at: https://ccdcoe.org/uploads/2018/10/Art-14-Assessing-the-Impact-of-Aviation-Security-on-Cyber-Power.pdf
4. Abro, G., Zulkifli, S., Masood, R., Asirvadam, V., Laouiti, A. (2022). Comprehensive Review of UAV Detection, Security, and Communication Advancements to Prevent Threats. Drones, 6 (10), 284. https://doi.org/10.3390/drones6100284
5. Riahi Manesh, M., Kaabouch, N. (2019). Cyber-attacks on unmanned aerial system networks: Detection, countermeasure, and future research directions. Computers & Security, 85, 386–401. https://doi.org/10.1016/j.cose.2019.05.003
6. Drone Hacking: Exploitation and Vulnerabilities. Available at: https://spadok.org.ua/books/Drone+Hacking-+Exploitation+and+Vulnerabilities.pdf
7. Throwback Attack: Keylogging virus infects U.S. drone fleet (2022). Control Engineering. Available at: https://www.controleng.com/throwback-attack-keylogging-virus-infects-u-s-drone-fleet/
8. Zakharchenko, I., Tristan, A., Chornogor, N., Berdnik, P., Kalashnyk, G., Timochko, A. et al. (2022). Modeling of Object Monitoring Using 3D Cellular Automata. Problems of the Regional Energetics, 4 (56), 61–73. https://doi.org/10.52254/1857-0070.2022.4-56.06
9. Song, Z., Zhang, H., Zhang, X., Zhang, F. (2019). Unmanned Aerial Vehicle Coverage Path Planning Algorithm Based on Cellular Automata. 2019 15th International Conference on Computational Intelligence and Security (CIS), 123–126. https://doi.org/10.1109/cis.2019.00034
10. Ioannidis, K., Sirakoulis, G. Ch., Andreadis, I. (2011). A path planning method based on cellular automata for cooperative robots. Applied Artificial Intelligence, 25 (8), 721–745. https://doi.org/10.1080/08839514.2011.606767
11. Adamatzky, A. I. (1996). Computation of shortest path in cellular automata. Mathematical and Computer Modelling, 23 (4), 105–113. https://doi.org/10.1016/0895-7177(96)00006-4
12. Behring, C., Bracho, M., Castro, M., Moreno, J. A. (2001). An Algorithm for Robot Path Planning with Cellular Automata. Theory and Practical Issues on Cellular Automata, 11–19. https://doi.org/10.1007/978-1-4471-0709-5_2
13. Xie, J., Chen, J. (2022). Multiregional Coverage Path Planning for Multiple Energy Constrained UAVs. IEEE Transactions on Intelligent Transportation Systems, 23 (10), 17366–17381. https://doi.org/10.1109/tits.2022.3160402
14. Fedorova, A., Beliautsou, V., Zimmermann, A. (2022). Colored Petri Net Modelling and Evaluation of Drone Inspection Methods for Distribution Networks. Sensors, 22 (9), 3418. https://doi.org/10.3390/s22093418
15. Xu, D., Borse, P., Altenburg, K., Nygard, K. (2006). A Petri Net Simulator for Self-organizing Systems. Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases. Madrid, 31–35. Available at: https://www.researchgate.net/publication/234805596

16. Gonçalves, P., Sobral, J., Ferreira, L. A. (2017). Unmanned aerial vehicle safety assessment modelling through petri Nets. Reliability Engineering & System Safety, 167, 383–393. https://doi.org/10.1016/j.ress.2017.06.021
17. Wang, X., Guo, Y., Lu, N., He, P. (2023). UAV Cluster Behavior Modeling Based on Spatial-Temporal Hybrid Petri Net. Applied Sciences, 13 (2), 762. https://doi.org/10.3390/app13020762
18. NS-3. Network Simulator. Available at: https://www.nsnam.org/
19. OMNeT++. Available at: https://omnetpp.org/
20. GAMA Platform. Available at: https://gama-platform.org/
21. Nikolaiev, M., Novotarskyi, M. (2024). Comparative Review of Drone Simulators. Information, Computing and Intelligent Systems, 4, 79–98. https://doi.org/10.20535/2786-8729.4.2024.300614
22. GPS Jamming, Spoofing and Hacking (2025). NorthStandard. Available at: https://north-standard.com/insights-and-resources/resources/articles/gps-jamming-spoofing-and-hacking
23. Stopochkina, I., Novikov, O., Voitsekhovskyi, A., Ilin, M., Ovcharuk, M. (2025). Simulation of UAV networks on the battlefield, taking into account cyber- physical influences that affect availability. Theoretical and Applied Cybersecurity, 6 (2). https://doi.org/10.20535/tacs.2664-29132024.2.318182
24. Pro zviazok vid Serhiya Flesh. Available at: https://t.me/s/serhii_flash
25. McNabb, M (2024). What Kind of Drones is Ukraine Buying? DroneLife. Available at: https://dronelife.com/2024/11/04/what-kind-of-drones-is-ukraine-buying/
26. Code and data for the paper: «Simulation of Unmanned Aerial Vehicles Networks under Electronic Warfare and Malware Propagation Conditions». Available at: https://github.com/KryvavyiPotii/drone-network