

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ
ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО» МІНІСТЕРСТВО
ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ
ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО» МІНІСТЕРСТВО
ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

МЕЛЕНЧУКОВ МИКИТА ЄВГЕНОВИЧ

УДК 004.7

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ТА БЕЗПЕКИ
РОЗПОДІЛЕНИХ СИСТЕМ

123 Комп'ютерна інженерія

12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело

Меленчуков М. Є.

Науковий керівник: Волокита Артем Миколайович, к. т. н., доцент

Київ – 2026

АНОТАЦІЯ

Меленчуков М.Є. Методи та засоби підвищення ефективності та безпеки розподілених систем. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – Комп'ютерна інженерія з галузі знань 12 – Інформаційні технології. – Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», Київ, 2026.

Дисертаційна робота присвячена вирішенню актуальної науково-прикладної задачі підвищення ефективності та безпеки функціонування розподілених обчислювальних систем в умовах інформаційної протидії, візантійських відмов, цілеспрямованих кібератак (зокрема ін'єкцій хибних даних, саботажу та атак типу on-off) і маніпуляції інформаційними потоками з боку скомпрометованих вузлів. За результатами дослідження створено комплекс методів та засобів, що забезпечують підвищення ефективності прийняття колективних рішень та безпеки управління в децентралізованому середовищі шляхом інтеграції методів глибокого навчання (зокрема, рекурентних нейронних мереж з довгою короткочасною пам'яттю), алгоритмів досягнення консенсусу та оптимізованої зваженої кластеризації. Верифікацію та підтвердження отриманих наукових результатів виконано на прикладі високодинамічного класу мобільних розподілених систем – роїв безпілотних літальних апаратів, що функціонують у середовищі літаючих спеціальних мереж.

Стрімке зміщення парадигми управління у високодинамічних кіберфізичних системах (від автономних роїв безпілотних літальних апаратів до критичної сенсорної інфраструктури) у бік децентралізованого прийняття рішень надає особливої актуальності пошуку нових методів координації в умовах обмеженої довіри. Попри розвиток протоколів візантійського консенсусу, існуючі рішення часто демонструють критичну вразливість до специфічних загроз – маніпуляцій репутацією та ефекту «миготіння» вузлів, що робить актуальним перегляд класичних моделей захисту. Необхідність забезпечення безперервності управління в реальному часі диктує вимогу відмови від статичних параметрів оцінки вузлів на

користь нелінійних механізмів адаптації ваги довіри. Саме тому актуальність роботи полягає у подоланні розриву між топологічною ефективністю мережі та її стійкістю до скоординованих внутрішніх атак, що є вирішальним фактором для надійного функціонування сучасних розподілених систем за умов високої невизначеності та динамічності середовища.

Метою дисертаційної роботи є підвищення безпеки та ефективності функціонування розподілених систем із динамічною топологією мережі та обмеженими ресурсами обміну даними шляхом розробки методів і засобів колективного прийняття рішень і координації, зокрема для мультиагентних систем.

Об'єктом дослідження є процеси колективного прийняття рішень і координації у розподілених системах з динамічною топологією зв'язків і обмеженими ресурсами обміну даними (зокрема у бездротових мережах), при наявності ненадійних або зловмисних вузлів. При цьому предметом дослідження виступають методи та засоби підвищення безпеки й ефективності зазначених процесів. Методичною основою роботи є системний аналіз і формалізація моделей розподіленої взаємодії, побудова математичних описів динамічної довіри, карантину з гістерезисом, зваженого кворуму та адаптивного порога; аналітична оцінка їх коректності та обчислювальної складності.

Досліджені відомі методи керування групами безпілотних апаратів та розподіленими обчислювальними системами, що характеризуються динамічною зміною топології та функціонують у децентралізованому середовищі. На основі аналізу літературних джерел зроблено постановку задачі про необхідність розробки нових методів та засобів для підвищення ефективності та безпеки, оскільки встановлено критичне протиріччя між вимогами до оперативності прийняття рішень в розподілених системах та необхідністю захисту від внутрішніх загроз. Зроблено висновок про недостатню ефективність класичних детермінованих алгоритмів (зокрема алгоритму зваженої кластеризації на основі довіри) та низьку безпеку статичних моделей оцінювання в умовах динамічних атак типу *on-off*.

Вперше розроблено та обґрунтовано математичну модель оцінювання довіри в розподілених системах, яка є основою для підвищення безпеки функціонування системи. На відміну від існуючих адитивних моделей, вона формалізує простір станів агентів та модель загроз спотворення змісту даних (зокрема, ін'єкцію хибних цілей), враховуючи нелінійні часові залежності поведінки вузлів. Використання рекурентних нейронних мереж дозволяє виявляти приховані аномалії, забезпечуючи підвищення безпеки за рахунок прогнозної здатності без зниження обчислювальної ефективності.

Удосконалено метод узгодження рішень із динамічною оцінкою довіри для підвищення ефективності та безпеки прийняття колективних рішень у розподілених системах. Ключовою особливістю методу є застосування механізму онлайн-донавчання для розрахунку адаптивного порогу (θ). На відміну від методів із фіксованим порогом, запропонований метод автоматично підлаштовує чутливість системи під рівень невизначеності середовища (у діапазоні від 0,46 до 0,92). Це дозволило на локальному рівні підвищити точність класифікації даних на 4,3% та повноту виявлення валідних подій на 7,8%, що безпосередньо впливає на підвищення ефективності функціонування системи.

Удосконалено метод вибору лідера для підвищення безпеки управління та топологічної ефективності децентралізованих мереж. Новизна методу полягає в інтеграції динамічного коефіцієнта довіри як змінної експоненти ($Trust^{2\theta}$) у функцію корисності. На відміну від традиційних моделей, такий підхід дозволяє реалізувати механізм «м'якої ізоляції» підозрілих вузлів, що підвищує безпеку системи, запобігаючи перехопленню керування скомпрометованими учасниками, та зберігає ефективність мережевої взаємодії.

Розроблено комплексний програмний полігон як засіб для підвищення ефективності та безпеки розподілених систем шляхом моделювання та верифікації алгоритмів керування у розподілених системах. Програмну реалізацію виконано на базі передового технологічного стеку ROS 2 та симулятора Gazebo. Застосовано гібридну архітектуру розробки: низькорівневе керування переміщеннями агентів винесено у закриті C++ плагіни для дотримання суворих часових рамок фізичного

симулятора, а високорівнева логіка консенсусу, машинного навчання та оцінки довіри реалізована на базі Python. Створено аналітичний інструментальний засіб для проведення ретроспективного моделювання, що дозволило відокремити логіку алгоритмів від недетермінованих мережових затримок симулятора. Архітектура оптимізована для периферійних вузлів з обмеженими ресурсами – споживання оперативної пам'яті процесом моделювання доведено до мінімуму (до 2–4 ГБ), що гарантує придатність для використання у реальних бортових комплексах.

Комплексне експериментальне дослідження, що включало моделювання функціонування рою агентів в серії з 20 унікальних сценаріїв, які відрізнялися складністю топології, протяжністю маршрутів та інтенсивністю атак хибними даними, переконливо довело досягнення мети роботи щодо суттєвого підвищення загальної ефективності та безпеки системи. Інтеграція адаптивних локальних оцінок у процес глобального узгодження (консенсусу) забезпечила зростання середньої точності фізичного виконання групових завдань на 8% (до рівня 0,81). Частка непродуктивних витрат ресурсів, пов'язаних із відвідуванням хибних цілей, зменшилася на 8% в абсолютному вимірі. Загальний показник функціональної стабільності розподіленої системи (F1-Score) підвищився на 5,9%. За результатами симуляції, метод вибору лідера продемонстрував зниження частоти обрання скомпрометованих вузлів-лідерів на 11,6% (з 6,0% до 5,3%) порівняно з еталонними методами; при цьому загальний коефіцієнт топологічної ефективності зріс на 0,8% (до 0,943), а середні втрати топології безперервно знизилися на 17,4%.

Розроблений комплекс методів та відповідних інструментальних засобів дозволив створити експериментальну модель адаптивної системи керування, що забезпечує підвищення ефективності та безпеки функціонування автономних розподілених систем в умовах деструктивного інформаційного впливу.

Ключові слова: розподілені системи, ефективність розподілених систем, безпека розподілених систем, штучна нейронна мережа, виявлення аномалій, багатокритеріальне прийняття рішень, LSTM, автономна навігація рою, БПЛА, оптимізація, децентралізовані системи, прийняття управлінських рішень, алгоритми консенсусу

ABSTRACT

Melenchukov M.Y. Methods and tools for increasing the efficiency and security of distributed systems. – Qualifying scientific work on the rights of a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 123 – Computer Engineering in the field of knowledge 12 – Information Technologies. – National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, 2026.

The dissertation is devoted to solving the relevant scientific and applied task of increasing the efficiency and security of the functioning of distributed computing systems under conditions of information opposition, Byzantine failures, targeted cyberattacks (in particular, false data injections, sabotage, and on-off attacks) and manipulation of information flows by compromised nodes. Based on the results of the research, a complex of methods and tools was created that ensure an increase in the efficiency of collective decision-making and control security in a decentralized environment by integrating deep learning methods (in particular, long short-term memory recurrent neural networks), consensus algorithms, and optimized weighted clustering. Verification and confirmation of the obtained scientific results were performed on the example of a highly dynamic class of mobile distributed systems - swarms of unmanned aerial vehicles functioning in the environment of flying ad-hoc networks.

The rapid shift of the control paradigm in highly dynamic cyber-physical systems (from autonomous swarms of unmanned aerial vehicles to critical sensor infrastructure) towards decentralized decision-making makes the search for new coordination methods under conditions of limited trust particularly relevant. Despite the development of Byzantine consensus protocols, existing solutions often demonstrate critical vulnerability to specific threats – reputation manipulations and the “flashing” effect of nodes, making it relevant to review classical protection models. The need to ensure continuous control in real time dictates the requirement to abandon static node evaluation parameters in favor of nonlinear adaptive trust weight mechanisms. That is why the relevance of the work lies in overcoming the gap between the topological efficiency of the network and its resilience to coordinated internal attacks, which is a decisive factor for the reliable

functioning of modern distributed systems under conditions of high uncertainty and dynamic environment.

The aim of the dissertation is to develop methods and tools for increasing the efficiency and security of distributed systems, which will allow guaranteeing resilience to intelligent attacks and data manipulation through dynamic trust evaluation between nodes in real time.

The object of research is the processes of collective decision-making and coordination in distributed systems with varying communication topology and limited data exchange resources (in particular in wireless networks), in the presence of unreliable or malicious nodes. At the same time, the subject of research is the methods and tools for increasing the security and efficiency of these processes. The methodological basis of the work is system analysis and formalization of distributed interaction models, construction of mathematical descriptions of dynamic trust, hysteresis quarantine, weighted quorum, and adaptive threshold; analytical evaluation of their correctness and computational complexity.

Known methods of controlling groups of unmanned vehicles and distributed computing systems, which are characterized by dynamic topology changes and function in a decentralized environment, were investigated. Based on the analysis of literature sources, a problem statement was made regarding the need to develop new methods and tools to increase efficiency and security, since a critical contradiction was established between the requirements for prompt decision-making in distributed systems and the urgent need for protection against internal threats. It was concluded that classical deterministic algorithms (in particular, trust-based weighted clustering) lack efficiency and static evaluation models have low security in the context of dynamic on-off attacks.

For the first time, a mathematical model for trust evaluation in distributed systems was developed and substantiated, which is the basis for increasing the security of system operation. Unlike existing additive models, it formalizes the state space of agents and the threat model of data content distortion (in particular, the injection of false targets), taking into account the nonlinear time relationships of nodes' behavior. The use of recurrent neural networks allows the detection of hidden anomalies, ensuring increased security

through predictive capability without reducing computational efficiency.

The method of decision consensus with dynamic trust evaluation was improved to increase the efficiency and security of collective decision-making in distributed systems. A key feature of the method is the application of an online fine-tuning mechanism for calculating the adaptive threshold (θ). Unlike fixed-threshold methods, the proposed method automatically adjusts the sensitivity of the system to the level of environmental uncertainty (in the range from 0.46 to 0.92). This allowed increasing local data classification accuracy by 4.3% and valid event detection recall by 7.8% at the local level, which directly affects the increase of the system functioning efficiency.

The leader election method was improved to increase the control security and topological efficiency of decentralized networks. The novelty of the method consists in the integration of a dynamic trust coefficient as a variable exponent ($Trust^{2\theta}$) into the utility function. In contrast to traditional models, this approach allows implementing a “soft isolation” mechanism for suspicious nodes, which increases system security by preventing control hijacking by compromised participants, while maintaining the efficiency of network interaction.

A comprehensive software testbed was developed as a tool to increase the efficiency and security of distributed systems by modeling and verifying control algorithms in distributed systems. Software implementation was carried out on the basis of the advanced ROS 2 technology stack and the Gazebo simulator. A hybrid development architecture was applied: low-level control of agent movements was transferred to closed C++ plugins to comply with the strict time limits of the physical simulator, and the high-level logic of consensus, machine learning, and trust evaluation was implemented on the basis of Python. An analytical tool for conducting trace-driven simulation was created, allowing the separation of algorithm logic from non-deterministic network delays of the simulator. The architecture is optimized for peripheral nodes with limited resources – the RAM consumption of the simulation process is minimized (to 2–4 GB), guaranteeing suitability for use in real onboard complexes.

Comprehensive experimental research, which included modeling the functioning of an agent swarm in a series of 20 unique scenarios varying in topology complexity, route

length, and false data attack intensity, convincingly proved the achievement of the work's objective of significantly increasing overall system efficiency and security. The integration of adaptive local estimates into the global consensus process ensured an 8% increase in the average physical execution precision of group tasks (Visit Precision) (to a level of 0.81). The share of unproductive resource expenditures associated with visiting false targets decreased by 8% in absolute terms. Overall distributed system reliability (F1-Score) increased by 5.9%. According to the simulation results, the leader election method demonstrated an 11.6% decrease in the frequency of electing compromised leader nodes (from 6.0% to 5.3%) compared to reference methods; wherein the total topological efficiency coefficient increased by 0.8% (to 0.943), and the average topology loss continuously decreased by 17.4%.

The developed complex of methods and corresponding tools allowed creating an experimental model of an adaptive control system, which ensures increased efficiency and security of autonomous distributed system functioning under conditions of destructive information influence.

Keywords: distributed systems, efficiency of distributed systems, security of distributed systems, artificial neural network, anomaly detection, multi-criteria decision making, LSTM, autonomous swarm navigation, UAV, optimization, decentralized systems, management decision making, consensus algorithms.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Меленчуков М. Є., Волокита. А. М., «Нейронні мережі в виявленні атак на розподілені системи,» *Технічні науки та технології*. 2024. 1, №35, сс. 135-145, DOI: 10.25140/2411-5363-2024-1(35)-135-145 (Фахове наукове видання України категорії Б). Здобувач здійснив аналітичну систематизацію архітектур нейронних мереж для забезпечення безпеки розподілених систем та визначив критерії ефективності інтелектуальних методів ідентифікації аномальної активності; Волокита А. М. здійснив загальне наукове керівництво, визначив пріоритетні напрями дослідження методів машинного навчання у розподілених системах та верифікував результати аналізу.
2. Меленчуков М. Є., Волокита. А. М., «Дослідження моделей виявлення атак на розподілені системи за допомогою згорткових нейронних мереж,» *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2024. №3, сс. 224-229, DOI: 10.31891/2219-9365-2024-79-29 (Фахове наукове видання України категорії Б). Здобувач розробив та провів порівняльний аналіз 9 конфігурацій згорткових нейронних мереж для захисту розподілених систем, реалізував препроцесинг та балансування вибірок реальних наборів даних кібератак, а також експериментально обґрунтував вибір оптимальних гіперпараметрів для максимізації точності детекції; Волокита А. М. формалізував процес вибору архітектури нейромережі для забезпечення безпеки розподілених систем.
3. Меленчуков М. Є., Волокита. А. М., «Метод вибору лідера для підвищення ефективності комунікації в рої БПЛА,» *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025. №3, сс. 409-414, DOI: 10.31891/2219-9365-2025-83-50 (Фахове наукове видання України категорії Б). Здобувач удосконалив адаптивний метод вибору лідера на основі модифікованого алгоритму зваженої кластеризації з лінійною

обчислювальну складність $O(N)$, що забезпечило підвищення комунікаційної ефективності розподілених систем; Волокита А. М. запропонував концептуальну модель ієрархічного управління та децентралізованої координації в динамічних розподілених системах.

4. Меленчуков М. Є., Волокита. А. М., «Адаптивний метод вибору лідера в мобільних розподілених системах на основі інтеграції динамічної довіри,» *Проблеми інформатизації та управління*. 2026. 1, №85, сс. 35-41, DOI: 10.18372/2073-4751.85.21091 (Фахове наукове видання України категорії Б). Здобувач формалізував інтеграцію динамічної довіри в алгоритм вибору лідера та через *Trace-Driven Simulation* підтвердив підвищення безпеки розподілених систем; Волокита А. М. здійснив наукове консультування щодо синтезу механізмів безпеки та ефективності.
5. Меленчуков М. Є., Волокита. А. М., «Метод динамічної оцінки довіри за допомогою модифікованого гістерезису у роях БПЛА,» *Телекомунікаційні та інформаційні технології*. 2026. 1, №90, сс. 218-224, DOI: 10.31673/2412-4338.2026.019021 (Фахове наукове видання України категорії Б). Здобувач обґрунтував метод динамічного оцінювання довіри під час виконання місії рою дронів, реалізувавши механізм безперервного оновлення коефіцієнта довіри; запропонував коефіцієнт динамічного порогу довіри у вигляді модуля модифікованого гістерезису, який забезпечує постійну регуляцію ваги репутації вузлів розподіленої системи, що дозволяє адаптивно виявляти та нейтралізувати вплив підозрілих учасників; Волокита А. М. здійснив наукове керівництво щодо синтезу механізмів безпеки та валідації результатів.

Апробація наукових результатів дисертації:

6. Melenchukov M., Volokyta A. Analysis of defense and attack methods in distributed systems. The International Conference on Security, Fault Tolerance, Intelligence, (Kyiv, June 28, 2024). Kyiv, 2024. URL: <https://icsfti-proc.kpi.ua/issue/view/16681> (дата звернення: 20.03.2026). Здобувач провів

класифікацію методів захисту та аналіз вразливостей вузлів у розподілених системах та систематизував вектори деструктивного впливу кіберзагроз на їх відмовостійкість; Волокита А. М. визначив вектори впливу кіберзагроз на показники безпеки розподілених систем.

7. Melenchukov M. Improving safety and efficiency of distributed systems: a predictive navigation model for uav swarms. The International Conference on Security, Fault Tolerance, Intelligence, (Kyiv, June 30, 2025). Kyiv, 2025. URL: <https://icsfti-proc.kpi.ua/article/view/354612> (дата звернення: 20.03.2026).
Здобувач формалізував децентралізовану архітектуру управління, що інтегрує ймовірнісне 3D-картографування (OctoMap) та предиктивне алгоритмічне планування руху, та експериментально підтвердив підвищення ефективності функціонування системи при одночасному зниженні частоти структурних колізій.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	15
ВСТУП	16
РОЗДІЛ 1 АНАЛІЗ МЕХАНІЗМІВ ЕФЕКТИВНОСТІ ТА БЕЗПЕКИ РОЗПОДІЛЕНИХ СИСТЕМ	24
1.1. Проблеми ефективності та узгодженості у динамічних розподілених системах	24
1.2. Аналіз проблем безпеки та стійкості у розподілених середовищах.....	28
1.2.1. Ландшафт загроз та модель порушника.....	28
1.2.2. Проблема довіри	37
1.2.3 Задача досягнення консенсусу	43
1.3. Систематизація задач самоорганізації та керування.....	48
1.4. Аналіз методів вибору лідера	56
1.4.1. Роль лідера в архітектурі розподіленого керування	56
1.4.2. Алгоритми зваженої кластеризації на основі довіри (Trust WCA)	59
1.4.3. Безпечні модифікації класичних алгоритмів (Secure Bully/Ring)	60
1.4.4. Блокчейн-орієнтовані підходи та консенсусні вибори	62
1.4.5. Біо-інспіровані алгоритми з феромонами довіри.....	64
1.4.6. Аналіз компромісу «Швидкість – Довіра» та проблем адаптивності	66
1.5. Методи координації у розподілених мережах класу FANET	68
1.5.1. Специфіка роїв як об'єкта дослідження	68
1.5.2. Чому стандартні рішення не працюють для дронів	70
Висновки до розділу 1	72
РОЗДІЛ 2 МЕТОДИ ДИНАМІЧНОЇ ОЦІНКИ ДОВІРИ І ВИБОРУ ЛІДЕРА У РОЗПОДІЛЕНИХ СИСТЕМАХ	74
2.1. Математична модель узгодженості рою за умов візантійських відмов.....	74
2.1.1. Модель системи та середовища	74
2.1.2. Модель загроз та маніпуляції даними	77
2.2 Метод динамічної довіри та стійкого консенсусу з нейромережевим підсиленням.....	78
2.2.1. Концепція та архітектура методу.....	78
2.2.2. Модель інформаційного обміну та структура повідомлень.....	80
2.2.3. Метрики узгодженості	84
2.2.4. Алгоритм розрахунку довіри та механізм карантину.....	87

2.2.5. Розрахунок динамічного коефіцієнта довіри через LSTM.....	94
2.3. Метод вибору лідера на основі інтегрованої довіри та нейромереж	98
2.3.1. Метрики ефективності комунікації	98
2.3.2. Формування комплексної вагової функції.....	103
2.3.3. Процедура виборів та стабільність кластера	108
2.4. Оцінка ресурсних витрат та стійкості	109
2.4.1. Аналіз обчислювальної складності.....	109
2.4.2. Аналіз комунікаційного навантаження	112
Висновки до розділу 2	114
РОЗДІЛ 3 ЗАСОБИ УЗГОДЖЕННЯ РІШЕНЬ ІЗ ДИНАМІЧНОЮ ДОВІРОЮ ТА ВИБОРУ ЛІДЕРА У РОЗПОДІЛЕНИХ СИСТЕМАХ	116
3.1. Вибір технологічного стеку та середовища моделювання.....	116
3.2 Структура засобу динамічної оцінки довіри та стійкого консенсусу	124
3.3 Структура засобу для перевірки методу вибору лідера.....	145
Висновки до розділу 3	150
РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ДОВІРИ ТА ВИБОРУ ЛІДЕРА	152
4.1. Методика досліджень та опис тестового стенду	152
4.2. Аналіз динаміки довіри та ефективності рішень агентів	157
4.3. Порівняльний аналіз алгоритмів вибору лідера на польотних даних	169
Висновки до розділу 4	177
ВИСНОВКИ	179
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	182
ДОДАТОК А Частина програмного коду	193
ДОДАТОК Б Список публікацій здобувача.....	236
ДОДАТОК В Теоретичні основи та комунікаційні обмеження сучасних розподілених систем	239
ДОДАТОК Г Типологія задач керування об'єктами у розподілених системах ...	243

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- BFT – Byzantine Fault Tolerance (Стійкість до візантійських відмов)
- CH – Cluster Head (Головний вузол кластера / Лідер кластера)
- EC – Edge Computing (Граничні обчислення)
- EWMA – Exponentially Weighted Moving Average (Експоненційне зважене ковзне середнє)
- FANET – Flying Ad-hoc Network (Літаюча спеціальна мережа / Літаюча ad-hoc мережа)
- FDI – False Data Injection (Ін'єкція хибних даних)
- IDS – Intrusion Detection System (Система виявлення вторгнень)
- IoT – Internet of Things (Інтернет речей)
- LSTM – Long Short-Term Memory (Довга короткочасна пам'ять / Рекурентна нейронна мережа з довгою короткочасною пам'яттю)
- MANET – Mobile Ad-hoc Network (Мобільна спеціальна мережа)
- PBFT – Practical Byzantine Fault Tolerance (Практичний алгоритм стійкості до візантійських відмов)
- PKI – Public Key Infrastructure (Інфраструктура відкритих ключів)
- QoS – Quality of Service (Якість обслуговування)
- RNN – Recurrent Neural Network (Рекурентна нейронна мережа)
- ROS – Robot Operating System (Операційна система для роботів)
- RTT – Round Trip Time (Час кругового обігу сигналу)
- SDN – Software Defined Networks (Програмно-визначені мережі)
- TDS – Trace-Driven Simulation (Ретроспективне моделювання)
- Trust-WCA – Trust-based Weighted Clustering Algorithm (Алгоритм зваженої кластеризації з урахуванням довіри)
- WCA – Weighted Clustering Algorithm (Алгоритм зваженої кластеризації)
- БПЛА – Безпілотний літальний апарат
- РЕБ – Радіоелектронна боротьба

ВСТУП

Обґрунтування вибору теми дослідження. Сучасні розподілені обчислювальні та кіберфізичні системи характеризуються зміщенням парадигми від централізованого управління до децентралізованого формування колективних рішень у реальному часі, що ґрунтовно підтверджується у комплексі сучасних досліджень [1], [2]. Ефективність таких систем визначається не лише продуктивністю окремих обчислювальних вузлів, а насамперед здатністю мережі швидко узгоджувати спільні дії за умов обмеженої пропускної здатності каналів зв'язку, варіативних затримок передавання даних, динамічної топології та неоднорідності ресурсів, , як зазначають автори робіт з розподіленої координації [3], [4].

Практичні сценарії розгортання розподілених мереж – від Інтернету речей (IoT) та сенсорних мереж моніторингу критичної інфраструктури до мультиагентних робототехнічних комплексів (зокрема, роїв безпілотних літальних апаратів – БПЛА) – вимагають від системи одночасно високої оперативності координації (швидкий розподіл навантаження, синхронізація станів, вибір координатора) і підвищеного рівня безпеки, що підкреслюється у систематичних оглядах вразливостей [5], [6]. Система повинна зберігати стійкість до збоїв і навмисних інформаційних впливів, забезпечуючи коректне функціонування навіть за наявності частково недостовірних або зашумлених даних, що формує окремий напрямок досліджень надійності бездротових сенсорних мереж [7], [8]. У статті [9] автори пропонують адаптивний ланцюговий протокол візантійського консенсусу. Це вирішує проблему узгодженості в умовах високої мобільності вузлів, проте не покриває проблему маніпуляцій репутацією в динамічних сценаріях. Водночас у праці [10] автори розглядають полегшений механізм асинхронного консенсусу. Це вирішує проблему завищених комунікаційних витрат, проте не покриває ситуації тривалого часткового погіршення зв'язку або виникнення ефекту «миготіння» вузлів.

Особливої ваги набуває функціонування розподілених систем в умовах

неповної довіри до учасників, коли частина вузлів може тимчасово втрачати зв'язок, працювати некоректно або бути скомпрометованою зловмисниками, що детально аналізується у дослідженнях розподілених атак [11]. У статті [12] автори досліджують управління довірою на базі технології блокчейн. Це вирішує проблему надійного збереження історії репутації, проте не покриває проблему надмірних обчислювальних і часових затримок для систем управління в реальному часі. Альтернативним підходом для зменшення затримок є оптимізація комунікаційних витрат при обміні даними, що розглядається у працях з оптимізації бездротових мереж [13]. Це вирішує проблему енергоефективності та маршрутизації, проте не охоплює питання захисту від ефекту «миготіння» зловмисних вузлів та забезпечення оцінки їх функціональної стабільності.

Водночас критичною вразливістю сучасних розподілених систем залишається їхня вразливість до скоординованих внутрішніх атак, що провокує гостру потребу в захищених моделях оцінювання довіри [14], [15]. Хоча існуючі механізми захисту частково нівелюють ці ризики для статичних або повільно змінних топологій, специфіка високодинамічних мереж (наприклад, роїв БПЛА) критично загострює цю проблему. Зокрема, класичні алгоритми візантійського консенсусу [16] демонструють математично доведене різке падіння швидкості збіжності при зростанні активності шкідливих агентів. Це робить застосування існуючих методів вкрай неефективним в умовах жорсткого реального часу без інноваційних превентивних механізмів динамічної ізоляції порушників.

Зазначені недоліки існуючих методів визначають необхідність розробки нових механізмів, тому для динамічних розподілених систем із жорсткими часовими обмеженнями залишаються відкритими два ключові питання. По-перше, як забезпечити ефективне прийняття рішень за наявності функціонально нестабільних або зловмисних учасників із дотриманням вимог до безперервності управління та жорстких обмежень реального часу. Хоча відомі дослідження розглядають окремі аспекти застосування машинного навчання та класичних метрик довіри в контексті кіберзагроз [14], [17], [18], у них не вирішено проблему забезпечення ефективного прийняття колективних рішень в умовах

невизначеності, зумовленої динамічною топологією та викривленням даних. Це зумовлює необхідність розроблення нового методу динамічної оцінки довіри та створення відповідних інструментальних засобів для його реалізації. По-друге, як забезпечити динамічний баланс між безпекою (довірою до лідера) та загальною ефективністю координації за умов мінливого рівня загроз. Більшість існуючих алгоритмів вибору лідера та кластеризації [19], [20], навіть розраховуючи коефіцієнти довіри динамічно, під час прийняття рішень агрегують їх з іншими параметрами вузлів використовуючи статичні або лінійні вагові коефіцієнти. Вони не здатні ситуативно адаптувати вагу (значущість) фактору довіри відносно інших критеріїв залежно від обставин: намагання заздалегідь надати безпеці максимально високої ваги для ізоляції підозрілих вузлів часто призводить до вибору топологічно неефективних лідерів, що суттєво погіршує продуктивність розподіленої системи, тоді як недостатня вага залишає систему вразливою. Це зумовлює потребу в розробленні нових методів вибору лідера з нелінійними механізмами, які дозволяють керувати саме динамічною вагою репутації та знаходити раціональний компроміс між безпекою і топологічною ефективністю без критичного падіння продуктивності системи. Для оцінки ефективності роботи методу вибору лідера в умовах деструктивних впливів традиційні показники апаратної надійності (наприклад, середній час наробітку на відмову, який вимірюється в годинах) не є репрезентативними. Тому в роботі введено та використано цільовий показник функціональної стабільності лідера.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота входить в план наукової роботи кафедри обчислювальної техніки КПІ ім. Ігоря Сікорського і виконана в рамках наступних пошукових досліджень (ініціативних тематик): «Високопродуктивні комп'ютерні системи та мережі: теорія, методи і засоби апаратної та програмної реалізації» (факультет інформатики та обчислювальної техніки – керівник: доц. А. М. Волокита), № договору: Д/р №0121U108261, дата реєстрації: 11.02.2021; «Розроблення моделей та методів функціонування відмовостійких кіберфізичних розподілених систем з динамічною структурою» (факультет інформатики та обчислювальної техніки –

керівник: професор М. А. Новотарський), № договору: Д/р №0126U001999, дата реєстрації: 15.02.2026.

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення безпеки та ефективності функціонування розподілених систем із динамічною топологією мережі та обмеженими ресурсами обміну даними шляхом розробки методів і засобів колективного прийняття рішень і координації, зокрема для мультиагентних систем.

Для досягнення мети необхідно розв'язати такі завдання:

1. Проаналізувати існуючі методи керування розподіленими системами, виявити протиріччя між вимогами до оперативності прийняття рішень та необхідністю захисту від інтелектуальних атак і обґрунтувати необхідність розробки нових адаптивних підходів.
2. Сформулювати систему припущень і математичну модель оцінювання довіри в розподіленому середовищі з урахуванням змінної зв'язності, асинхронності взаємодії, втрат/затримок повідомлень, нелінійних часових залежностей поведінки та можливих відхилень окремих вузлів для виявлення прихованих аномалій та стратегічних атак.
3. Розробити метод динамічної оцінки довіри та узгодження рішень, запропонувавши правило агрегування багатофакторних ознак поведінки вузлів і механізм автоматичної адаптації порогів прийняття рішень (потрібного рівня довіри) для забезпечення балансу між безпекою та ефективністю в умовах невизначеності середовища.
4. Удосконалити метод вибору лідера шляхом інтеграції оцінки динамічної довіри до загальних критеріїв ранжування кандидатів для забезпечення ізоляції підозрілих вузлів та зниження ризику перехоплення керування в умовах мінливого рівня загроз.
5. Створити програмні засоби, що реалізують обидва методи, підтримують сценарне відтворення експериментів, збір журналів і метрик, а також дозволяють порівнювати конфігурації без зміни інших компонентів системи.

6. Визначити показники оцінювання та методику експериментів для обох методів; провести порівняльне оцінювання їх ефективності, виконати аналіз обчислювальної та комунікаційної складності; сформулювати практичні рекомендації щодо налаштування параметрів, меж застосовності та відтворюваності результатів у різних сценаріях розподілених систем.

Об'єкт дослідження. Процеси колективного прийняття рішень і координації у розподілених системах з динамічною топологією зв'язків і обмеженими ресурсами обміну даними (зокрема у бездротових мережах), при наявності функціонально нестабільних або зловмисних вузлів.

Предмет дослідження. Методи та засоби підвищення безпеки й ефективності процесів колективного прийняття рішень і координації у розподілених системах з динамічною топологією зв'язків і обмеженими ресурсами обміну даними (зокрема у бездротових мережах), при наявності функціонально нестабільних або зловмисних вузлів.

Методи дослідження. Методичною основою роботи є системний аналіз і формалізація моделей розподіленої взаємодії, побудова математичних описів динамічної довіри, карантину з гістерезисом, зваженого кворуму та адаптивного порога; аналітична оцінка їх коректності та обчислювальної складності.

Експериментальна перевірка виконувалась у симуляційному середовищі *Gazebo* інтегрованому з *ROS2* із власними вузлами обміну повідомленнями й журналюванням, із подальшою статистичною обробкою результатів (точність, повнота, *F1*, частка хибних рішень, частка прийнятих рішень) і порівнянням із базовими варіантами. Для перевірки узгодженості висновків застосовувалися методи порівняльного аналізу, оцінювання чутливості до параметрів згладжування та перевірка відтворюваності за фіксованими конфігураціями запусків.

Наукова новизна отриманих результатів. Наукова новизна одержаних результатів роботи полягає у наступному:

1. Вперше розроблено та обґрунтовано метод динамічного оцінювання довіри до вузлів розподілених систем, який, на відміну від існуючих методів, одночасно враховує узгодженість поточного стану вузла з очікуваною динамікою його стану,

узгодженість запропонованих вузлом дій із пропозиціями інших вузлів та безперервне оновлення коефіцієнтів довіри, що дозволяє адаптивно виявляти та нейтралізувати вплив підозрілих вузлів розподіленої системи в умовах динамічних змін середовища, підвищуючи безпеку системи;

2. Вперше запропоновано метод визначення відносної ваги лідера розподіленої системи із застосуванням мультиплікативного коефіцієнта на основі динамічного порогу довіри вузлів, який, на відміну від існуючих механізмів керування, за рахунок постійної регуляції нейромережею зазначеного порогу, дозволяє підвищити функціональну стабільність лідера та ефективність комунікації у розподіленій системі;

3. Удосконалено метод вибору координатора (лідера) у розподілених системах, який, на відміну від існуючих методів, використовує нелінійну вагову функцію, в якій динамічний параметр виступає експонентою до коефіцієнта довіри, що дозволяє ситуативно посилювати вагу репутації під час взаємодії вузлів розподіленої системи та забезпечувати ізоляцію підозрілих вузлів без порушення загальної координації.

Практичне значення отриманих результатів. Запропоновані методи та одержані результати дозволяють реалізувати на практиці комплексне рішення для організації самоорганізованих розподілених обчислювальних систем та мереж Інтернету речей (*IoT*), що функціонують в умовах невизначеності. Запропоновані методи реалізовані у вигляді програмних рішень, що представляє собою модулі інтелектуального управління для вузлів розподіленої мережі. Рішення включають підсистему динамічної оцінки довіри на базі рекурентної нейронної мережі (*LSTM*) з підтримкою механізму постійного навчання (*online learning*) та алгоритм адаптивного призначення координатора (лідера) кластера.

Розроблені рішення у вигляді інструментальних засобів дозволяють:

1. Досягнуто підвищення точності узгодження рішень та адаптивності системи до динамічних змін середовища за рахунок використання розробленого методу динамічної оцінки довіри з автоматичним регулюванням порогів чутливості;

2. Досягнуто підвищення безпеки та ефективності самоорганізації мережі за рахунок удосконаленого методу вибору координатора, який забезпечує ізоляцію підозрілих вузлів без порушення загальної працездатності системи;
3. Досягнуто можливості ефективного функціонування комплексних алгоритмів захисту на вузлах з обмеженими апаратними ресурсами за рахунок розробленої гібридної програмної архітектури.

Результати роботи можуть бути застосовані при розробці систем управління роями автономних дронів (як репрезентативного прикладу розподілених систем, що діють в умовах високої динаміки та невизначеності), а також у системах цивільного та спеціального призначення, індустріальних мережах (*IIoT*) та сенсорних мережах розумних міст, де критично важливими є швидка координація, стійкість до відмов та здатність працювати в умовах часткової недостовірності даних.

Публікації.

За результатами дослідження опубліковано:

- 5 статей у фахових виданнях України (категорія Б);
- 2 тези доповідей на міжнародних і всеукраїнських конференціях.

Особистий внесок здобувача. Дисертаційна робота є самостійно виконаним науковим дослідженням. Усі результати, що викладені в дисертації та виносяться на захист, отримані автором особисто. Конкретний особистий внесок здобувача полягає у такому: [21] – здійснив аналітичну систематизацію архітектур нейронних мереж для забезпечення безпеки розподілених систем та визначив критерії ефективності інтелектуальних методів ідентифікації аномальної активності; [22] – розробив та провів порівняльний аналіз конфігурацій згорткових нейронних мереж для захисту розподілених систем, реалізував препроцесинг та балансування вибірок реальних наборів даних кібератак, а також експериментально обґрунтував вибір оптимальних гіперпараметрів для максимізації точності детекції; [23] – удосконалив адаптивний метод вибору лідера на основі модифікованого алгоритму зваженої кластеризації з лінійною обчислювальною складністю $O(N)$,

що забезпечило підвищення комунікаційної ефективності розподілених систем; [24] – формалізував механізм інтеграції коефіцієнта динамічної довіри в алгоритм вибору лідера та шляхом Trace-Driven Simulation підтвердив підвищення безпеки розподілених систем через зниження ймовірності перехоплення керування вузлів; [25] – обґрунтував метод динамічного оцінювання довіри та ввів коефіцієнт динамічного порогу довіри на базі математичного апарату модифікованого гістерезису, що шляхом ізольованого імітаційного моделювання підтвердило повне усунення ефекту «миготіння» вузлів; [11] – провів класифікацію методів захисту та аналіз вразливостей вузлів у розподілених системах та систематизував вектори деструктивного впливу кіберзагроз на їх відмовостійкість; [26] – формалізував децентралізовану архітектуру управління, що інтегрує ймовірнісне 3D-картографування та предиктивне алгоритмічне планування руху, та експериментально підтвердив підвищення ефективності функціонування системи при одночасному зниженні частоти структурних колізій.

Апробація результатів дисертації. Основні результати роботи опубліковано та обговорено на всеукраїнських та міжнародних конференціях, зокрема:

1. Melenchukov M., Volokyta A. Analysis of defense and attack methods in distributed systems. // The International Conference on Security, Fault Tolerance, Intelligence (ICSFTI-2024), Kyiv, Ukraine, June 28, 2024. [Online]. Available: <https://icsfti-proc.kpi.ua/issue/view/16681>
2. Melenchukov M. Improving safety and efficiency of distributed systems: a predictive navigation model for uav swarms. // The International Conference on Security, Fault Tolerance, Intelligence (ICSFTI-2025), Kyiv, Ukraine, June 30, 2025. [Online]. Available: <https://icsfti-proc.kpi.ua/article/view/354612>

Структура і обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел із 92 найменувань. Загальний обсяг дисертації становить 248 сторінок, з яких 181 сторінки основного тексту, 4 додатки на 56 сторінках. Дисертація містить 30 рисунків, 87 формул, 11 таблиць.

РОЗДІЛ 1

АНАЛІЗ МЕХАНІЗМІВ ЕФЕКТИВНОСТІ ТА БЕЗПЕКИ РОЗПОДІЛЕНИХ СИСТЕМ

1.1. Проблеми ефективності та узгодженості у динамічних розподілених системах

Стрімкий розвиток інформаційно-керуючих комплексів та широкомасштабних кіберфізичних систем супроводжується фундаментальним переходом від жорстко централізованих архітектур до децентралізованих розподілених мереж. Як підкреслюють автори фундаментального курсу [27], хоча відмова від єдиного центру управління й суттєво підвищує стійкість системи в цілому, вона водночас ініціює нові багаторівневі виклики (базові принципи яких детально закладені ще у класичній теорії реального часу [1]), що безпосередньо пов'язані з відсутністю глобального годинника, асинхронністю взаємодії вузлів та обмеженістю доступних комунікаційних ресурсів. Крім того, у глобальному звіті [28] зазначається, що зі зростанням масштабів мереж обміну даними між автономними пристроями ключовою проблемою стає зростаючий розрив між високою щільністю цифрових потоків і реальною пропускнуою здатністю бездротових каналів зв'язку. Детальне теоретичне обґрунтування базових принципів, класифікації та комунікаційних обмежень, характерних для сучасних розподілених систем, винесено у Додаток В.

Яскравим концептуальним прикладом такої еволюції є розвиток високодинамічних мереж на базі рухомих мультиагентних комплексів. Як показано в дослідженні [29], архітектура керування подібними системами (зокрема, роями безпілотних літальних апаратів) невпинно зміщується від жорсткої ієрархії до повної децентралізації, формуючи концепцію Flying Ad-Hoc Networks (FANET). Як детально обґрунтовано в оглядовій роботі [30], використання традиційної централізованої архітектури, де всі потоки даних замикаються на єдиний вузол (наземну станцію керування – GCS, або хмарний сервер), можливо і вирішувало завдання простоти координації, проте виявилось

критично вразливим в умовах високої динаміки та активного ворожого середовища.

Обмеження централізованої моделі детально досліджуються одразу в декількох фундаментальних працях. Зокрема, у статті [31] автори виділяють наступні критичні проблеми:

1. Єдина точка відмови (*Single Point of Failure*): Втрата каналу зв'язку з центром (внаслідок дії РЕБ або виходу за межі радіогоризонту) призводить до повної втрати боєздатності групи.
2. Затримка транзиту (*Round Trip Time*): Як додатково зазначається у згаданому огляді комунікаційних проблем [29], необхідність передачі даних від сенсора на землю, прийняття рішення оператором і передачі команди назад створює затримку, неприпустиму для високодинамічних тактичних ситуацій.
3. Масштабованість: Пропускна здатність каналу «земля-повітря» швидко вичерпується при збільшенні кількості агентів у рої.

Перелічені дослідження глибоко розкривають фізичні та комунікаційні бар'єри централізованого підходу, що однозначно доводить необхідність переходу до розподілених парадигм самоорганізації. Відповідно, у статті [2] автори обґрунтовують доцільність переходу до архітектури Mesh-мереж (рис. 1.1) та застосування концепції периферійних обчислень (*Edge Computing*) як відповіді на ці виклики. У їхній моделі принципово вирішено проблему перевантаження магістральних каналів «земля-повітря» шляхом перенесення обробки даних та прийняття тактичних рішень безпосередньо на борт БПЛА (*On-board processing*). Цей підхід дозволяє реалізувати принцип локальності: дрони взаємодіють переважно з найближчими сусідами, обмінюючись лише критично важливою інформацією. Проте ця концепція не повністю покриває проблему глобальної синхронізації при втраті частини повідомлень.

Однак повна децентралізація породжує фундаментальну проблему – підтримку узгодженого стану системи (*State Consistency*). У відсутності єдиного «арбітра» (сервера), який зберігає еталонну версію правди, кожен агент формує

власну локальну картину світу (*Local World Model*). Через асинхронність передачі повідомлень, пакетні втрати та затримки розповсюдження інформації в багатострибкових (*multi-hop*) мережах локальні стани різних агентів неминуче розходяться.

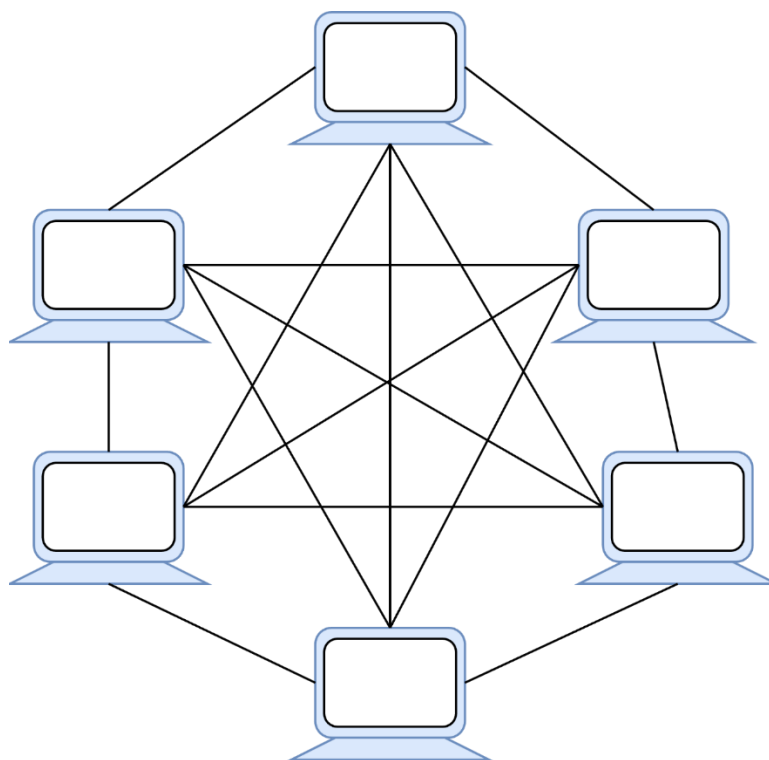


Рис. 1.1 Mesh-мережа

Теорема *CAP* у контексті мобільних систем. Проблема узгодженості в розподілених системах детально описується фундаментальною теоремою *CAP*, запропонованою (*Brewer's theorem*) у роботі [32]. Ця теорема математично доводить неможливість одночасного забезпечення трьох властивостей:

- *Consistency* (Узгодженість): Усі вузли бачать одні й ті ж дані в один і той же час.
- *Availability* (Доступність): Кожен запит отримує відповідь про успішне виконання або помилку (система завжди працює).
- *Partition Tolerance* (Стійкість до розділення): Система продовжує функціонувати, незважаючи на втрату зв'язку між частиною вузлів.

Для мобільних роїв БПЛА (*FANET*) властивість *Partition Tolerance* є даністю,

яку неможливо ігнорувати: розриви зв'язку через складнощі рельєфу, маневрування або активні радіозавади (РЕБ) є нормою, а не винятком [33]. Отже, розробники систем керування змушені обирати між *Consistency* (CP-системи) та *Availability* (AP-системи) [32].

Як зазначається в дослідженні архітектур реального часу [1], для систем керування рухомими об'єктами абсолютним пріоритетом є Доступність (*Availability*). Дрон не може «зависнути» в очікуванні синхронізації бази даних з іншими агентами – він повинен безперервно продовжувати політ, виконувати стабілізацію та уникнення перешкод. Це означає, що як показано в аналізі CAP-теорему [32], у високонавантажених децентралізованих мережах ми змушені жертвувати миттєвою глобальною узгодженістю (*Strong Consistency*) на користь моделі узгодженості в кінцевому рахунку (*Eventual Consistency*).

Однак, як ґрунтовно досліджено у праці [34], для прийняття відповідальних колективних рішень (наприклад, зміна місії, спільна атака цілі, перешикування строю) рівень неузгодженості має бути суворо обмеженим. Якщо різні частини рою будуть діяти на основі суперечливих даних (наприклад, одна група вважає, що лідер живий, а інша – що він знищений), це неминуче призведе до хаосу та провалу місії (проблема *Split-brain*).

Для вирішення цього класичного протиріччя – забезпечення скоординованих дій в умовах об'єктивно слабкої узгодженості даних – виникає необхідність у механізмах динамічного вибору локального координатора. Лідер виступає тимчасовою точкою синхронізації: він агрегує суперечливі локальні знання, приймає фінальне рішення та примусово транслює його групі, приводячи підсистему до тимчасового узгодженого стану [35].

Ефективність такого підходу критично залежить від якості алгоритму вибору лідера. Якщо обраний лідер не здатен забезпечити швидке розповсюдження команд (має великий час встановлення з'єднання або знаходиться на периферії топологічного графа зв'язності), процес відновлення узгодженості затягується, що в динамічних умовах бою рівнозначно повній втраті керування. У статті [30] автори детально проаналізували вплив високої мобільності вузлів у FANET на

стабільність топології. Запропоновані ними комунікаційні архітектури ефективно вирішують проблеми базової маршрутизації та виявлення сусідів, проте вони не покривають глобальну задачу швидкої реконфігурації та збереження стабільного лідера, що залишається однією з найскладніших невирішених проблем у проектуванні розподілених систем реального часу.

1.2. Аналіз проблем безпеки та стійкості у розподілених середовищах

1.2.1. Ландшафт загроз та модель порушника

У статті [6] автори дослідили специфіку захисту комунікацій у мережах *FANET* (*Flying Ad-hoc Networks*), які розглядаються як особливий клас розподілених систем керування. Запропоновані ними моделі частково вирішують завдання автентифікації на каналному рівні, проте залишають відкритою головну проблему таких мереж – відсутність чіткого фізичного периметра захисту та централізованих точок контролю трафіку. У класичних мережах безпека часто будується за принципом «фортеці»: довірена внутрішня зона відокремлена від зовнішнього світу брандмауерами. У роях БПЛА кожен вузол є одночасно і терміналом, і маршрутизатором, що перебуває у ворожому або неконтрольованому середовищі. Це робить систему вразливою не лише до зовнішніх впливів, а й до внутрішніх загроз, які походять від легітимних, але скомпрометованих учасників. Для систематизації ландшафту загроз у роботі [36] було запропоновано підхід на основі моделі *OSI*. Він ефективно класифікує вразливості телекомунікаційних рівнів, проте не повною мірою покриває логічні та прикладні загрози, специфічні саме для децентралізованих кіберфізичних систем.

Загрози фізичного та каналного рівнів: доступність спектра. На найнижчому рівні фундаментом функціонування рою є бездротовий канал зв'язку. Оскільки радіоефір є спільним середовищем, найкритичнішою загрозою тут виступають атаки на доступність (*Availability*), передусім – глушіння (*Jamming*). Як зазначають дослідники у [36], зловмисник, використовуючи джерела радіовипромінювання на частотах керування або передачі даних, здатен значно знизити співвідношення сигнал/шум (*SNR*) на приймачах дронів, що призводить до

деградації каналу.

Як класифікують у роботі [30], атаки глушіння (рис. 1.2) можуть бути різноманітними: від постійного випромінювання шуму, що блокує канал повністю, до «розумного» (*deceptive*) глушіння, яке імітує легітимний трафік, змушуючи протоколи канального рівня (наприклад, *CSMA/CA* в *802.11*) постійно відкладати передачу через зайнятість середовища. У контексті *FANET* це призводить до розриву зв'язності графа мережі (*network partitioning*), коли рій розпадається на ізольовані підгрупи, нездатні координувати спільну місію. У базовій праці [33] автори провели ґрунтовний аналіз вразливостей таких мереж і продемонстрували, що серйозною загрозою є також перехоплення (Eavesdropping) та аналіз трафіку. Їхні висновки допомагають зрозуміти природу атак, проте вони не пропонують ефективних методів приховування патернів трафіку, що дозволяє супротивнику виявляти місцезнаходження командних вузлів (лідерів) навіть за умови повного шифрування змісту пакетів.

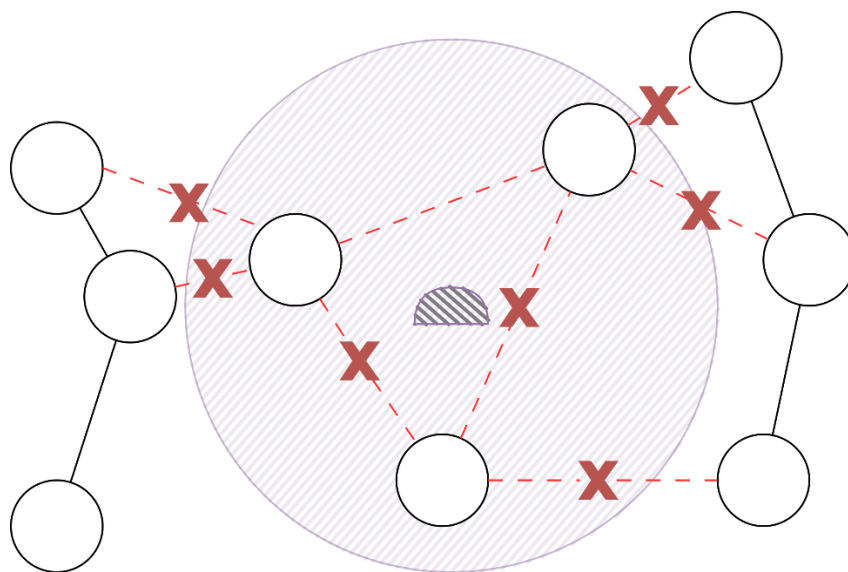


Рис. 1.2 Атака глушіння

Мережевий рівень: атаки на топологію та маршрутизацію. Специфіка *ad-hoc* мереж полягає в тому, що маршрутизація здійснюється кооперативно: кожен вузол зобов'язаний ретранслювати пакети сусідів [33]. Це створює широке поле для маніпуляцій, якщо один із вузлів виявляється зловмисним. Порушник може

експлуатувати довіру сусідів для руйнування логічної структури мережі.

Однією з найнебезпечніших атак на цьому рівні є атака *Sybil*. Її суть полягає у тому, що один фізичний зловмисний вузол генерує та привласнює собі безліч фіктивних цифрових ідентифікаторів (*virtual identities*). У дослідженні [14] автори ґрунтовно проаналізували цю вразливість і показали, що у системах, де рішення приймаються голосуванням або де ресурси розподіляються порівну між учасниками, така атака дозволяє зловмиснику отримати непропорційно великий вплив. Розроблена ними система автентифікації частково вирішує проблему підміни ідентифікаторів, проте вона виявляється малоефективною у випадку процедури вибору лідера в умовах відсутності єдиного центру. У такій ситуації один скомпрометований дрон може легітимно (на рівні криптографії) «проголосувати» сотню разів від імені неіснуючих вузлів, захопивши контроль над групою, і механізми [14] не зможуть цьому запобігти.

Інший клас атак спрямований на перехоплення потоків даних. Атака *Blackhole* («чорна діра») експлуатує механізми реактивних протоколів маршрутизації (таких як *AODV*). Зловмисний вузол на будь-який запит пошуку маршруту (*RREQ*) миттєво відповідає, що має найкоротший шлях до цілі, навіть не перевіряючи свою таблицю маршрутизації. Це змушує сусідні вузли спрямовувати весь трафік через нього. Отримавши дані, зловмисник просто відкидає їх (*packet drop*), створюючи ефект поглинання інформації. У класичній праці [37] автори запропонували механізми виявлення такої поведінки (наприклад, пасивне прослуховування). Це ефективно вирішує проблему виявлення простих «чорних дір», проте не покриває більш інтелектуальних атак, коли поведінка зловмисника змінюється з часом.

Більш витонченою версією є атака *Grayhole* («сіра діра») або вибіркова відмова в обслуговуванні (*Selective Forwarding*). На відміну від *Blackhole*, зловмисник поводить себе розумніше: він справно ретранслює певні типи пакетів (наприклад, пакети виявлення сусідів або підтвердження маршруту), створюючи ілюзію функціональної стабільності (рис. 1.3). Однак критично важливі дані (команди керування, звіти сенсорів) він вибірково відкидає. Методи моніторингу (*watchdogs*), запропоновані у згаданій вище роботі [37], виявляються безсилими

проти таких атак, оскільки для спостерігача вузол виглядає частково працездатним, а локальні втрати пакетів алгоритм списує на тимчасову деградацію радіоканалу, не застосовуючи глобальних санкцій до порушника.

Прикладний рівень: маніпуляція даними та інсайдерські загрози. Навіть якщо канали захищені від глушіння, а маршрутизація працює коректно, залишається загроза на рівні змісту даних – ін'єкція хибних даних (*False Data Injection, FDI*). Це класичний приклад «атаки інсайдера». Скомпрометований дрон має валідні криптографічні ключі та проходить процедури автентифікації, тому система розглядає його як легітимного учасника.

Зловмисник використовує цей статус для передачі правдоподібної, але хибної телеметрії. Наприклад, у задачі формування строю він може повідомляти хибні координати *GPS*, змушуючи сусідів зміщуватися і руйнувати формацію. У статті [18] автори дослідили, як у задачах розподіленого моніторингу або навчання (*Federated Learning*) такий вузол може надсилати спеціально підготовлені вектори градієнтів, які не покращують спільну модель, а навпаки – отруюють її (*Model Poisoning*), перешкоджаючи збіжності алгоритму або впроваджуючи приховані вразливості (*backdoors*). Запропоновані у [18] математичні фільтри (наприклад, *Krum*) частково вирішують проблему агрегації викривлених даних, однак не покривають специфіку атак в умовах динамічної топології (*FANET*). Небезпека *FDI* полягає в тому, що атака відбувається на логічному рівні, і традиційні засоби мережевої безпеки проти неї безсилі.

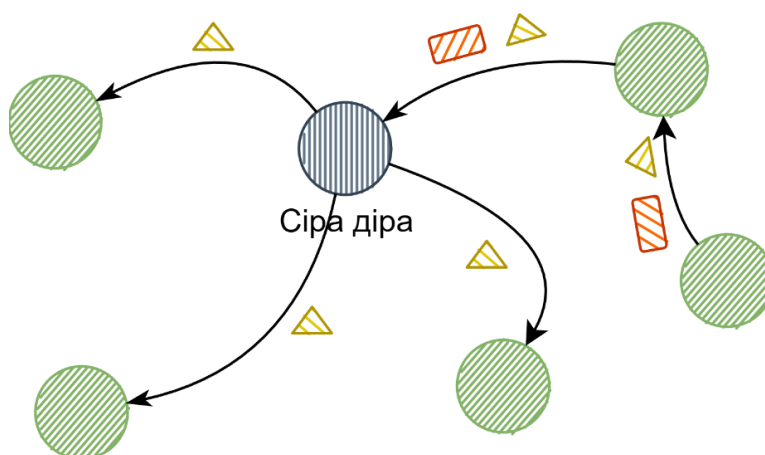


Рис. 1.3 Grayhole атака

Модель «Візантійського» вузла як узагальнення загроз. Для теоретичного аналізу стійкості систем усі вищезазначені типи поведінки узагальнюються у моделі «Візантійської помилки» (*Byzantine Fault*). На відміну від простих помилок зупинки (*Crash Failures*), де вузол просто перестає відповідати, або помилок упущення (*Omission Failures*), візантійський вузол може демонструвати довільну, непередбачувану поведінку.

Термін походить від фундаментальної праці «Проблеми візантійських генералів» [34], де формалізовано ситуацію, коли зрадники можуть координувати свої дії, щоб завадити лояльним генералам дійти згоди. У контексті кіберфізичних систем візантійський апарат може діяти «дволико»: надсилати повідомлення про атаку одному сусіду і повідомлення про відступ іншому, тим самим порушуючи узгодженість розподіленого реєстру або стану системи. У статті [34] автори математично довели, що для досягнення консенсусу в такій системі кількість добросовісних вузлів повинна обов'язково перевищувати дві третини від загальної кількості ($N \geq 3f + 1$, де f – кількість зрадників). Їхня теорема є фундаментальною і вичерпно вирішує проблему визначення меж стійкості закритих систем, проте не враховує швидкоплинність високодинамічних мереж. Це накладає жорсткі обмеження на архітектуру систем керування і вимагає застосування спеціалізованих алгоритмів *Byzantine Fault Tolerance (BFT)*, оскільки класичні мажоритарні методи стають неефективними без превентивної ізоляції порушників.

Отже, ландшафт загроз у розподілених роях формується комбінацією фізичних вразливостей каналів, атак на протоколи самоорганізації та логічних маніпуляцій даними. Як обґрунтовано в статті [21], ефективна система захисту не може покладатися лише на статичні методи (наприклад, шифрування), а вимагає впровадження систем виявлення атак (СВА) на базі машинного навчання.

Оскільки взаємодія у роях безпілотників (*FANET*) базується на безперервному обміні телеметрією та координаційними повідомленнями, будь-яке деструктивне втручання (наприклад, спроби перехоплення маршруту під час атаки *Blackhole* або перевантаження каналу) неминуче відображається на часових та об'ємних

характеристиках радіообміну. Для розпізнавання складних загроз безпосередньо у високодинамічному середовищі застосовують методи моніторингу трафіку з подальшим виділенням інформативних метрик. У статті [21] розглядається один із підходів до такої задачі та наводяться приклади типових ознак потоку, які можна аналізувати за допомогою нейронних мереж (таблиця 1.1). Динаміка цих узагальнених метрик (як-от відхилення часу прийняття пакетів чи аномальні затримки передачі) є характерним індикатором компрометації каналу або сусіднього пристрою. Використання подібного набору показників для навчання математичних моделей (наприклад, згорткових (*CNN*) або рекурентних (*RNN*, *LSTM*) архітектур) дозволяє ефективно ідентифікувати латентні ознаки кібератак ще до того, як вони спричинять фізичний розпад формації чи втрату лідера рою.

Таблиця 1.1

Приклади ознак для можливого аналізу нейронними мережами

Ознаки	Трактування
Fl-dur	Тривалість потоку
Fl-iat-max	Максимальний час між потоками
Tot-fw-pk	Об'єднання пакетів у напрямку передачі
Tot-l-fw-pkt	Загальний розмір пакета що відправляється
Tot-bw-pk	Загальна кількість пакетів що приймаються
Fw-pkt-l-min	Найменший розмір пакета що відправляється
Fw-pkt-l-avg	Середній розмір пакета що відправляється
Fw-iat-min	Довжина найменшого проміжку часу між надсиленням двох пакетів
Bw-iat-tot	Загальний час прийняття всіх пакетів
Bw-iat-avg	Середній час прийняття пакетів
Bw-iat-std	Середній час надсилення на прийняття двох послідовних пакетів
Bw-iat-max	Найбільший період очікування між прийняттям двох пакетів
Bw-iat-min	Найменший період очікування між прийняттям двох пакетів

Однак, застосування таких адаптивних моделей (СВА) безпосередньо на борту дронів робить систему захисту вразливою до атак змагального типу (*adversarial attacks*), метою яких є генерація оманливих даних під виглядом легітимної телеметрії. У цьому випадку скомпрометований вузол рою цілеспрямовано маніпулює своїм мережевим паттерном, щоб уникнути виявлення. Зловмисник формує трафік, який статистично виглядає легітимним, але примушує нейромережу віднести його до безпечного класу.

Для протидії подібним маніпуляціям у публікації [21] виконано огляд механізмів забезпечення стійкості моделей машинного навчання. Зокрема, у статті наводяться приклади поширених атак змагального типу на системи виявлення аномалій та розглядаються актуальні підходи до захисту від них (таблиця 1.2).

Дослідження [21] показує, що детекція аномалій є одним із найбільш розповсюджених та ефективних підходів до виявлення загроз у розподілених системах. Водночас у роботі доведено, що інтеграція таких рішень вимагає комплексного захисту самих моделей машинного навчання від атак змагального типу. Згідно з результатами статті, суттєвою перевагою сучасних СВА на базі штучного інтелекту є їхня здатність динамічно відбирати інформативні ознаки потоку (таблиця 1.1), обробляти великі масиви даних, мінімізувати кількість хибних спрацьовувань та адаптуватися шляхом безперервного донавчання на нових векторах атак. Також розглядається впровадження захисних механізмів (таблиця 1.2), таких як дистиляція моделей, обмеження простору допустимих значень параметрів та використання робастних класифікаторів (наприклад, *DkNN*), що дозволяє суттєво підвищити стійкість розподілених мереж до цілеспрямованих високотехнологічних впливів. Цей напрямок протидії, що об'єднує машинне навчання з механізмами захисту від змагальних атак та дозволяє розпізнавати атаки типу *DDoS*, *Probe*, *SQL*-ін'єкції тощо є передумовою для ефективного управління довірою в рої. Відповідно, використання подібних інтелектуальних інструментів для попереднього аналізу телеметрії формує необхідне наукове підґрунтя для подальшої розробки стійких до маніпуляцій методів децентралізованої координації.

Таблиця 1.2

Приклади атак змагального типу та методів захисту від них

Атаки змагального типу	Спосіб адаптації аномальних детекторів до такого виду атак
Атака за допомогою мапи помітності на основі якобіана (JSMA). При її здійсненні складається мапа чутливості вхідних параметрів, за допомогою якої визначається зміна яких параметрів дозволить змінити кінцеву класифікацію даних	Звуження допустимих діапазонів «безпечних» значень які можуть набувати параметри
	Донавчання моделей на датасети що містять дані отримані при виконанні змагальних атак
	Впровадження методу «Дистиляції», при якому відбувається розділення початкової нейронної мережі на дві, при такому підході друга мережа приймає ймовірнісний вектор класифікації та робить подальший висновок чи відбувається атака
Метод швидкого градієнта (FGSM) - атаки при яких максимізується значення функції втрат із метою зміни результату класифікації. Для прикладу додавши до зображення панди непомітні для людського ока шуми, можна отримати нове зображення панди яке буде класифікуватись як зображення гібона	Використання шару маскування градієнта (GCM) який буде модифікувати дані що проходять крізь нього, тим самим приховуючи градієнт
	Донавчання моделей на датасети що містять дані отримані при виконанні змагальних атак
	Використання автокодувальників для забезпечення виявлення оманливих даних без залучення окремих датасетів із даними змагальних атак
DeepFool - для атаки ітеративним підходом знаходиться напрямок найменших змін вхідних даних з метою зміни кінцевої класифікації	Використання Minimax захисту що застосовує генератор оманливих даних (GAN) та додатковий класифікатор для визначення підробок
	Застосування аналізу Фур'є на вхідних зображеннях та мапах параметрів
	Додавання шуму
Атака Carlini & Wagner-a у якій знаходяться найменші необхідні модифікації для зміни класифікації моделі враховуючи умови обмеження змін вхідних параметрів класифікатора	Стиснення вхідних даних, таке як розмиття зображення, зменшення глибини кольорів
	Застосування генератора оманливих даних (GAN) для підготовки моделі до виявлення оманливих даних
	Створення надійних та стійких класифікаторів за допомогою глибокого навчання на основі k-найближчих сусідів (DkNN)

Незважаючи на ефективність наведених вище методів, вибір оптимальної

архітектури нейронної мережі для систем виявлення вторгнень суттєво залежить від специфіки вхідних даних та експлуатаційних умов. Важливим аспектом є дослідження впливу гіперпараметрів на точність нейронних мереж під час аналізу статичних мережевих журналів, що було продемонстровано у роботі [22]. Оцінка ефективності класифікаційної моделі здійснювалася за критерієм точності протягом ітераційного процесу навчання. Динаміка збіжності метрик для різних конфігурацій оптимізаторів та функцій активації [22] проілюстрована на рис. 1.4. На графіках спостерігаються суттєві флуктуації показників точності для комбінацій *(ReLU, Adam)* та *(SiLU, Adam)*, що математично підтверджує високу чутливість алгоритмів оптимізації до структурних характеристик цільової функції.

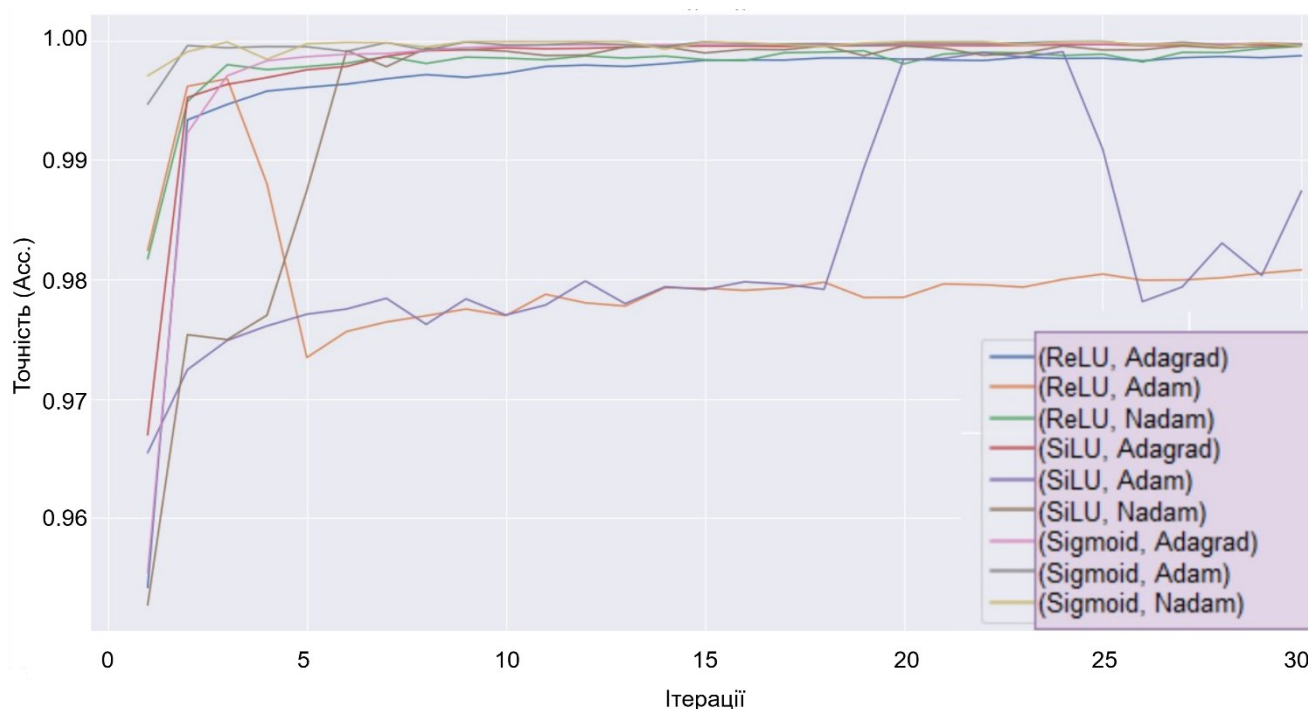


Рис. 1.4 Динаміка зміни точності виявлення кібератак під час навчання

Підсумкові результати експериментів, наведені у [22] (таблиця 1.3), емпірично доводять, що зміна лише одного гіперпараметра детермінує критичне коливання граничної точності моделі (від 33.75% до 100% при використанні оптимізатора *Adam*).

Таким чином, дослідження [22] демонструє результативність інструментів штучного інтелекту для детекції втручань у розподілені системи.

Таблиця 1.3

Залежність точності від гіперпараметрів

	Adagrad	Adam	Nadam
ReLU	97.883%	33.75%	83.316%
SiLU	99.966%	61.733%	99.916%
Sigmoid	99.966%	100%	100%

Разом з тим, розглянута модель обмежена аналізом статичних журналів подій. У контексті вирішення задач безпеки для високомобільних безпілотних мереж (*FANET*) такого підходу є недостатньо через відсутність механізмів обробки часових рядів. Відповідно, аналіз існуючих досліджень вказує на актуальність пошуку архітектур, здатних фіксувати історичний контекст взаємодії вузлів у динамічному середовищі.

1.2.2. Проблема довіри

В умовах високої динаміки та невизначеності, притаманних ролям безпілотних апаратів (*FANET*), традиційні механізми захисту інформації, побудовані на принципах криптографії, виявляються недостатніми. Як зазначається у фундаментальному огляді механізмів репутації [38], криптографічні методи («жорстка» безпека або *Hard Security*) ефективно забезпечують конфіденційність каналів зв'язку, цілісність повідомлень та автентифікацію учасників. Тобто, вони дають надійну відповідь на питання «хто ти?» та «чи не змінено повідомлення?». Однак вони залишають поза увагою фундаментальне питання доброчесності поведінки: «чи дієш ти коректно в інтересах місії?». Ця прогалина стає критичною, коли загроза походить не ззовні, а зсередини периметра системи. Для вирішення цієї проблеми необхідне впровадження систем управління довірою (*Trust Management*) [12], [39], які доповнюють криптографічний захист механізмами «м'якої» безпеки (*Soft Security*), орієнтованими на безперервний моніторинг та виявлення внутрішніх загроз безпосередньо у процесі виконання місії.

У класичних розподілених системах та корпоративних мережах домінує модель статичної довіри. Як зазначено в роботі [14], вона базується на бінарній

логіці: вузол, який успішно пройшов процедуру автентифікації (наприклад, пред'явив коректний цифровий сертифікат, підписаний довіреним центром сертифікації), вважається повністю довіреним (*fully trusted*) на весь час дії мандата. Така модель ефективна в середовищах із контрольованим фізичним доступом та стабільною інфраструктурою, однак у тактичних мережах *FANET* вона створює критичну вразливість [33].

Основна проблема, як наголошують автори дослідження [40], полягає в тому, що статичні політики не враховують можливість компрометації вузла після етапу розгортання. Як детально розглядається в огляді вразливостей безпілотних мереж [5], дрон, який на початку місії був «своїм», може бути фізично захоплений супротивником або програмно зламаний через вразливості каналу керування чи операційної системи. У такому випадку зловмисник отримує повний доступ до криптографічного матеріалу (приватних ключів) і стає «авторизованим інсайдером». Оскільки його сертифікати залишаються валідними, статична система безпеки продовжує довіряти такому вузлу, дозволяючи йому безперешкодно брати участь у маршрутизації, голосуванні та обміні сенсорними даними. Більше того, як показано в сучасному дослідженні [6], механізми відкликання сертифікатів (*CRL*), що працюють у дротових мережах, у мобільних *ad-hoc* мережах (і зокрема у *FANET*) часто неефективні через високі затримки та періодичну втрату зв'язку з центром керування.

На противагу цьому, концепція динамічної довіри (*Dynamic Trust*) розглядає довіру не як сталий статус, а як часово-змінну величину. Формально довіра визначається як суб'єктивна ймовірність того, що агент виконає очікувану дію коректно в даний момент часу та в певному контексті [40]. Рівень довіри не надається апріорі, а формується ітеративно, еволюціонуючи на основі прямих спостережень за поведінкою учасника та непрямих свідчень від інших вузлів. Це надає системі властивість адаптивності: якщо авторизований вузол починає діяти деструктивно (наприклад, вибірково відкидати пакети або спотворювати координати), рівень довіри до нього автоматично знижується. При досягненні критичного порогу такий вузол ізолюється від процесів прийняття рішень ще до

того, як адміністратор встигне відкликати його сертифікати.

Для реалізації концепції динамічної довіри розроблено низку методів оцінки репутації, які збирають «сирі» дані про поведінку вузлів, фільтрують їх і агрегують у кількісну метрику функціональної стабільності. Ці методи можна класифікувати за джерелом інформації та способом обробки.

Пряме спостереження (*Watchdog*): Одним із класичних механізмів моніторингу, що активно розглядається у сучасному дослідженні [5], є використання режиму «прослуховування» (*promiscuous mode*) бездротового інтерфейсу. Ідея базується на тому, що бездротове середовище є широкомовним. Вузол-відправник А, передавши пакет вузлу-ретранслятору В для доставки до вузла С, не вимикає приймач, а прослуховує ефір, щоб перевірити, чи переслав В цей пакет далі. Якщо В не виконує ретрансляцію протягом визначеного тайм-ауту або змінює зміст пакету, вузол А фіксує подію «невдача» і збільшує лічильник недовіри до В. Цей метод є децентралізованим і ефективним для виявлення простих атак на маршрутизацію (наприклад, *Blackhole*). Проте він має суттєві технічні обмеження:

- Колізії: В умовах перевантаженого ефіру вузол А може не почути передачу від В через інтерференцію з іншими сигналами, що призведе до помилкового звинувачення.
- Спрямовані антени: Якщо дрони використовують вузьконаправлені промені для зв'язку, вузол А фізично не зможе контролювати передачу від В до С.
- Обмежена потужність: Якщо вузол В навмисно знижує потужність передавача так, щоб сигнал дійшов до С, але не був почутий А, механізм *Watchdog* не спрацює.

Непрямі свідчення та обмін репутацією (*CONFIDANT, CORE*): Щоб пришвидшити виявлення зловмисників (особливо тих, з якими вузол ще не взаємодівав напряму), учасники мережі можуть обмінюватися інформацією про репутацію сусідів. Протокол *CONFIDANT* (*Cooperation Of Nodes – Fairness In Dynamic Ad-hoc NeTworks*) розширює локальний моніторинг компонентом

соціального контролю. Архітектура включає чотири модулі: Монітор (спостереження), Система репутації (зберігання рейтингів), Менеджер шляху (вибір маршрутів) і Менеджер довіри (обробка повідомлень). Коли вузол виявляє зловмисну поведінку, він генерує повідомлення тривоги і розсилає його своїм довіреним сусідам («друзям»). Сучасні протоколи зберігають такий підхід, що дозволяє попередити решту мережі і ізолювати порушника колективно [6]. Головним викликом тут є достовірність самих свідчень. Зловмисники можуть атакувати саму систему репутації, вдаючись до «наклепу» (*Bad-mouthing*) – надсилання хибних звинувачень проти чесних вузлів для зниження їхнього рейтингу, або «звеличення» (*Ballot-stuffing*) – штучного підвищення рейтингу своїм співникам. Протокол *CORE* вирішує цю проблему шляхом використання механізму функціональної репутації: вага голосу вузла при розрахунку загального рейтингу прямо залежить від його власної репутації. Крім того, *CORE* пропонує враховувати лише позитивні відгуки, подібні механізми залишаються актуальними і в сучасних *IoT* системах, що робить атаку наклепу неможливою, хоча й уповільнює виявлення шкідників [6].

Математичні моделі агрегації (*Beta Reputation System*): Для математично коректного перетворення історії спостережень (кількість успішних транзакцій α та невдалих β) у прогнозоване значення довіри широко використовується байєсівський підхід на основі Бета-розподілу. Очікуване значення довіри обчислюється як $E = (\alpha + 1) / (\alpha + \beta + 2)$. Ця модель, яка продовжує використовуватись як основа для сучасних блокчейн-орієнтованих систем управління довірою [12], має важливі переваги:

- Врахування невизначеності: На початкових етапах взаємодії, коли даних немає ($\alpha = \beta = 0$), довіра дорівнює 0.5 (нейтральна позиція).
- Пам'ять системи: Модель дозволяє реалізувати механізм «забування» (*aging factor*), надаючи більшу вагу недавнім подіям ($w_{new} > w_{old}$), що критично для адаптації до змін поведінки.
- Статистична стійкість: Використання щільності ймовірності дозволяє оцінювати не лише середнє значення довіри, а й впевненість у цій оцінці,

що зменшує вплив випадкових помилок каналу на репутацію вузла.

Попри значний теоретичний прогрес, існуючі системи репутації стикаються із серйозними проблемами при практичному застосуванні у високодинамічних роях БПЛА, що діють в умовах активної протидії («Візантійських» загроз).

По-перше, встановлено, що більшості традиційних систем притаманна інертність та недостатня здатність до виявлення складних нелінійних часових патернів. У дослідженні [41] авторами запропоновано базову архітектуру байєсівської системи управління довірою, яка вирішує проблему обробки апріорної статистичної невизначеності на ранніх етапах розгортання мережі. Проте, як доведено у праці [14], такий класичний Бета-розподіл не покриває проблему нестаціонарності поведінки скомпрометованих вузлів. Зловмисник здатний цілеспрямовано акумулювати статистичну історію легітимних операцій для неправомірного підвищення репутації, після чого різко змінює патерн і здійснює атаку *On-Off*. Ба більше, у роботі [15] авторами проаналізовано математичну інертність байєсівських архітектур, що підтверджує їх неспроможність своєчасно адаптувати матрицю довіри до різких змін стану і блокувати ін'єкцію хибної телеметричної інформації.

З метою мінімізації впливу попередніх спостережень поширення набули моделі з експоненційним коефіцієнтом згасання (*decay factor*) [42]. Це вирішує проблему домінування глибокої передісторії над новими подіями, проте вимагає прецизійного оптимізаційного налаштування параметра для *FANET*. Відповідно до дослідження [6], необґрунтовано високі значення згасання неминуче спричиняють хибнопозитивну (*false-positive*) ізоляцію легітимних вузлів через звичайні флуктуації радіоканалу. Натомість, застосування синтезованих згладжувальних фільтрів експоненційно знижує швидкість реакції на дійсну зміну патерну деструктивної поведінки. Це корелює з результатами огляду [7], де математично доведено функціональну нездатність агрегаторів такого типу превентивно блокувати деструктивну дію на етапі її ініціалізації.

Сучасні архітектури потребують інтеграції нейромережових та графових підходів для автоматизованої класифікації загроз. У дослідженні [43] авторами

запропоновано графові нейронні мережі (*GNN*), чия математична модель забезпечує комплексну безперервну обробку просторової взаємодії вузлів. Запропоноване рішення надійно вирішує проблему швидкого розпізнавання колаборативних атак (зокрема, атаки *Sybil* шляхом спектрального аналізу аномалій зв'язків), але залишається слабко масштабованим. Це підтверджується тим, що згідно з працею [44], завдання консенсусної синхронізації глобальної матриці суміжності для рою призводить до квадратичного зростання комунікаційної складності під час сталої реконфігурації графа мережі. Експериментально доведено [45], що відсутність спеціалізованих тензорних прискорювачів на типових платах мікродронів робить імплементацію таких графових систем інженерно малоефективною для парадигми граничних обчислень.

На основі вищезазначеного, парадигма децентралізованої поведінкової обробки на межі мережі зумовлює перехід до рекурентних нейромережових структур локального розгортання [46]. У дослідженні [39] авторами розроблено архітектуру на основі довгої короткострокової пам'яті, що вирішує проблему надійної ідентифікації прихованих динамічних атак в умовах відсутності матриці суміжності. Модель локалізує обчислення на центральних процесорах кожного незалежного борту, успішно трансформуючи задачу управління довірою у площину класифікації багатовимірних часових рядів. Архітектурна перевага такої мережі, як формалізовано у статті [47], криється у використанні блоку скидання стану. Відповідний елемент органічно вирішує проблему запізненого виявлення *On-Off* атак шляхом нелінійного нівелювання статистичної ваги історії легітимних транзакцій. Це повністю корелює з жорсткими вимогами до бортових апаратних платформ БПЛА, оскільки обробка рекурентних даних гарантує превентивну ізоляцію скомпрометованих вузлів без перевантаження системи радіозв'язку глобальними обмінами. Таким чином, рекурентні моделі ефективно нівелюють першу фундаментальну проблему – інертність алгоритмів в умовах динамічних атак.

По-друге, окрім складності виявлення нелінійних патернів, існує проблема контекстуальної обмеженості. Традиційні методи (*Watchdog*, *CONFIDANT*)

історично розроблялися для протоколів маршрутизації (*MANET*) і фокусуються на мережевому рівні – перевірці факту ретрансляції пакетів. Однак в умовах кіберфізичних систем успішна ретрансляція не є гарантією валідності даних. «Візантійський» вузол може справно виконувати функції маршрутизатора (підтримуючи високу мережеву репутацію), але водночас здійснювати ін'єкцію хибних даних (*False Data Injection*) на прикладному рівні. Як зазначається у сучасних оглядах вразливостей БПЛА [36], більшість існуючих рішень не корелюють поведінку на різних рівнях (*cross-layer analysis*), що робить їх сліпими до семантичних атак, де загроза криється не у втраті пакету, а у спотворенні його змісту.

По-третє, складність досягнення консенсусу щодо репутації у розподіленій системі. В умовах високої мобільності та частих розривів зв'язку різні частини рою можуть мати різні (іноді діаметрально протилежні) погляди на репутацію одного й того ж вузла. Без надійного механізму синхронізації, стійкого до «Візантійських» генералів (таких як сучасний алгоритм *LAP-BFT* [10]), локальні таблиці репутації стають неузгодженими. Це може призвести до розколу рою, коли одна група дронів вважає певного лідера легітимним, а інша – зловмисним, що унеможливлює скоординовані дії.

Таким чином, ефективна система довіри для рою БПЛА повинна бути не лише динамічною, а й багатофакторною (інтегрувати метрики мережевої поведінки з перевіркою узгодженості даних) та швидкодіючою (мінімізувати час реакції на атаки *On-Off*). Це вимагає розробки нових підходів, які б поєднували аналіз відхилень у реальному часі з механізмами стійкого консенсусу.

1.2.3 Задача досягнення консенсусу

Ключовою умовою функціонування будь-якої децентралізованої системи керування є здатність її учасників узгоджувати спільні параметри місії за умови відсутності єдиного довіреного центру. У кіберфізичних системах, до яких належать рої БПЛА (*FANET*), спектр даних, що вимагають узгодження, є широким: від дискретних подій (розподіл цілей, початок етапу атаки, вибір лідера) до

неперервних величин (синхронізація часу, узгодження векторів швидкості для уникнення зіткнень, злиття карт місцевості), на чому наголошується у сучасних дослідженнях архітектури таких мереж [46].

У середовищі *FANET* ця задача набуває критичного значення, оскільки розкол у прийнятті рішень (*split-brain*) може призвести не лише до логічних помилок, а й до фізичних зіткнень апаратів або зриву операції. Задача ускладнюється потенційною наявністю зловмисних вузлів, поведінка яких виходить за рамки простих відмов (зупинок). Вони можуть активно протидіяти процесу узгодження, надсилаючи суперечливу інформацію різним частинам рою (атака дволикості або «*equivocation*»). Як показує аналіз живучості безпілотних мереж [8], саме здатність мережі швидко відновлювати єдину несуперечливу картину стану є критичною для уникнення катастрофічних збоїв. Відповідна постановка відома як задача досягнення консенсусу в умовах візантійських відмов (*Byzantine Fault Tolerance – BFT*).

Проблема візантійських відмов, яка є критичною для кібербезпеки літаючих мереж згідно із сучасними розширеними класифікаціями загроз [5], визначає умови, за яких розподілена система може гарантувати дві ключові властивості: коректність (*Safety* – всі чесні вузли домовляться про одне й те саме припустиме значення) та життєздатність (*Liveness* – рішення буде прийнято за скінченний час) попри наявність f зрадників.

Фундаментальним результатом теорії розподілених обчислень [34] є доведення того, що в асинхронній мережі (де час доставки повідомлень не обмежений) без використання механізмів автентифікації (цифрових підписів) досягнення консенсусу можливе лише за умови, що загальна кількість вузлів N задовольняє нерівність:

$$N \geq 3f + 1. \quad (1.1)$$

Ця умова $(3f + 1)$ продиктована необхідністю подолати невизначеність, яку вносять зрадники. Розглянемо сценарій голосування: якщо зловмисний вузол не надсилає повідомлення, чесні вузли в асинхронній мережі не можуть відрізнити, чи він мовчить через зраду, чи просто через затримку в мережі (згідно з *FLP*-

теоремою про неможливість детермінованого консенсусу [48]). Тому система повинна мати змогу приймати рішення, ігноруючи f потенційно повільних або мовчазних вузлів. Як зазначається у фундаментальних математичних моделях, з решти $N - f$ відповідей, зловмисники (яких теж f) можуть надати хибні дані. Щоб чесні вузли могли сформувати більшість і «перекричати» дезінформацію, їх має бути більше, ніж зловмисників у найгіршому випадку. Тобто чесних вузлів має бути $N - f > f + f$, звідки $N > 3f$.

У контексті рою БПЛА це накладає жорсткі обмеження на архітектуру та вартість місії: для нейтралізації навіть одного скомпрометованого дрона $f = 1$ група повинна складатися щонайменше з чотирьох апаратів ($N = 4$). Це підвищує вимоги до апаратної надлишковості системи.

Історично першим практичним алгоритмом, що вирішив проблему BFT в умовах асинхронності (де існують певні, хоч і невідомі межі затримок), став алгоритм практичної стійкості до візантійських відмов (*PBFT*, – *Practical Byzantine Fault Tolerance*). Як зазначається у сучасних дослідженнях механізмів консенсусу для БПЛА [10], цей протокол використовує трифазну процедуру обміну повідомленнями:

- Pre-prepare: Лідер пропонує значення.
- Prepare: Вузли обмінюються підтвердженнями отримання пропозиції, щоб переконатися, що лідер не надіслав іншим щось інше.
- Commit: Вузли домовляються про затвердження значення після отримання кворуму підтверджень $2/3$.

Хоча *PBFT* гарантує математичну стійкість, його пряме застосування у мобільних роях є проблематичним через високі накладні витрати (*communication overhead*). Кількість повідомлень, необхідних для досягнення консенсусу в *PBFT*, зростає як $O(N^2)$, де N – розмір мережі. У бездротових мережах *FANET*, які часто є багатострибковими (*multi-hop*) і мають обмежену пропускну здатність, таке квадратичне зростання трафіку призводить до колізій, перевантаження каналу та експоненційного зростання затримок (*Latency*). Крім того, *PBFT* чутливий до тайм-аутів: якщо повідомлення затримуються через нестабільність радіоканалу,

протокол ініціює процедуру зміни лідера (*View Change*), що є ще більш витратною операцією і фактично блокує роботу системи на тривалий час (на що прямо вказують автори протоколу [10], обґрунтовуючи відмову від класичного *PBFT* для ескадрилей БПЛА).

Альтернативним підходом є використання технологій розподіленого реєстру (*Blockchain*) та імовірнісних алгоритмів консенсусу, таких як *Proof-of-Work (PoW)* або *Proof-of-Stake (PoS)* [49]. Хоча ці методи забезпечують високу децентралізацію і масштабованість у глобальних мережах, вони непридатні для тактичного керування в реальному часі з двох причин:

- Ймовірнісна фіналізація: Блокчейни зазвичай гарантують незмінність даних лише з певною ймовірністю, що зростає з часом (кількістю підтверджень блоків). Для контуру керування БПЛА, де рішення «повертати вліво» має бути прийняте і виконане за мілісекунди, очікування фіналізації є неприпустимим.
- Ресурсоємність: Витрати обчислювальних ресурсів на майнінг або складну криптографію, а також обсяг трафіку для синхронізації всього ланцюжка блоків є неприпустимими для бортових комп'ютерів малих БПЛА з обмеженим енергобюджетом, що підтверджується в сучасних оглядах методів консенсусу для ресурсоємних мереж [50].

Враховуючи обмеження класичних *BFT*-протоколів, сучасні дослідження в галузі керування роями зміщують фокус від «точного консенсусу» (*Exact Consensus*), де всі вузли мають погодити ідентичне бінарне значення, до «наближеного» або робастного консенсусу (*Resilient Asymptotic Consensus*), що підтверджується розробкою нових адаптивних протоколів для роїв [9]. У задачах керування рухом (наприклад, узгодження висоти польоту, вирівнювання швидкостей, формування строю) не обов'язково мати побітну рівність значень у пам'яті кожного дрона. Достатньо, щоб стани вузлів $x_i(t)$ асимптотично збігалися до спільного значення x^* , що лежить у безпечному діапазоні (*convex hull*) початкових значень чесних вузлів:

$$\lim_{t \rightarrow \infty} |x_i(t) - x_j(t)| = 0, \forall i, j \in \mathcal{H}, \quad (1.2)$$

де $\mathcal{H}$ – множина чесних вузлів.

Найбільш перспективним представником цього класу є алгоритми типу *W-MSR* (*Weighted-Mean Subsequence Reduced*). Ідея методу полягає у фільтрації екстремальних значень перед їх усередненням. На кожному кроці t кожен дрон i :

1. Збирає набір значень $V_i(t)$ від своїх сусідів.
2. Сортують отримані значення.
3. Відкидає f найбільших та f найменших значень (процедура «*trimming*»), припускаючи, що зломисники намагатимуться «розтягнути» рій, подаючи екстремальні координати.
4. Обчислює нове значення свого стану як середнє зважене по решті значень. Такий наближений консенсус забезпечує належну безпеку при безперервному керуванні динамікою БПЛА [9].
5. Переваги такого підходу для *FANET* є очевидними:
6. Локальність: Алгоритм не вимагає глобального знання топології або складних багатоетапних голосувань; кожен вузол взаємодіє лише з безпосередніми сусідами.
7. Лінійна складність: Обсяг комунікацій зростає лінійно $O(N \cdot d)$, де d – середній ступінь вузла, що економить пропускну здатність каналу.
8. Стійкість до перешкод: Метод природним чином фільтрує не лише зломисні дані, а й випадковий шум у вимірах сенсорів, розглядаючи його як незначні відхилення («викиди») [6].

Однак, *W-MSR* має свої обмеження: для гарантованої збіжності граф мережі повинен мати специфічну топологічну властивість – $(2f + 1)$ -робастність [51]. Це означає, що граф має бути достатньо зв'язним, щоб будь-яка підмножина вузлів мала достатньо вхідних зв'язків ззовні, і інформація від чесних вузлів не могла бути повністю заблокована зломисниками. Це повертає нас до необхідності підтримки якісної топології та вибору функціонально-стабільного лідера, здатного оптимізувати маршрутизацію для забезпечення цієї умови.

Аналіз проблеми консенсусу показує, що існує фундаментальний компроміс [50]. «Важкі» детерміновані протоколи (*PBFT*) забезпечують сувору безпеку, але

ціною неприпустимих затримок і блокувань у динамічних мережах. Прості алгоритми усереднення є швидкими, але вразливі до маніпуляцій даними.

Оптимальним вектором досліджень для задачі дисертації є розробка гібридного підходу, що поєднує [3]:

- Швидкість наближених алгоритмів (на базі фільтрації аномалій типу *W-MSR*) для контуру керування.
- Надійність механізмів динамічної довіри (*Trust Management*), які використовуються для адаптивного зважування голосів учасників, замість простого відкидання фіксованої кількості f значень [39].

Саме така комбінація дозволить досягти балансу між оперативністю керування та стійкістю до візантійських загроз у динамічному середовищі рою, адаптуючись до реальної кількості функціонально-нестабільних (скомпрометованих) вузлів у системі.

1.3. Систематизація задач самоорганізації та керування

Визначення функціонального призначення системи керування розподіленими об'єктами вимагає декомпозиції задач, які така система повинна вирішувати в автоматичному режимі. Специфіка сучасних розподілених систем, зокрема тих, що функціонують у високодинамічних середовищах (тактичні мобільні мережі, автономні рої дронів тощо), полягає у відсутності глобального синхронізованого часу та єдиного джерела істини. Оскільки детальна типологія базових топологічних та мережевих задач (таких як *Neighbor Discovery*, підтримка просторового строю *Formation Control*, узгодження фізичних параметрів та епідемічні механізми маршрутизації) носить загальнотеоретичний характер і є надлишковою для безпосередньої постановки власної задачі, розгорнутий опис типів керування у розподілених системах винесено в Додаток Г. У монографії [1] автори формулюють принципи проектування систем і наголошують, що розподілене прийняття рішень є їх фундаментом. Це обґрунтовує важливість задачі, проте їхні підходи орієнтовані на передбачувані індустріальні застосування і не покривають проблеми функціонування мобільних мереж в умовах ворожої

протидії, що підкреслює необхідність окремого дослідження в дисертації.

Проведений у Додатку В системний аналіз типології задач керування розподіленими системами дозволяє сформулювати цілісне, багаторівневе бачення проблеми організації взаємодії між автономними агентами в умовах невизначеності. Якщо розглядати кожен окрему функціональну задачу – від низькорівневого утримання геометричного строю та уникання колізій до високорівневої динамічної реконфігурації місії чи розподіленої обробки сенсорних даних – як ізольовану інженерну проблему, то спроба їх безпосередньої, «клаптикової» реалізації на прикладному рівні неминуче веде до архітектурного глухого кута. Практичний досвід розробки та експлуатації складних розподілених систем показує, що такий «наївний» (*ad-hoc*) підхід, де кожен агент намагається самостійно вирішувати всі аспекти координації для кожної окремої підзадачі, призводить до експоненціального зростання ентропії програмного коду. У статті [2] детально аналізується втрата керованості при наївному підході до граничних обчислень БПЛА. Хоча автори добре описують проблематику розмиття контролю, вони залишають поза увагою створення уніфікованого механізму усунення цієї ентропії на тактичному рівні комунікацій.

У такій децентралізованій «анархії» виникає ефект «спагеті-архітектури», де бізнес-логіка прийняття рішень (наприклад, «атакувати ціль») безнадійно переплетена з низькорівневою логікою мережевої взаємодії (наприклад, «чекати підтвердження пакету»), а механізми синхронізації дублюються в різних підсистемах: навігаційний модуль реалізує свої таймери, система керування корисним навантаженням – свої протоколи узгодження, а модуль обміну цілевказуванням – власні алгоритми голосування. Це створює ідеальний ґрунт для виникнення критичних, важковідтворюваних помилок: гонок даних (*race conditions*), коли різні підсистеми бачать різний стан системи в один і той самий момент часу, взаємних блокувань (*deadlocks*), коли агенти нескінченно очікують ресурсів один від одного, та абсолютно непередбачуваної, стохастичної поведінки системи в граничних станах, що робить верифікацію надійності такої системи практично неможливою. Більше того, за відсутності уніфікованого, наскрізного

шару керування, обмежені частотні та енергетичні ресурси бездротової мережі використовуються вкрай неефективно через надмірну службову «балаканину» (*chatter*) десятків неузгоджених між собою протоколів. У роботі [13] автори доводять, що конкуренція десятків локальних протоколів за ефір є критичною перешкодою. Вони вирішують це через оптимізацію енергоспоживання із застосуванням машинного навчання, проте такий алгоритмічно важкий шлях не позбавляє мережу архітектурних недоліків і є занадто повільним для мілісекундної взаємодії швидкісних агентів.

Глибокий системний погляд на виклики, що постають перед розподіленими системами спеціального призначення (РСП) – такими як рої бойових дронів, тактичні сенсорні мережі або розподілені системи радіоелектронної боротьби – дозволяє стверджувати: для ефективного, надійного, детермінованого та своєчасного виконання будь-якої прикладної задачі тактичного рівня система не повинна намагатися вирішувати їх «в лоб», вигадуючи велосипед для кожного нового типу взаємодії. Натомість, архітектура керування повинна будуватися на міцному фундаменті двох ортогональних, але критично необхідних мета-задач. Ці мета-задачі слугують своєрідною розподіленою «операційною системою» для групи агентів, забезпечуючи базові примітиви координації, синхронізації часу та безпеки. У статті [4] автори успішно демонструють ефективність використання базового фреймворку (*ROS 2*) для уніфікації логіки. Це блискуче вирішує проблему швидкого розгортання робототехніки, проте класична архітектура *ROS 2* була спроектована для довіреного середовища і без радикальних змін не здатна колективно протистояти внутрішнім візантійським атакам у зграї БПЛА.

Саме першою фундаментальною проблемою, з якою стикається будь-яка група автономних агентів у цьому контексті, є проблема ефективності прийняття колективних рішень в умовах жорстко обмеженого часу та дефіцитних комунікаційних ресурсів. Теоретично, повністю децентралізовані, «плоскі» архітектури (*peer-to-peer, mesh*), де всі вузли є абсолютно рівноправними і відсутня будь-яка ієрархія, виглядають привабливо завдяки своїй високій живучості та відсутності єдиної точки відмови. Однак, суворе математичне моделювання

процесів узгодження в таких мережах розкриває їхні критичні, системні недоліки при масштабуванні, тобто при збільшенні кількості агентів N . У повністю плоскій архітектурі, для досягнення узгодженого глобального стану (наприклад, для затвердження нового плану польоту або розподілу цілей), кожен вузол повинен обмінятися інформацією з кожним іншим вузлом, щоб переконатися, що всі учасники отримали дані і погодилися з ними. У статті [30] автори досліджують комунікаційні моделі *FANET* і доводять, що за плоскої схеми кількість службового трафіку зростає як $(O(N^2))$. Дослідники чудово ілюструють теплове навантаження на ефір, проте не пропонують ефективних детермінованих алгоритмів структурного зменшення трафіку, обмежуючись констатацією факту.

Для дротових мереж це може бути прийнятним, але для мобільних систем, що функціонують у спільному бездротовому середовищі передачі даних (*shared medium*) з обмеженою смугою пропускання та напівдуплексним режимом роботи радіомодемів, така комунікаційна складність є фатальною. Зі зростанням групи виникає лавиноподібний ефект «*broadcast storm*»: ефір забивається службовими пакетами синхронізації, різко зростає рівень інтерференції, а кількість колізій (накладання сигналів) стає такою, що механізми *CSMA/CA* перестають справлятися. У результаті, корисна пропускна здатність (*throughput*) системи стрімко прямує до нуля, оскільки канал зайнятий спробами агентів домовитися, а не передачею корисних даних. Це неминуче призводить до неприпустимого збільшення латентності системи – часу між виникненням події (наприклад, виявленням загрози) та реакцією групи на неї. У динамічних тактичних сценаріях, де рішення мають прийматися та розповсюджуватися за мілісекунди, витратити секунди на багатоступінчасті раунди переговорів «всіх з усіма» є стратегічно неприпустимим. Більше того, відсутність чіткої ієрархії унеможливорює ефективне, детерміноване вирішення конфліктів за спільні неподільні ресурси (наприклад, одночасний доступ двох дронів до однієї посадкової зони, використання одного частотного слоту для передачі відео, або атака на одну й ту ж ціль). У плоскій системі вирішення таких конфліктів вимагає складних імовірнісних алгоритмів із випадковими затримками (*random backoff*), які не дають жодних гарантій часу

виконання. У роботі [31] доведено, що гарантовані детерміновані затримки є ключовою умовою керування роями БПЛА. Автори вирішують проблему обґрунтування пріоритезації команд, проте цілком покладаються на статичні ролі виокремлених вузлів в мережі, залишаючи нерозкритим механізм динамічної адаптації під час місії.

Логічний висновок з глибокого аналізу проблеми масштабованості та оперативності безальтернативно вказує на необхідність введення динамічної, адаптивної ієрархії в контур керування. Наявність виділеного координатора, або Лідера, дозволяє радикально змінити топологію інформаційних потоків, оптимізуючи її до зіркоподібної структури. Це дозволяє лінеаризувати складність комунікацій, знижуючи її з квадратичного $O(N^2)$ до лінійного $O(N)$, оскільки рядові вузли взаємодіють лише з лідером, а не транслують повідомлення всім підряд. Лідер виступає як центральний серіалізатор подій та обчислювальний вузол: він збирає агреговану телеметрію, приймає рішення локально на основі повної картини і транслується його як директиву (*heartbeat* або *command frame*). Цей підхід замінює тривалий, багатоетапний стохастичний процес досягнення домовленості швидким, детермінованим процесом розсилки команд. Крім того, лідер стає єдиним джерелом істини та арбітром при розподілі ресурсів, що автоматично усуває можливість колізій, гонок даних та невизначеності. Однак, враховуючи специфіку застосування у ворожому середовищі, де будь-який вузол (включно з поточним лідером) може бути знищений, заглушений засобами радіоелектронної боротьби (РЕБ) або вийти з ладу через апаратний збій, система категорично не може покладатися на статично призначеного командира. Тому першою ключовою підзадачею дослідження є розробка або адаптація стійких алгоритмів динамічного вибору лідера, які повинні бути здатні не лише коректно обрати координатора при холодному старті системи, а й, що набагато важливіше, миттєво детектувати його втрату (через аналіз таймаутів *heartbeats*) та проводити процедуру переобрання (*failover*) за мінімальний час, забезпечуючи безперервність керування та мінімізуючи критичний «період безвладдя», коли система є найбільш вразливою. У класичній праці [52] розглядаються алгоритми швидкого вибору

лідера, що усуває $O(N^2)$ складність. Втім, цей базовий метод фокусується виключно на математиці відмов через ізоляцію зв'язку (*Crash Faults*) і залишається сліпим до активного шкідливого впливу, коли лідер діє як візантійський зрадник.

Втім, успішне впровадження інституту лідерства, вирішуючи проблему ефективності та швидкодії, парадоксальним чином створює нові, ще більш небезпечні вектори загроз, пов'язані з інформаційною безпекою та надійністю даних. Другою фундаментальною проблемою стає забезпечення коректного функціонування системи в умовах, коли окремі її компоненти – як рядові вузли, так і сам лідер – можуть бути скомпрометовані. У класичних розподілених системах (наприклад, у веб-серверах або хмарних базах даних) зазвичай розглядається модель відмов «*Crash Faults*» (відмова-зупинка) – вузол або працює коректно, або просто замовкає. Однак для розподілених систем спеціального призначення, що діють у зоні активного протиборства, така спрощена модель є абсолютно недостатньою і небезпечною. Тут необхідно враховувати можливість виникнення «Візантійських відмов» (*Byzantine Faults*), коли вузол продовжує працювати і залишатися на зв'язку, але поводить себе довільно або деструктивно. Це може проявлятися через «зрадливого» лідера, який навмисно розсилає суперечливі команди різним частинам групи (наприклад, одну частину посилає в пастку, а іншу змушує відступити) для дезорганізації строю, або через «брехливих» підлеглих, захоплених зловмисником чи уражених програмним вірусом, які надсилають сфальсифіковану телеметрію (наприклад, хибні координати *GPS* або фантомні цілі), змушуючи чесного лідера приймати катастрофічні рішення, які формально виглядатимуть обґрунтованими. Додатковим шаром загроз є активні мережеві маніпуляції з боку третьої сторони (*Man-in-the-Middle*), такі як вибіркове відкидання пакетів, атаки повторного відтворення (*replay attacks*) або підміна змісту повідомлень. У статті [7] для виявлення подібних маніпуляцій пропонується залучати методи машинного навчання. Це відмінно вирішує завдання класифікації аномалій мережевої поведінки у великих сенсорних системах і дозволяє регулювати ваги учасників, але впровадження таких методів стикається з фундаментальним компромісом: глибокі нейронні мережі забезпечують високу

точність виявлення, проте генерують надмірні вимоги до обчислювальних ресурсів БПЛА, тоді як простіші моделі не мають достатнього «ефекту пам'яті» для відстеження тривалої поведінки вузлів і показують значно гірші результати. Але ключовим недоліком є не сам метод регуляції ваг, а відсутність механізму ситуативної зміни вимог до порогу довіри. Замість того, щоб адаптивно підвищувати чи знижувати рівень пильності (коефіцієнт динамічного порогу) залежно від швидкоплинної тактичної обстановки та ступеня загроз, існуючі підходи пропонують уніфікований підхід, що в умовах динамічних середовищ залишає систему або надмірно обтяженою хибними тривогами, або відверто вразливою до компрометації.

У таких агресивних умовах «наївна» архітектура, побудована на сліпій довірі до повідомлень від авторизованих учасників, приречена на поразку. Простая наявність криптографічного захисту каналу та цифрового підпису лише підтверджує технічне авторство повідомлення, але жодним чином не гарантує його правдивість, семантичну коректність чи добросовісність намірів відправника. Логічний висновок вказує на те, що механізм керування вищого рівня повинен спиратися на математично доведену стійкість до зловмисних дій, що приводить нас до необхідності вирішення задачі Консенсусу в умовах візантійських відмов. Задача консенсусу в даному контексті виходить далеко за рамки простої синхронізації баз даних; вона полягає в колективній верифікації пропозицій та станів. Алгоритми *BFT* дозволяють групі чесних вузлів автоматично відфільтрувати шкідливий вплив та ігнорувати скомпрометованих агентів, доки кількість зрадників не перевищує теоретичний критичний поріг (класично $f < N/3$ для частково синхронних мереж).

Впровадження протоколів візантійського консенсусу, основи якого були закладені у фундаментальній праці [53], забезпечує розподіленій системі такі критично важливі властивості, як сильна узгодженість (*Strong Consistency*) – гарантію того, що всі справні вузли мають абсолютно ідентичний погляд на глобальний стан системи («єдина версія правди»), незважаючи на спроби дезінформації. Це вирішує проблему досягнення згоди за наявності відмов, проте

класичні підходи не покривають специфіки сучасних динамічних атак. У статті [54] автори досліджують і доводять необхідність адаптації *BFT*-систем до реальних умов вразливості, яка б додатково забезпечувала незворотність (*Finality*) – захист від зміни історії операцій, що унеможлиблює ситуацію, коли зловмисник може скасувати вже виконану команду; та живучість (*Liveness*) – здатність системи продовжувати прогрес і приймати нові рішення, навіть якщо частина вузлів саботує процес. Таким чином, другою ключовою підзадачею дослідження є синтез та оптимізація методу досягнення візантійського консенсусу, спрямований на адаптацію існуючих громіздких рішень (розроблених для стабільних дротових мереж) для забезпечення тонкого балансу між криптографічною стійкістю та високою швидкістю в нестабільних мобільних мережах з високими затримками.

Підсумовуючи архітектурний аналіз, декомпозиція загальної проблеми керування дозволила виділити ядро наукового дослідження, відкинувши другорядні технічні деталі. Ми свідомо не розглядаємо окремо задачі маршрутизації, планування траєкторій чи первинної обробки сенсорних даних як першочергові, оскільки всі вони є похідними від базових механізмів координації. Робота фокусується на забезпеченні надійного архітектурного фундаменту – «скелету» системи, що складається з двох взаємопов'язаних колон: алгоритмів вибору лідера, які забезпечують оперативну ефективність, масштабованість та швидкість реакції (відповідають за «швидке» керування), та протоколів консенсусу, що гарантують інформаційну безпеку, цілісність даних та стійкість до внутрішніх атак (відповідають за «надійне» керування). Ці дві задачі є глибоко комплементарними та нерозривними: лідер необхідний для прискорення досягнення консенсусу, виступаючи ініціатором пропозицій та оптимізатором трафіку, а незалежний механізм консенсусу необхідний для легітимізації влади лідера, перевірки його рішень та захисту групи від узурпації влади зловмисником. Вирішення саме цього архітектурного дуалізму дозволить побудувати універсальну платформу, на якій реалізація будь-яких прикладних функцій розподіленої системи стане алгоритмічно простішою, детермінованою та надійною.

1.4. Аналіз методів вибору лідера

1.4.1. Роль лідера в архітектурі розподіленого керування

Впровадження виділеного координатора, або лідера, у топологію розподіленої системи (РС) є не просто організаційним заходом, а фундаментальним архітектурним рішенням, яке радикально трансформує модель взаємодії обчислювальних вузлів, змінюючи парадигму з повністю симетричної на ієрархічну або гібридну. У контексті алгоритмів розподіленого консенсусу та керування, поняття «лідер» позбавлене адміністративного чи соціального забарвлення; це суто функціональна роль, яка передбачає виконання вузлом специфічного, математично визначеного набору алгоритмічних обов'язків, критичних для підтримки детермінованості системи в умовах невизначеності мережевого середовища. Фундаментальною теоретичною базою для розуміння цієї ролі стала праця [55], де запропоновано парадигму реплікованого автомату станів (*Replicated State Machine – RSM*), яка на сьогодні є де-факто стандартом для побудови високонадійних розподілених систем. Автор закладає надійні основи реплікації для протидії збоєм, проте цей підхід базово розрахований на статичне середовище і не враховує високої динаміки підключень та відключень вузлів, притаманної мобільним мережам. Згідно з цією моделлю, для того, щоб множина розподілених реплік (вузлів) прийшла до ідентичного кінцевого стану, вони повинні не лише отримати однаковий набір вхідних команд, але й виконати їх у суворо однаковій послідовності. Саме забезпечення цієї глобальної послідовності і стає першочерговим завданням лідера, без якого система неминуче скочується в стан ентропії та розсинхронізації даних.

Найскладнішою проблемою, яку вирішує лідер у рамках *RSM*, є серіалізація потоку подій, або проблема глобального впорядкування (*Ordering*). У дослідженні [1] переконливо доведено, що у розподілених системах відсутній єдиний фізичний годинник, а локальні таймери вузлів завжди мають дрейф, що робить використання часових міток ненадійним критерієм істини. Хоча дослідження пропонує потужні концепції для жорсткого реального часу, воно орієнтоване на провідні промислові шини і залишає невирішеними питання синхронізації в умовах непередбачуваних

затримок бездротових нестабільних каналів. В умовах, коли повідомлення можуть затримуватися, губитися, дублюватися або приходити у довільному порядку через особливості маршрутизації, різні вузли системи без централізованого арбітра можуть бачити різну причинно-наслідкову послідовність подій (*causal ordering violation*). Наприклад, команда «скасувати місію» може прийти на один дрон пізніше, ніж команда «відкрити вогонь», хоча відправлені вони були у зворотному порядку (рис. 1.5).

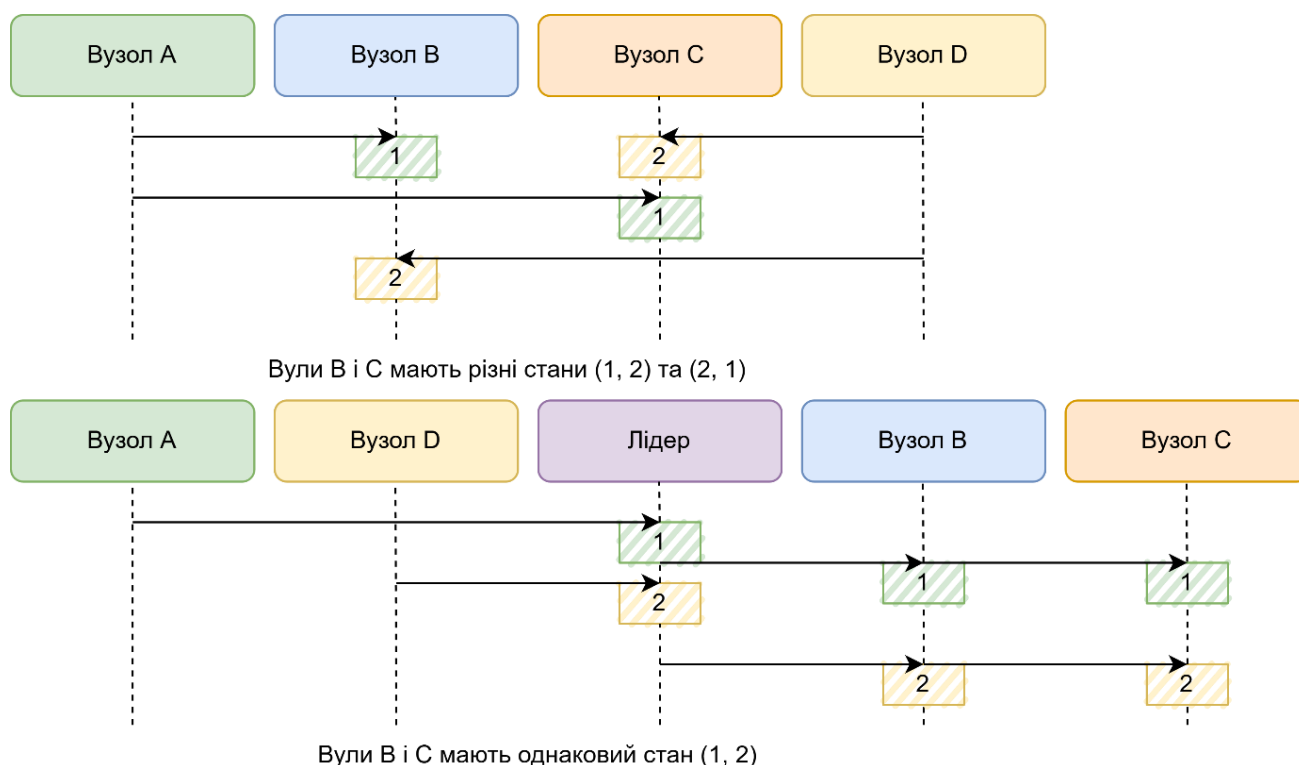


Рис. 1.5 Проблема глобального впорядкування та її вирішення із використанням лідера

Лідер усуває цю невизначеність, виступаючи як глобальний секвенсор. У фундаментальній роботі [56] наголошується на тому, що всі запити на зміну стану системи (*write requests*) від зовнішніх клієнтів або внутрішніх сенсорів повинні спрямовуватися виключно до лідера. Ця робота систематизує класичні підходи до групового розсилання, однак класичний «тотальний порядок» має критичний недолік – він створює вузьке місце (*bottleneck*) на лідері, що може призвести до відмови в обслуговуванні при інтенсивному потоці сенсорних даних у ройових

системах. Отримавши запити, лідер не просто ретранслює їх, а присвоює кожній команді унікальний монотонно зростаючий індекс (*sequence number*) або прив'язує їх до конкретної часової мітки поточної епохи керування. У статті [57] обґрунтовано механізм, що гарантує забезпечення властивості лінеаризації (*Linearizability*) – моделі узгодженості даних. Хоча цей механізм забезпечує доведену математичну коректність, його суворе застосування в чистому вигляді викликає значні блокування в асинхронних мережах, де миттєве підтвердження операцій неможливе. Тим не менш, ця модель стверджує: якщо лідер визначив, що подія А відбулася перед подією Б, то всі справні вузли системи, незалежно від затримок мережі, застосують ці зміни до своїх локальних автоматів станів саме в такому порядку, що забезпечує повний ізоморфізм кінцевого стану всіх реплік у групі.

Окрім вирішення логічних задач упорядкування, наявність лідера оптимізує використання ресурсів мережі через централізовану агрегацію телеметрії. У тактичних мобільних мережах (*MANET*) сирі дані часто перенавантажують вузькосмугові канали зв'язку. Дослідження авторів статті [58] пропонує підхід, де лідер агрегує дані, усуваючи дублікати, та розсилає лише компактні оновлення. Як демонструється в публікації [59], це суттєво економить енергію мобільних вузлів. Проте ці підходи розглядають структуру як відносно стабільну і не передбачають ситуативного регулювання порогу довіри до координатора залежно від зміни завадової обстановки.

Аналогічно, у вирішенні колізій за спільні ресурси (наприклад, класичний алгоритм взаємного виключення, що розглядається в роботі [60]) лідер слугує диспетчером, що усуває взаємні блокування (рис. 1.6). Однак такі системи покладаються на довіру «за замовчуванням» і не враховують необхідність застосування нелінійних функцій оцінки репутації в критичних умовах бою.

Лідер також виступає єдиною точкою входу для зовнішніх команд від оператора та відповідає за моніторинг життєздатності групи. Детектування збоїв є складною задачею в асинхронних системах, як зазначено в дослідженні [61]. Класичні детектори відмов працюють з фіксованими тайм-аутами і не мають ситуативних

порогів довіри, через що можуть хибно виключати корисні вузли, що опинилися під впливом РЕБ. Все це робить лідера єдиною точкою відмови (*Single Point of Failure*), створюючи необхідність у стійких алгоритмах переобрання з динамічною оцінкою рівня загроз.

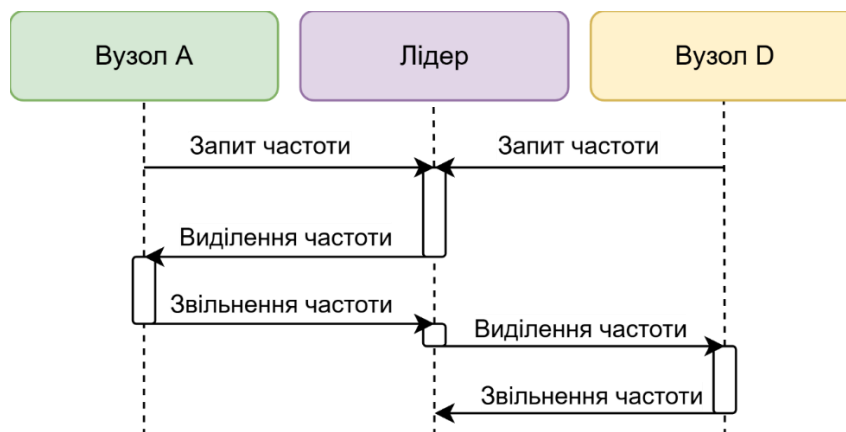


Рис. 1.6 Арбітраж ресурсів за наявності лідера

1.4.2. Алгоритми зваженої кластеризації на основі довіри (*Trust WCA*)

Еволюція алгоритмів вибору лідера в мобільних *ad-hoc* мережах (*MANET*) тісно пов'язана з розвитком методів кластеризації. У базовій роботі [62] запропоновано підхід, що обирає лідера шляхом розрахунку комбінованої ваги для кожного вузла, враховуючи фізичні параметри: залишковий заряд батареї, кількість сусідів та швидкість руху. Однак результати дослідження [63] доводять, що в умовах функціонування у ворожому середовищі орієнтація виключно на фізичні показники стає критичною вразливістю. Відповіддю на цей виклик стала розробка класу *Trust-based WCA* підходів, які розширюють класичну формулу додатковим коефіцієнтом – рівнем довіри.

Принцип інтеграції довіри базується на репутаційних моделях: кожен агент у мережі веде локальну таблицю репутації своїх сусідів на основі прямих спостережень або отримуючи рекомендації від інших вузлів, як запропоновано в статті [64]. У фундаментальному огляді [41] підкреслюється, що беззаперечною перевагою такого комплексного підходу є здатність алгоритму балансувати між життєздатністю вузла (енергія) та безпекою групи (довіра). Це дозволяє

математично формалізувати поняття функціональної стабільності та автоматизувати відсів потенційно небезпечних кандидатів. Водночас всі наявні архітектури мають спільний недолік: вони накопичують оцінки репутації на основі лінійних моделей, відкидаючи можливість застосування нелінійних функцій згладжування, які б дозволили швидко знижувати рейтинг при виявленні аномальної поведінки.

Втім, практичне застосування таких методів у тактичних мережах нашою є проблема статичної природи ваг у момент виборів. Як впливає з основ розбудови бездротових комунікацій, описаних у [65], миттєва динаміка фізичних каналів вимагає постійної адаптації. Відсутність нелінійного регулятора вимог до кандидатів та ігнорування мінливості затримок (джитера) призводить до того, що лідер з хорошою попередньою репутацією, навіть опинившись під дією завад або скомпрометованим, може тривалий час зберігати владу. Без впровадження коефіцієнта динамічного порогу довіри система уповільнює реакцію на загрози, оскільки потребує значного часу на накопичення негативної статистики.

1.4.3. Безпечні модифікації класичних алгоритмів (Secure Bully/Ring)

У той час як алгоритми на основі довіри намагаються оцінити доброчесність поведінки вузла, інший вектор досліджень спрямований на забезпечення суворої аутентичності та авторизації учасників процесу виборів. Цей підхід реалізується через розробку «безпечних» модифікацій класичних детермінованих алгоритмів, таких як *Secure Bully* або *Secure Ring*. Основна ідея полягає в інтеграції криптографічних примітивів безпосередньо в логіку протоколу для захисту від зовнішніх злоумисників та підміни повідомлень. На відміну від базової версії алгоритму «Задирака», де будь-який вузол може оголосити себе координатором, надіславши повідомлення з найвищим ідентифікатором, модифіковані версії вимагають суворого підтвердження повноважень. Як зазначають автори дослідження [66], без належної аутентифікації протоколи маршрутизації та виборів у сенсорних мережах вразливі до цілого класу атак, включаючи спуфінг,

селективне пересилання та атаки Сибілли. Захист досягається шляхом використання інфраструктури відкритих ключів (*PKI*) або симетричного шифрування, де кожне службове повідомлення про початок виборів або оголошення переможця супроводжується цифровим підписом. Отримувач, перш ніж оновити свій локальний стан або відповісти на запит, зобов'язаний верифікувати підпис відправника, перевірити актуальність його сертифіката згідно з визнаними стандартами [67] та звірити його права доступу. Такий механізм ефективно нівелює загрози типу «людина посередині» та унеможливорює ситуацію, коли сторонній пристрій, не маючи валідних криптографічних ключів, намагається узурпувати владу в кластері. Однак, незважаючи на строгу аутентифікацію, ці протоколи оперують бінарною логікою «свій/чужий» і не містять нелінійних функцій оцінки репутації для протидії скомпрометованим легітимним вузлам.

Однак впровадження криптографічного захисту в процедуру виборів лідера, яка в мобільних мережах може відбуватися досить часто, створює суттєве навантаження на обчислювальні ресурси системи. Головним недоліком цього класу алгоритмів є високі накладні витрати на виконання криптографічних операцій. Генерація та перевірка асиметричних цифрових підписів (наприклад, алгоритмів *RSA* або *ECDSA*) є ресурсомісткими задачами, які вимагають значного часу процесора та енергії батареї, що було експериментально підтверджено аналізом енергоспоживання криптографічних примітивів у роботі [6]. У контексті тактичних мікро-дронів або сенсорів Інтернету речей, що мають обмежені ресурси, додаткова затримка на верифікацію кожного повідомлення в ланцюжку виборів може призвести до неприпустимого збільшення загального часу відновлення керованості системою. Якщо в класичному алгоритмі час збіжності залежить переважно від мережових затримок, то в безпечних модифікаціях до нього додається кумулятивний час криптографічної обробки на кожному вузлі, що гальмує реакцію рою на критичні події. Відсутність механізмів адаптивного або ситуативного зменшення рівня цих перевірок під час інтенсивного бою не дозволяє системі балансувати між швидкістю та гнучким порогом безпеки.

Крім того, орієнтація на криптографічну перевірку кожного повідомлення

відкриває новий вектор атак, специфічний для ресурсних систем – атаки на виснаження (*Resource Exhaustion Attacks*) або відмову в обслуговуванні (*DoS*). Зловмисник, навіть не маючи можливості зламати шифр або підробити підпис, може реалізувати атаку, постійно ініціюючи процедуру перевиборів або надсилаючи велику кількість службових повідомлень із завідомо невірними підписами, як аналізується в дослідженні [68]. Легітимні вузли будуть змушені витратити свої обмежені енергетичні ресурси на перевірку цих підписів, щоб відхилити фальшиві запити. Як детально описано у фундаментальній праці [69], енергетичні обмеження є критичним фактором для бездротових сенсорних мереж, і будь-яка асиметрія у вартості обробки пакетів (де перевірка коштує дорожче за генерацію) дозволяє зловмиснику швидко розрядити батареї лідера та ключових вузлів. Таким чином, хоча безпечні модифікації класичних алгоритмів успішно вирішують проблему автентифікації джерела, вони часто виявляються вразливими до саботажу, спрямованого на фізичне виснаження системи. Згідно з висновками роботи [5], такі модифікації вносять затримки, які можуть бути критичними для задач реального часу. Жодна з цих моделей не застосовує концепцію динамічного порогу перевірок або нелінійного управління довірою для своєчасної ізоляції генератора такого аномального трафіку.

1.4.4. Блокчейн-орієнтовані підходи та консенсусні вибори

У пошуках архітектурних рішень, здатних забезпечити абсолютну прозорість та незворотність процесу керування у ворожому середовищі, дослідники все частіше звертаються до технологій розподіленого реєстру, які детально розглядаються в роботі [70]. Цей напрямок пропонує принципово іншу парадигму: замість того, щоб обирати лідера через прямий обмін повідомленнями, система досягає консенсусу щодо стану реєстру, а статус лідера надається вузлу, який отримав право сформувати наступний блок або має найвищий підтверджений рейтинг у ланцюжку. Основою таких методів є використання легковагових механізмів консенсусу, адаптованих для Інтернету речей, таких як «Доказ повноважень» (*Proof-of-Authority*) або «Доказ довіри» (*Proof-of-Trust*). Як

зазначають у своєму огляді [71], у цій моделі історія поведінки кожного агента – успішні місії, передані пакети, результати взаємоперевірок – фіксується у вигляді транзакцій, які криптографічно пов'язуються у незмінну послідовність блоків. Лідер у такій системі – це не просто вузол з найбільшим ідентифікатором, а актор, чия легітимність математично доведена всією історією мережевої взаємодії, збереженою на кожному пристрої рою. Це фактично унеможливорює сценарії, коли зловмисник, захопивши вузол, намагається миттєво фальсифікувати його репутацію, оскільки для цього йому довелося б переписати весь розподілений реєстр на більшості пристроїв одночасно. Водночас, ці механізми переважно спираються на жорсткі, статичні правила управління історією транзакцій, що унеможливорює застосування нелінійного оцінювання та інтеграцію динамічних порогів довіри, які б дозволили оперативно реагувати на нетипові зміни поведінки в умовах радіоелектронної боротьби.

Забезпечуючи безпрецедентний рівень цілісності даних та стійкості до маніпуляцій, блокчейн-орієнтовані підходи, однак, стикаються з жорсткими обмеженнями фізичного світу, які роблять їх малопридатними для високодинамічних тактичних мереж. Головним каменем спотикання є часова затримка (латентність) досягнення консенсусу. Процес формування блоку, його розповсюдження через ненадійні бездротові канали та отримання підтверджень від кворуму вузлів вимагає часу. Автори дослідження архітектури *FastFabric* [72] продемонстрували, що хоча оптимізація приватних блокчейнів дозволяє досягти вражаючої пропускну здатності, сама архітектура консенсусу все ще вносить затримки, які є критичними для контурів керування з жорстким реальним часом. Уявімо ситуацію бою, де рою дронів необхідно обрати нового лідера для координації ухилення від швидкісної загрози: якщо процедура виборів через блокчейн триватиме навіть на частки секунди довше, математична гарантія чесності виборів втрачає сенс через фізичне знищення системи. Подібні жорсткі гарантії досягнення консенсусу виключають можливість впровадження ситуативних, нелінійних функцій довіри, що згладжують вимоги до аутентифікації в моменти критичних загроз.

Крім того, підтримка розподіленого реєстру накладає значні накладні витрати на смугу пропускання та енергоспоживання, що суперечить вимогам до скритності та автономності тактичних засобів. Постійна синхронізація станів реєстру генерує інтенсивний фоновий трафік, який забиває радіоефір і демаскує позиції групи. Власний аналіз безпеки *IoT* у праці [73] підкреслює, що класичні блокчейн-протоколи є надто важкими для пристроїв з обмеженими ресурсами, вимагаючи впровадження спрощених архітектур, де частина функцій безпеки делегується шлюзам або більш потужним вузлам, що знову повертає проблему централізації. Таким чином, хоча блокчейн-методи забезпечують найвищий теоретичний рівень захисту від внутрішніх загроз, їхня практична імплементація в автономних мобільних системах наразі є обмеженою. Ця теза яскраво підтверджується експериментальними вимірюваннями в роботі [74], де доведено критичну невідповідність динаміки консенсусу динаміці фізичних процесів керування бездротовими сенсорами. Важливо зазначити, що жодна з існуючих моделей розподіленого реєстру не пропонує гібридних підходів із застосуванням нелінійних оцінок довіри для динамічної зміни розміру кворуму адаптивно до поточних радіoeлектронних умов.

1.4.5. Біо-інспіровані алгоритми з феромонами довіри

У прагненні подолати обмеження жорстко детермінованих алгоритмів, дослідники звернули увагу на природні механізми самоорганізації, притаманні соціальним комахам. Цей напрямок, відомий як ройовий інтелект (*Swarm Intelligence*), пропонує адаптивні підходи до вибору лідера, що базуються на метафорах поведінки мурашиних колоній (*ACO*) або бджолиних роїв (*ABC*). Фундаментальною ідеєю тут є відмова від централізованого реєстру чи складних криптографічних перевірок на користь імовірнісних механізмів, де «авторитет» вузла формується еволюційно. Як зазначають автори базової праці [75], мурахи знаходять оптимальні шляхи не завдяки глобальній мапі, а через стигмергію — непряму взаємодію через модифікацію середовища. Доцільність використання та висока ефективність сучасних гібридних мурашиних алгоритмів саме для

динамічної маршрутизації в літаючих мережах (*FANET*) була нещодавно підтверджена результатами роботи [3]. У контексті захищених мереж ця концепція трансформується у використання «цифрових феромонів довіри». Зокрема, у дослідженні [76] пропонується реалізувати біо-інспіровану кластеризацію, де агент «відкладає» на віртуальному каналі зв'язку певну кількість феромону після кожної успішної взаємодії. З часом вузол, до якого ведуть шляхи з найвищою концентрацією такого феромону, статистично виринає як природний лідер кластера. Такий підхід дозволяє системі органічно адаптуватися до змін: якщо лідер починає поводитися деструктивно, потік позитивного підкріплення припиняється, і рій поступово перемикається на альтернативного кандидата. Втім, головною вадою таких моделей на основі феромонів є їхня виключно лінійна, накопичувальна природа випаровування довіри: вони не містять вбудованих нелінійних механізмів екстреного зниження рейтингу, що залишає мережу тривалий час під управлінням скомпрометованого або пригніченого лідера в умовах радіоелектронного саботажу.

Попри високу адаптивність та природну стійкість до локальних збоїв, біо-інспіровані алгоритми мають критичний архітектурний недолік, що суттєво обмежує їх застосування у тактичних сценаріях – це повільна збіжність (*Slow Convergence*). Автори дослідження алгоритму штучної бджолиної колонії [77] демонструють, що для знаходження глобального оптимуму в багатовимірному просторі рішень потрібен поступовий ітеративний процес, який може займати значний час. У динамічному бою, де топологія мережі миттєво змінюється внаслідок швидкого маневрування дронів або активної дії засобів РЕБ, у системи просто немає ресурсів на тривалий період «навчання» та стабілізації рою. Додатково ситуація ускладнюється тим, що зменшення або «випаровування» довіри також не відбувається миттєво. Наприклад, як зазначають експерти в аналітичному огляді [78], хоча методи оптимізації роєм частинок (*PSO*) показують високу ефективність для статичних сенсорних мереж, їхня практична адаптація до високодинамічних середовищ вимагає складних механізмів перерахунку, що породжує значну інерційність у прийнятті рішень. Ця інерційність створює

небезпечне вікно вразливості, коли попередній лідер вже генерує некоректні або запізнілі команди, а новий ще не набрав достатньої «феромонної» ваги. Водночас у роботі [79] обґрунтовано доводиться, що хоча *PSO* й дозволяє обрати найбільш енергоефективного лідера, загальні обчислювальні витрати на кожну ітерацію виборів залишаються невиправдано високими для реактивних збройних систем. Отже, подібні алгоритми виявляються занадто інертними для завдань реального часу і не пропонують математичного апарату нелінійної оцінки функціональної стабільності вузлів, який міг би динамічно змінювати поріг визнання нового лідера в екстремальних ситуаціях.

1.4.6. Аналіз компромісу «Швидкість – Довіра» та проблем адаптивності

Узагальнюючи проведений у попередніх підрозділах аналіз методів організації керування в розподілених системах, можна констатувати наявність фундаментальної алгоритмічної дилеми, яку неможливо ефективно вирішити в рамках існуючих детермінованих підходів. Ландшафт сучасних рішень для вибору лідера виявляється глибоко поляризованим, де кожна група методів оптимізує один критерій ефективності за рахунок критичного погіршення інших. На одному полюсі спектру знаходяться класичні детерміновані алгоритми (типу «Задирака» або кільцевих), які забезпечують мінімальний час збіжності та низькі накладні витрати, що є ідеальним для систем реального часу. Однак, як зазначають у своїй ґрунтовній таксономії [19], ці методи розроблялися для ідеалізованих розподілених середовищ і повністю ігнорують фактор безпеки. Це робить їх абсолютно беззахисними перед найпростішими атаками на маніпуляцію ідентифікаторами або перехоплення керування у ворожому тактичному середовищі.

На протилежному полюсі знаходяться методи, що ставлять на перше місце безкомпромісну безпеку та цілісність даних – блокчейн-орієнтовані підходи та складні біо-інспіровані алгоритми. Вони забезпечують найвищий теоретичний рівень стійкості, але неминучою платою за це стає значна латентність. Дослідники у своєму систематичному огляді [49] підкреслюють, що механізми консенсусу

вносять акумульовані затримки, які часто значно перевищують допустимий час реакції системи на швидкоплинні фізичні загрози. Цей підхід філософськи унеможлиблює використання гнучких, нелінійних функцій управління довірою, які дозволили б системі тимчасово знизити критерії перевірок в умовах жорсткого дефіциту часу.

У цьому контексті найбільш перспективну, але недоопрацьовану нішу займають алгоритми зваженої кластеризації з урахуванням довіри (*Trust-WCA*). Цей клас методів являє собою «архітектурний компроміс»: вони зберігають високу швидкість збіжності, притаманну локальним алгоритмам кластеризації (на відміну від глобального консенсусу блокчейну), але при цьому базово інтегрують метрику безпеки у процес прийняття рішень. Проте, як показує критичний аналіз оглядів [41] та [42], їхнім «вузьким місцем» залишається фундаментальна недосконалість самого механізму розрахунку цієї метрики. Традиційні моделі *Trust-WCA* використовують статичні вагові коефіцієнти або виключно лінійні формули накопичення репутації (наприклад, просте усереднення успішних пакетів за фіксоване вікно часу). Такий жорсткий підхід створює небезпечну інерційність: система реагує на погіршення ситуації зі значним запізненням, оскільки їй потрібно накопичити статистику помилок. Крім того, як доведено в дослідженні [8], ці методи часто ігнорують поточну просторову динаміку каналу та варіацію затримок, фокусуючись лише на логічній доброчесності вузла, що є критичним недоліком в умовах радіоелектронної боротьби.

Спроби інтегрувати в ці алгоритми сучасні методи машинного або глибокого навчання існують, проте їхнє застосування переважно зосереджене на задачах виявлення вторгнень (*IDS*), що залишає поза увагою можливості адаптивного налаштування параметрів власне протоколу керування. Наприклад, у роботі [80], а також у дослідженнях, що використовують рекурентні мережі типу *LSTM* [47], моделі ефективно виконують функцію жорсткої бінарної класифікації загроз. Тобто нейромережа тренується категорично відповідати на питання: «Чи є цей вузол зловмисником?», що призводить до дискретних, різких змін у прийнятті рішень щодо ізоляції вузлів. Такий підхід, безумовно, підвищує безпеку, але він не

дозволяє гнучко керувати параметрами алгоритму в умовах часткової деградації зв'язку. Існуючі рішення часто демонструють дихотомію: це або «важкі» архітектури глибокого навчання з підкріпленням (*Deep RL*) з непомірно високими енерговитратами [13], або примітивні лінійні евристики. Натомість існує гостра потреба у використанні *LSTM* не як жорсткого класифікатора «свій-чужий», а як інструменту регресійного аналізу, який би формував оптимальний нелінійний поріг довіри (θ) відповідно до поточних завадових умов каналу, дозволяючи системі плавно адаптувати рівень вимог до кандидатів у лідери.

Таким чином, у поточній науковій базі існує виразна прогалина: відсутній метод вибору лідера, побудований на базі *Trust-WCA*, який би використовував легковагові рекурентні мережі для неперервного обчислення динамічного коефіцієнта довіри, а не лише для класифікації загроз. Необхідний принципово новий підхід, де нейромережа виступає аналізатором часових залежностей, що формує адаптивну нелінійну оцінку функціональної стабільності вузла на основі історії узгодженості його команд та позицій. Це дозволить органічно поєднати експлуатаційну швидкість *WCA* з інтелектуальною гнучкістю машинного навчання, забезпечуючи стійкість керування роєм в умовах екстремальної технологічної невизначеності.

1.5. Методи координації у розподілених мережах класу FANET

1.5.1. Специфіка роїв як об'єкта дослідження

Звужуючи область дослідження до конкретного класу розподілених систем, необхідно виділити літаючі ад-хок мережі (*Flying Ad-hoc Networks – FANET*) як окремий об'єкт, що якісно відрізняється від традиційних мобільних (*MANET*) або транспортних (*VANET*) мереж. Як зазначають у своєму фундаментальному огляді [81], ключовою відмінністю роїв безпілотних літальних апаратів є ступінь мобільності вузлів. Якщо в наземних мережах рух зазвичай обмежений двовимірною площиною та дорожньою інфраструктурою, то агенти *FANET* оперують у тривимірному просторі з високими швидкостями, що можуть досягати сотень кілометрів на годину. Ця високодинамічна 3D-мобільність призводить до

надзвичайно частих змін топології мережі: сусідні вузли можуть виходити із зони радіовидимості за лічені секунди, а маршрути передачі даних «живуть» дуже короткий час. У таких умовах статичні алгоритми вибору лідера або маршрутизації, розраховані на повільні зміни графа зв'язків, стають неефективними, оскільки інформація про стан сусідів застаріває ще до того, як буде використана для прийняття рішення. Крім того, взаємне переміщення вузлів викликає ефект Доплера, що додатково ускладнює підтримку стабільного фізичного з'єднання.

Іншим критичним аспектом, що визначає специфіку роїв, є нестабільність каналів зв'язку та непередбачуваність затримок. У роботі [82] підкреслюється, що на відміну від дротових або стаціонарних бездротових мереж, де час обігу пакету (*Round Trip Time – RTT*) є відносно стабільною величиною, у *FANET* цей параметр піддається значним флуктуаціям (джитер). Це зумовлено не лише зміною відстані між агентами, але й впливом атмосферних явищ, інтерференцією від власних двигунів та перешкодами на лінії прямої видимості (*Line-of-Sight blocking*) при маневруванні. Для систем керування реального часу, таких як алгоритми утримання строю, варіативність *RTT* є критичною: якщо лідер не отримує підтвердження команди вчасно, він може помилково вважати вузол втраченим і ініціювати непотрібну реконфігурацію. Тому будь-який ефективний метод керування в цьому середовищі повинен не просто оперувати фактом наявності зв'язку («є/немає»), а враховувати миттєві метрики якості каналу (*QoS*), інтегруючи їх у процес прийняття рішень щодо делегування повноважень.

Нарешті, найгострішим викликом для військових та рятувальних застосувань *FANET* є вразливість до фізичного захоплення або підміни вузлів. На відміну від захищених серверних кімнат, дрони оперують у ворожому середовищі, де існує висока ймовірність фізичного перехоплення апарату супротивником. Як детально розбирається у дослідженні [83], зломисник може вилучити криптографічні ключі з захопленого дрона (*node capture attack*) і повернути його в стрій як «легітимного» учасника, або ж запустити власний клон (*Sybil attack*), що імітує поведінку довіреного вузла. У такій ситуації традиційні методи криптографічної

аутентифікації стають безсилими: повідомлення від зрадника будуть підписані валідним цифровим ключем. Це актуалізує проблему довіри (*Trust Management*), яка виходить за рамки кібербезпеки і стає задачею поведінкового аналізу. Система повинна мати здатність виявляти аномалії в телеметрії або логіці дій «свого» вузла (наприклад, навмисне створення затримок або відхилення від курсу) і динамічно знижувати його рівень довіри, ізолюючи від керування роєм ще до того, як він зможе завдати критичної шкоди місії [5].

1.5.2. Чому стандартні рішення не працюють для дронів

Узагальнення проведеного аналізу дозволяє виявити фундаментальну невідповідність між вимогами до керування автономними групами безпілотних апаратів та характеристиками існуючих протоколів узгодження. Спроба прямого перенесення класичних алгоритмів відмовостійкості (*BFT*) у середовище літаючих мереж (*FANET*) наштовхується на жорсткі фізичні обмеження каналів зв'язку. Традиційні протоколи візантійського консенсусу, такі як *PBFT*, вимагають квадратичної кількості повідомлень ($O(N^2)$) для досягнення згоди. Як зазначають у своєму дослідженні [84], у бездротових мережах з обмеженою смугою пропускання такий інтенсивний обмін службовими повідомленнями призводить до швидкого насичення каналу. У критичні моменти, коли рій повинен миттєво реагувати на зміну тактичної обстановки, ефір виявляється «забитим» пакетами голосування, що блокує передачу команд керування та телеметрії. Це створює парадоксальну ситуацію: механізм, покликаний забезпечити надійність, фактично стає причиною відмови в обслуговуванні (*DoS*) через перевантаження мережі.

З іншого боку, сучасні розподілені реєстри (блокчейн), які теоретично вирішують проблему довіри, виявляються непридатними через часові затримки. Динаміка польоту дронів вимагає прийняття рішень у контурах керування з частотою оновлення в десятки герц (мілісекундний діапазон). Водночас, як показує аналіз [70], навіть оптимізовані для Інтернету речей блокчейн-системи мають латентність фіналізації транзакцій на рівні секунд. Час, необхідний для формування блоку, розповсюдження його мережею та отримання підтверджень від

вузлів, значно перевищує час реакції, необхідний для уникнення зіткнень або утримання строю. В умовах 3D-маневрування така інерційність є фатальною: поки система «довірено» обирає лідера через блокчейн, фізична конфігурація рою вже змінюється настільки, що прийняте рішення втрачає актуальність.

Не менш проблематичним є використання популярних алгоритмів кластеризації, таких як *WCA*. Хоча вони є обчислювально легкими, їхнім критичним недоліком є ігнорування метрик якості каналу в реальному часі. У класичній роботі [85] вага вузла розраховується на основі статичних параметрів (заряд батареї, кількість сусідів) або усередненої мобільності. Однак для керування строєм критично важливим параметром є час доставки повідомлень (*latency/jitter*). Алгоритм *WCA* може обрати лідером вузол з повним зарядом батареї, який, однак, знаходиться в зоні дії засобів РЕБ або має переповнений буфер, що призводить до великих затримок у ретрансляції команд. У огляді [46] підкреслюється, що ігнорування затримок (*RTT*) при виборі координатора призводить до розсинхронізації дій групи та можливого розпаду строю, навіть якщо формально лідер є «функціонально-стабільним» та енергоефективним.

Таким чином, для забезпечення ефективного функціонування роїв дронів у ворожому середовищі існує нагальна необхідність у розробці спеціалізованого методу. Цей метод повинен бути гібридним: обирати лідера досить швидко, базуючись на мінімізації часу доставки команд (що критично для аеродинамічної стабільності групи), але при цьому досить надійно, враховуючи динамічний рівень довіри для фільтрації скомпрометованих вузлів. Як підсумовується в роботі [5], ефективне рішення повинно уникати «важких» криптографічних накладних витрат, характерних для класичних *BFT*-систем, і натомість спиратися на легковагові механізми оцінки репутації, здатні адаптуватися до високої динаміки топології та обмежених ресурсів БПЛА.

Висновки до розділу 1

Проведений системний аналіз проблем керування групами безпілотних апаратів дозволив виявити критичне протиріччя між вимогами до оперативності прийняття рішень та необхідністю захисту від інтелектуальних атак. Специфіка літаючих *ad-hoc* мереж (*FANET*) диктує жорсткі часові обмеження: контур керування повинен реагувати на зміни тактичної обстановки за мілісекунди. Водночас, функціонування у ворожому середовищі створює загрозу появи «Візантійських» вузлів, які можуть імітувати нормальну роботу, але при цьому вносити деструктивні зміни в логіку колективної поведінки.

Виконаний огляд існуючих підходів показав, що жоден із сучасних класів алгоритмів не забезпечує необхідного балансу. Класичні детерміновані алгоритми забезпечують швидкість, але не мають механізмів перевірки доброчесності учасників. Алгоритми на базі розподілених реєстрів (блокчейн) [70] гарантують цілісність, але вносять неприпустиму часову затримку для систем реального часу. Водночас, підходи на основі класичного візантійського консенсусу (*BFT*) [10] є вразливими до умов високої мережевої невизначеності: динамічна зміна топології рою та непередбачувані затримки зв'язку призводять до неможливості підтримки кворуму та паралічу прийняття рішень. Сучасні підходи на основі машинного навчання [80] переважно виконують функцію жорсткої бінарної класифікації загроз і не забезпечують гнучкого адаптивного керування. Найбільш перспективні методи зваженої кластеризації (*Trust-WCA*) у своїх поточних реалізаціях страждають від статичності моделей оцінювання: вони використовують фіксовані правила або навчаються «офлайн» на попередньо зібраних даних. Такий підхід робить систему безпорадною перед новими типами атак, які не були передбачені на етапі розробки. Це формує наукову проблему: відсутність методів, які б використовували можливості онлайн-донавчання рекурентних нейронних мереж (зокрема *LSTM*) для формування адаптивного коефіцієнта довіри в реальному часі. Саме здатність системи навчатися безпосередньо в процесі виконання місії, динамічно коригуючи свою чутливість до аномалій, є критичною перевагою, яка наразі не реалізована в існуючих протоколах.

Для розв'язання цієї проблеми у дисертаційному дослідженні ставляться такі задачі:

1. Розробити метод динамічного оцінювання довіри під час функціонування рою безпілотних літальних апаратів в умовах візантійських відмов, який забезпечить безперервне оновлення показників репутації вузлів на основі аналізу їхньої узгодженості з групою. Метод повинен мінімізувати ефект дестабілізації станів диспетчеризації в мережі та запобігати цілеспрямованим маніпуляціям даними (зокрема атакам типу *On-Off*).
2. Запропонувати математичну модель та інструментальні засоби розрахунку адаптивних порогів довіри, які дозволять трансформувати мінливу поведінку об'єктів керування у верифіковану метрику, що забезпечить виявлення та ізоляцію скомпрометованих учасників рою в умовах високої динаміки топології.
3. Удосконалити метод вибору лідера рою на основі алгоритму зваженої кластеризації. Завдання полягає у розробці механізму гнучкого врахування довіри, що дозволить ситуативно змінювати ступінь впливу репутації вузлів залежно від поточного стану комунікаційного середовища та ефективно блокувати деструктивний вплив окремих учасників без порушення глобальної координації групи.
4. Розробити алгоритмічне та програмне забезпечення для реалізації запропонованих методів керування, яке б забезпечувало інтеграцію модулів стабілізації показників довіри з інструментами аналізу часових рядів для формування інтегрального сигналу загроз у режимі реального часу.
5. Провести імітаційне моделювання та експериментальні дослідження розробленого програмно-алгоритмічного комплексу з метою оцінки часу збіжності алгоритмів, контролю параметрів керованості рою та підтвердження підвищення стійкості інформаційно-керуючої системи до деструктивних впливів порівняно з існуючими аналогами.

РОЗДІЛ 2

МЕТОДИ ДИНАМІЧНОЇ ОЦІНКИ ДОВІРИ І ВИБОРУ ЛІДЕРА У РОЗПОДІЛЕНИХ СИСТЕМАХ

2.1. Математична модель узгодженості рою за умов візантійських відмов

2.1.1. Модель системи та середовища

У контексті даного дослідження під розподіленою системою розуміється сукупність автономних обчислювальних вузлів (агентів), які функціонують у спільному середовищі, взаємодіють між собою через бездротові канали зв'язку із динамічною топологією та об'єднані цілями, що вимагають колективного узгодження рішень без виділеного центру керування. Вузлами такої системи можуть виступати різноманітні рухомі об'єкти: наземні роботизовані платформи, підводні апарати, розумні транспортні засоби або мобільні сенсорні мережі. Зважаючи на найбільш жорсткі вимоги до просторової динаміки, обмеженості обчислювальних ресурсів та критичності часових затримок, подальша формалізація математичної моделі та розробка методів у роботі розглядаються на репрезентативному прикладі роїв безпілотних літальних апаратів (БПЛА).

Для формального опису процесів керування та узгодження рішень у рої безпілотних літальних апаратів (БПЛА) в дисертаційній роботі побудована математична модель, яка враховує дискретну природу агентів, динаміку їхнього просторового переміщення та обмеження комунікаційного середовища [30]. У даному підрозділі вводиться теоретико-множинний та графовий опис системи, визначаються вектори станів окремих вузлів та формалізуються умови інформаційної взаємодії. Такий підхід створює необхідний базис для подальшої розробки методу динамічної оцінки довіри, що базується на оцінці достовірності даних, та методу вибору лідера, що комплексно враховує як комунікаційну ефективність, так і сформований коефіцієнт довіри.

Рій дронів розглядається як часово-змінний граф (2.1)

$$G_t = (V, E_t), \quad (2.1)$$

де $V = \{1, \dots, N\}$ - множина дронів, а $E_t \subseteq V \times V$ -множина каналів зв'язку у момент

часу t . Канал зв'язку між вузлами i та j вважається наявним у момент часу t , якщо їхня взаємна відстань не перевищує фіксованого порогу радіуса $R > 0$. Умову наявності каналу зручно записати через евклідову норму: якщо $x_i(t), x_j(t) \in \mathbb{R}^3$ – поточні просторові координати відповідних дронів, то

$$\|x_i(t) - x_j(t)\| \leq R \Rightarrow (i, j) \in E_t. \quad (2.2)$$

Запропонована графова модель системи та формалізація обмежень радіуса бездротового зв'язку для дослідження комунікаційної ефективності була опублікована здобувачем у праці [23].

У межах цієї моделі вважаємо канали двонапрямними за умов симетричного розміщення та однакових радіомодулів, тож наявність каналу $(i, j) \in E_t$ призводить до існування і оберненого каналу зв'язку $(j, i) \in E_t$. Разом з тим, загальна форма $E_t \subseteq V \times V$ дозволяє врахувати можливі асиметрії без зміни позначень. Кожен дрон $i \in V$ у фіксований момент часу t характеризується трійкою параметрів, що безпосередньо використовуються в подальших формулах відбору лідера:

- Заряд батареї $E_i \in [0, 1]$ - нормалізований рівень енергії (1 відповідає повному заряду).
- Кількість сусідів $d_i(t)$ – ступінь вузла у графі G_t тобто число різних $i \neq j$ для яких $(i, j) \in E_t$.
- Позиція дрону у просторі $x_i(t) \in \mathbb{R}^3$.

Зручно також явно визначити множину сусідів дрона i у момент часу t :

$$N_i(t) = \{j \in V; \{j\}: \|x_i(t) - x_j(t)\| \leq R\}. \quad (2.3)$$

Тоді $d_i(t) = |N_i(t)|$. Нормалізований ступінь $\tilde{d}_i$, що з'явиться далі у функції корисності, отримується поділом на $d_{max}(t) = \max_{k \in V} d_k(t)$.

Ми працюємо з часово-змінною множиною ребер E_t : топологія мережі може змінюватися зі зміною взаємного розташування дронів або при входженні/виході апаратів з області радіуса R . У процесі прийняття рішення розглядається фіксований момент t (квазістаціонарний зріз), тобто всі величини $E_t, N_i(t), d_i(t), x_i(t)$ трактуються як задані на момент вибору лідера. Таким чином

відбір координатора виконується за знятою «миттєвою» топологією.

У розробленій моделі середовище функціонування рою формалізується як сукупність дискретних об'єктів або подій, що вимагають реакції системи. Введемо поняття множини особливостей середовища $F = \{f_1, f_2, \dots, f_M\} \in \Omega$. Кожна особливість f_k характеризується вектором атрибутів:

$$\text{attr}(f_k) = \{p_k, \text{type}_k, \pi_k\}, \quad (2.4)$$

де $p_k \in \mathbb{R}^3$ – просторові координати особливості, type_k – її клас (наприклад, «цільова точка», «перешкода», «зона інтересу», «базова станція»), а π_k – пріоритет, що визначає важливість реакції на цей об'єкт.

Поведінка кожного окремого агента у такій системі моделюється не як довільний рух, а як детермінована або стохастична реакція на сприйняті особливості. Введемо простір допустимих команд (дій) $A = \{a_{\text{def}}, a_{\text{avoid}}, a_{\text{track}}, \dots\}$, де a_{def} – дія за замовчуванням (наприклад, рух за маршрутом). Логіка функціонування дрона описується відображенням «Стимул–Реакція» (Stimulus-Response Mapping) Ψ :

$$\Psi: F_{\text{obs}} \rightarrow A, \quad (2.5)$$

де $F_{\text{obs}} \subseteq F$ – підмножина особливостей, що потрапили в зону чутливості сенсорів дрона. Суть цього відображення полягає в тому, що наявність певної особливості f_k спонукає алгоритм керування генерувати відповідну команду a_k . Наприклад, виявлення перешкоди ($\text{type} = \text{«obstacle»}$) викликає команду обльоту, а виявлення цілі ($\text{type} = \text{«target»}$) – команду зближення або спостереження.

Така формалізація дозволяє перейти від аналізу абстрактної «чесності» вузла до перевірки каузальної (причинно-наслідкової) узгодженості його дій. У кожному раунді керування Δt дрон i формує та транслює кортеж повідомлень:

$$\text{Msg}_i(t) = \langle \text{Status}_i(t), \text{Obs}_i(t), \text{Cmd}_i(t) \rangle, \quad (2.6)$$

де Status_i містить власні координати та заряд, Obs_i – список ідентифікованих особливостей (стимулів), а Cmd_i – пропоновану команду (реакцію). Це створює основу для виявлення аномалій: якщо дрон пропонує команду Cmd_i , яка суперечить функції Ψ для заявлених спостережень Obs_i (наприклад, пропонує «летіти прямо» при наявності «перешкоди»), або якщо його набір спостережень

Obs_i суттєво відрізняється від спостережень сусідів у тій же зоні, така поведінка класифікується як підозріла.

2.1.2. Модель загроз та маніпуляції даними

Базуючись на введеній моделі середовища, формалізуємо поведінку зловмисних вузлів. Припускається, що в рої існує підмножина скомпрометованих агентів $V_{byz} \subset V, |V_{byz}| < N/3$, які технічно здатні брати участь у протоколі обміну, але мають на меті дезорганізацію колективних дій. У даній роботі розглядається модель «Візантійських» генералів [34] з розширеними можливостями фальсифікації (Byzantine Data Falsification).

Специфіка моделювання подібних загроз, зокрема навмисного коливання рейтингу довіри (On-Off атаки) та дестабілізуючого ефекту миготіння станів мережі, досліджувалася здобувачем у роботі [25].

Зловмисна поведінка моделюється як застосування функції спотворення $\mathcal{A}$ до компонентів повідомлення перед його відправкою. Якщо для чесного вузла повідомлення $Msg_i(t)$ є результатом коректного вимірювання та застосування функції реакції Ψ , то для вузла $k \in V_{byz}$ повідомлення модифікується:

$$Msg_k^{mal}(t) = \mathcal{A}(Msg_k^{true}(t)) = \langle Status_k, \mathcal{A}_{obs}(Obs_k), \mathcal{A}_{cmd}(Cmd_k) \rangle. \quad (2.7)$$

Виділимо три базові класи атак, що порушують причинно-наслідковий зв'язок «Стимул–Реакція»:

1. Фальсифікація сприйняття (Perceptual Hallucination): Зловмисник повідомляє про виявлення особливостей середовища, яких фізично не існує в його зоні огляду, з метою спровокувати хибну реакцію рою (наприклад, різке гальмування перед неіснуючою перешкодою). Математично це описується як ін'єкція хибної множини стимулів F_{fake} :

$$Obs_k^{mal} = Obs_k^{true} \cup F_{fake}, \text{ де } F_{fake} \cap F = \emptyset. \quad (2.8)$$

2. Саботаж реакції (Action Suppression): Агент коректно ідентифікує реальні стимули (наприклад, ціль або небезпеку), але навмисно ігнорує їх при формуванні команди, пропонуючи дію «за замовчуванням» або

протилежну до необхідної. Це відповідає атаці на функцію відображення:

$$Cmd_k^{mal} \neq \Psi(Obs_k^{true}). \quad (2.9)$$

Частковим випадком є «сліпота» сенсорів (Sensor Blindness Attack), коли з множини спостережень вилучаються критичні об'єкти:

$$Obs_k^{mal} = Obs_k^{true} \setminus \{f_{critical}\}. \quad (2.10)$$

3. Підміна контексту (Contextual Spoofing): Найскладніший тип атаки, при якому зловмисник маніпулює атрибутами реально існуючої особливості (наприклад, змінює її тип з «свій» на «чужий» або спотворює координати p_k на вектор зсуву δ), щоб легітимізувати шкідливу команду.

$$f'_j \in Obs_k^{mal}: attr(f'_j) = attr(f_j) + \delta_{attack}. \quad (2.11)$$

Така маніпуляція дозволяє зловмиснику пропонувати деструктивні команди, які формально виглядають логічними відносно «підробленої» картини світу.

Розроблений у наступних підрозділах метод динамічної оцінки довіри дозволяє приймати виважені колективні рішення, автоматично зменшуючи вплив джерел недостовірної інформації. Важливо зазначити, що в реальних умовах спотворення даних (реалізація функції $\mathcal{A}$) не завжди є наслідком цілеспрямованої зловмисної дії. Часто аномальна поведінка вузла може бути результатом апаратної несправності дрона, деградації сенсорів, або ж виникати через природні обмеження видимості внаслідок складних погодних умов чи наявності фізичних перешкод. Запропонований підхід є універсальним щодо природи джерела помилок: він аналізує кінцевий результат – узгодженість даних, що дозволяє системі ефективно ізолювати як «зрадників», так і несправні або «осліплені» вузли, забезпечуючи безпеку місії без необхідності з'ясування першопричини відхилення.

2.2 Метод динамічної довіри та стійкого консенсусу з нейромережевим підсиленням

2.2.1. Концепція та архітектура методу

Попередньо визначена модель загроз демонструє, що функціонування рою в умовах візантійської поведінки неможливе без надійного механізму фільтрації

вхідних даних [41]. Оскільки зловмисні або несправні вузли здатні генерувати формально коректні, але змістовно хибні повідомлення (фальсифікація сприйняття, підміна контексту), проста агрегація команд від усіх учасників неминуче призведе до дезорганізації строю. Критичною проблемою стає не так втрата пакетів, як неможливість відрізнити «чесну» пропозицію маневру від саботажу на рівні протоколу обміну. Тому першочерговим завданням є створення методу, здатного в режимі реального часу оцінювати достовірність інформації від сусідів та забезпечувати узгодження рішень навіть за наявності активної протидії.

Як рішення пропонується розроблений автором метод динамічної оцінки довіри та стійкого консенсусу з нейронним підсиленням [25]. Його архітектура базується на припущенні, що аномальну поведінку можна виявити через аналіз неузгодженості дій вузла з контекстом групи. Метод поєднує три взаємопов'язані механізми:

1. Динамічне оцінювання довіри. Кожен дрон у процесі обміну повідомленнями формує оцінку функціональної стабільності своїх сусідів, використовуючи комплекс метрик узгодженості: просторової (відповідність позиції), командної (адекватність пропозицій) та часової [42]. На відміну від статичних «білих списків», ця оцінка безперервно оновлюється: якщо реакція дрона на середовище суперечить спостереженням більшості, його рівень довіри знижується, що автоматично зменшує його вплив на систему.
2. Стійкий консенсус із вагами довіри. Ухвалення колективного рішення (наприклад, про обліт перешкоди) відбувається не за принципом «один вузол – один голос», а через зважене голосування. Голоси агентів із високою репутацією є визначальними, тоді як пропозиції підозрілих вузлів ігноруються. Це дозволяє математично нівелювати вплив вектора атаки $\mathcal{A}$ без необхідності блокування учасників.
3. Адаптивне нейронне підсилення. Для подолання проблеми статичних порогів у контур оцінювання інтегрується компактна рекурентна нейромережа (*LSTM*) [47]. Вона аналізує часові ряди відхилень у поведінці

сусідів і прогнозує очікуваний рівень узгодженості. Це дозволяє системі адаптуватися до складних умов (наприклад, турбулентності чи інтенсивного маневрування), динамічно змінюючи чутливість алгоритму до аномалій, щоб уникнути хибних спрацювань.

Наукова новизна методу полягає у відмові від бінарної класифікації вузлів («свій/чужий») на користь неперервної метрики довіри, яка регулюється нейромережею «на льоту». Таке поєднання дозволяє зберегти керованість роєм навіть у ситуаціях, коли частина агентів транслює суперечливі команди, що є недосяжним для класичних жорстких алгоритмів консенсусу.

Метою методу є створення універсального фільтра на рівні прийняття рішень, який забезпечує:

- ізоляцію впливу недостовірних даних (*Data Injection*);
- адаптивну реакцію на зміну тактичної обстановки;
- стабільне узгодження спільних дій в умовах невизначеності;
- збереження низької обчислювальної складності для роботи на бортових контролерах.

Реалізація цього методу створює надійний фундамент для наступних етапів керування, зокрема для вибору лідера, гарантуючи, що координатором може стати лише вузол з підтвердженою історією адекватної поведінки.

2.2.2. Модель інформаційного обміну та структура повідомлень

Ефективність роботи методу динамічної оцінки довіри та стійкого консенсусу з нейронним підсиленням залежить від того, яким чином організовано інформаційний обмін у роі. На відміну від простих моделей, де передається лише одна команда чи стан, тут структура повідомлень поділяється на кілька рівнів, що дозволяє відстежувати узгодженість та своєчасність дій учасників.

У рамках кожного раунду комунікації кожен дрон формує та надсилає сусідам повідомлення одного з трьох типів, що відповідає сучасним вимогам до надійності обміну в літаючих мережах [33]:

1. Повідомлення стану (*status*). Містить ключові характеристики дрона:

поточну позицію $x_i(t)$, заряд батареї $E_i(t)$, орієнтацію та швидкість. Ці дані відображають фактичний стан апарата і є основою для перевірки узгодженості з повідомленнями інших учасників.

2. Повідомлення пропозиції (proposal). Містить локальне рішення дрона щодо наступного кроку з визначеного простору дій $a_{local} \in A$ – наприклад, вибір точки маршруту чи напрямку руху. Цей елемент використовується у фазі прийняття рішень і дозволяє оцінити, наскільки дрон дотримується узгодженої стратегії.
3. Повідомлення голосування (vote). Відображає вибір дрона серед запропонованих варіантів рішення. Голосування здійснюється з урахуванням довіри до сусідів, і саме на цьому етапі реалізується механізм стійкого консенсусу, що концептуально спирається на принципи кворуму в розподілених системах [71].

Кожен дрон не має глобальної інформації про увесь рій. Він оперує лише локальними даними, які отримує від своїх сусідів у межах радіусу зв'язку. Цей набір даних формує вікно спостережень, що включає:

- отримані повідомлення стану, пропозицій і голосувань;
- часові мітки їх отримання;
- власні параметри дрона.

На основі цього вікна обчислюються метрики узгодженості, що використовуються у моделі довіри.

Процес обміну в кожному раунді можна описати трьома фазами:

1. Фаза стану. Усі дрони розсилають свої поточні параметри.
2. Фаза пропозицій. Кожен дрон формує пропозицію наступної дії та поширює її серед сусідів.
3. Фаза голосування. Дрони обирають рішення, зважуючи пропозиції відповідно до рівнів довіри, й надсилають свої голоси.

У результаті цього циклу система отримує набір голосів, з яких формується спільне рішення. Поділ на три типи повідомлень забезпечує багаторівневу перевірку узгодженості:

- повідомлення стану дозволяють контролювати реальний фізичний стан дрона;
- повідомлення пропозицій перевіряють, чи дотримується він колективної логіки;
- повідомлення голосувань безпосередньо впливають на кінцевий вибір.

Завдяки цьому система має змогу виявляти відхилення ще до етапу фінального консенсусу. Якщо дрон надсилає некоректні дані у фазі стану або пропозицій, це одразу відображається у зниженні довіри, що вплине на вагу його голосу у наступній фазі.

Робота методу організована у вигляді дискретних часових інтервалів – раундів $t = 1, 2, \dots$. Кожен раунд має фіксовану тривалість ΔT і складається з п'яти послідовних етапів. Нехай V – множина всіх дронів, а $N_i(t) \subseteq V$ – множина сусідів дрона i , від яких отримано сигнал у поточному раунді, тоді алгоритм прийняття рішень виглядає наступним чином (рис. 2.1):

1. Широкомовна розсилка (*Broadcast Status/Proposal*): На початку раунду кожен дрон i зчитує показники власних сенсорів, визначає свій стан $Status_i(t)$ (координати, заряд) та формує пропозицію $Cmd_i(t)$ відповідно до функції реакції Ψ . Сформований кортеж повідомлення транслюється у мережу:

$$Msg_i(t) \leftarrow \{Status_i(t), Cmd_i(t)\}. \quad (2.12)$$

2. Збір даних (*Collection*): Протягом вікна очікування $\tau_{wait} < \Delta T$ дрон i прослуховує ефір і накопичує множину повідомлень від сусідів:

$$\mathcal{M}_i(t) = \{Msg_j(t) \mid j \in N_i(t)\}. \quad (2.13)$$

Повідомлення, що надійшли з запізненням (після закриття вікна), відкидаються або переносяться на наступний раунд, що фіксується як порушення часової регулярності.

3. Перевірка узгодженості (*Consistency Check*): Для кожного отриманого повідомлення Msg_j дрон i виконує перевірку даних. На цьому етапі обчислюються локальні метрики відхилення (просторова, командна та часова), математичний апарат для яких наведено далі. Результатом є набір нормалізованих оцінок, що характеризують ступінь аномальності поведінки сусіда j у поточному раунді.

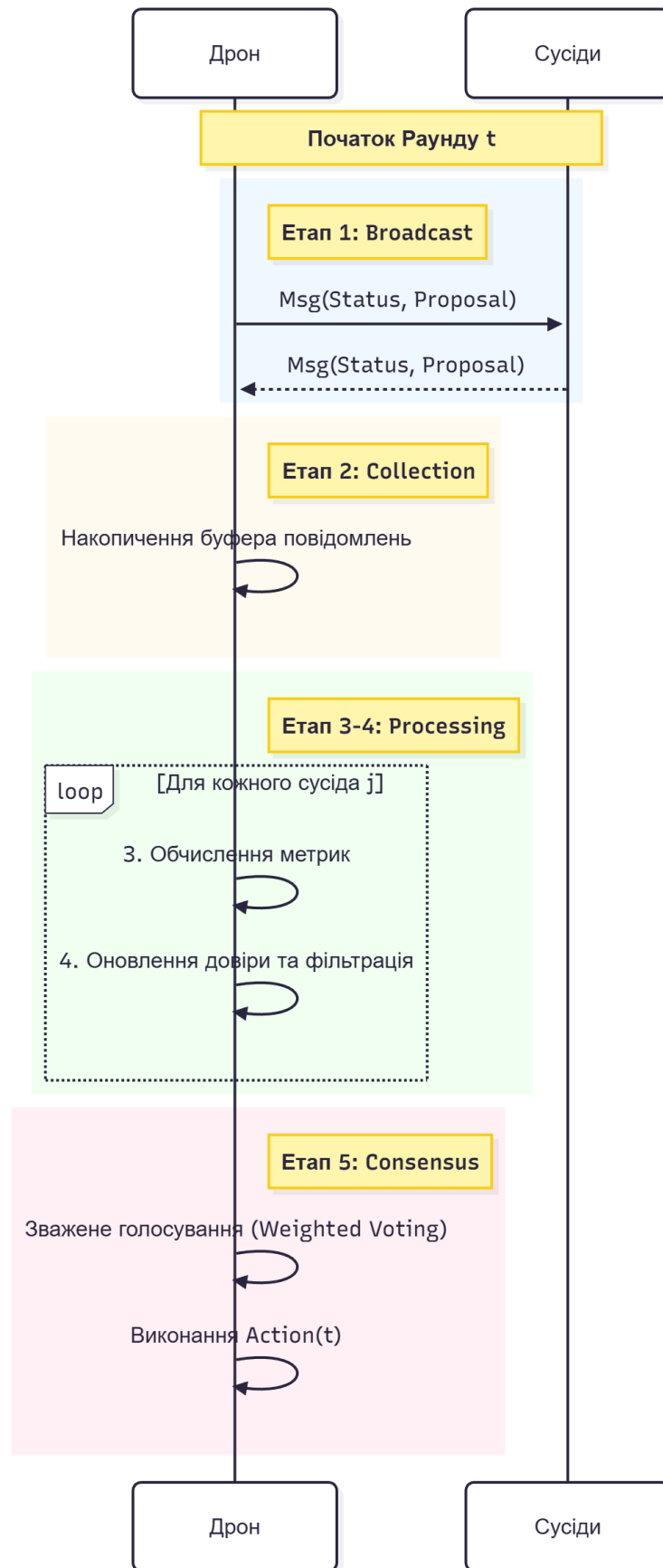


Рис. 2.1 Алгоритм прийняття рішень із застосуванням методу динамічної оцінки довіри та стійкого консенсусу з нейронним підсиленням

4. Оновлення довіри та фільтрація (*Trust Update and Filtering*): На основі обчислених метрик відбувається оновлення інтегральної репутації вузлів. Система застосовує механізм адаптивного порогу (реалізований через нейромережу, див. далі) для визначення статусу кожного сусіда. Вузли, чия поведінка систематично відхиляється від норми, переводяться у стан «карантину» (алгоритм ізоляції описано далі). Результатом етапу є набір ефективних ваг $\{\omega_{ij}(t)\}$, де вага зловмисних або функціонально-нестабільних вузлів примусово зануляється.

5. Голосування та Консенсус (*Consensus Vote*): Дрон i формує власне фінальне рішення, агрегуючи пропозиції сусідів з урахуванням їхніх ефективних ваг. Це дозволяє ігнорувати дані від скомпрометованих джерел. Дрон переходить до виконання дії $a \in A$, яка набрала найбільшу зважену підтримку:

$$Action_i(t) = \operatorname{argmax}_{a \in A} \sum_{j \in N_i} \omega_{ij}(t) \cdot \mathbb{I}(Cmd_j = a). \quad (2.14)$$

2.2.3. Метрики узгодженості

Щоб оцінювати функціональну стабільність кожного дрона у роі, необхідно мати кількісні показники, які відображають, наскільки його поведінка відповідає колективним очікуванням. У методі динамічної оцінки довіри для цього вводяться три базові метрики: узгодженість позицій, узгодженість команд та регулярність повідомлень.

Нехай $x_i(t) \in \mathbb{R}^3$ – вектор положення дрона i у момент часу t . Для перевірки узгодженості розраховується середня позиція сусідів $N_i(t)$, що відповідає принципам кластеризації на основі просторової близькості [20]:

$$\bar{x}_{N_i(t)} = \frac{1}{|N_i(t)|} \sum_{j \in N_i(t)} x_j(t). \quad (2.15)$$

Відхилення дрона i від цього середнього значення обчислюється як

$$\Delta x_i(t) = \|x_i(t) - \bar{x}_{N_i(t)}\|. \quad (2.16)$$

Тоді метрика узгодженості позицій нормалізується у діапазон $[0, 1]$.

$$s_{pos}(i, t) = \exp\left(-\frac{\Delta x_i(t)}{\sigma_{pos}}\right), \quad (2.17)$$

де $\sigma_{pos} > 0$ – параметр масштабу, що визначає чутливість. Чим ближчий дрон до середньої позиції своїх сусідів, тим ближче s_{pos} до 1.

Наступним кроком буде визначення узгодженості команд. Нехай $u_i(t)$ – керуючий вектор (наприклад, напрямок руху чи наступна цільова точка), обраний дроном i . Для порівняння розглядається множина пропозицій від сусідів: $\{u_j(t): j \in N_i(t)\}$. Визначимо середню команду сусідів:

$$\bar{u}_{N_i}(t) = \frac{1}{|N_i(t)|} \sum_{j \in N_i(t)} u_j(t). \quad (2.18)$$

Відхилення дрона від цього середнього:

$$\Delta u_i(t) = \|u_i(t) - \bar{u}_{N_i}(t)\|. \quad (2.19)$$

Метрика узгодженості команд:

$$s_{cmd}(i, t) = \exp\left(-\frac{\Delta u_i(t)}{\sigma_{cmd}}\right), \quad (2.20)$$

де $\sigma_{cmd} > 0$ – параметр масштабу для команд. Високе значення s_{cmd} означає, що запропонована команда a_{local} є адекватною реакцією на спостережувані особливості середовища $\{f_1, f_2, \dots\}$, про які повідомляють сусіди. Остання з трьох метрик узгодженості – регулярність повідомлень. Ця метрика розроблена на основі загальних принципів оцінювання затримок у мережах передачі даних [86]. Нехай t_k^i – моменти часу, коли дрон i надсилав повідомлення. Очікуваний інтервал між повідомленнями дорівнює ΔT_{exp} . Фактичні інтервали:

$$\Delta T_k^i = t_k^i - t_{k-1}^i. \quad (2.21)$$

Тоді відносне відхилення регулярності визначається як

$$\delta T_i(t) = \frac{|\Delta T_k^i - \Delta T_{exp}|}{\Delta T_{exp}}. \quad (2.22)$$

Відповідна метрика:

$$s_{tim}(i, t) = \exp\left(-\frac{\delta T_i(t)}{\sigma_{tim}}\right), \quad (2.23)$$

де $\sigma_{tim} > 0$ – коефіцієнт чутливості. Високе значення s_{tim} означає, що дрон надсилає повідомлення з очікуваною регулярністю.

Таким чином маємо наступне вираження стану розподіленої системи через вищенаведені метрики:

- Якщо дрон рухається синхронно з сусідами, його s_{pos} високий.
- Якщо він пропонує команди, подібні до пропозицій більшості, s_{cmd} високий.
- Якщо він надсилає повідомлення без значних затримок, його s_{tim} близький до 1.

Усі три метрики мають значення від 0 до 1, де 1 відповідає ідеальній узгодженості, а значення, близькі до 0, сигналізують про підозрілу або ненадійну поведінку.

Окремі показники узгодженості – s_{pos} , s_{cmd} та s_{tim} – відображають різні аспекти поведінки дрона. Проте для подальшого аналізу та використання у процедурі довіри необхідно сформувати єдиний інтегральний показник, який би враховував їх комплексно. Визначимо загальну оцінку узгодженості дрона i у момент часу t як вагову комбінацію трьох метрик:

$$s_i(t) = \omega_{pos} \cdot s_{pos}(i, t) + \omega_{cmd} \cdot s_{cmd}(i, t) + \omega_{tim} \cdot s_{tim}(i, t), \quad (2.24)$$

тут $\omega_{pos}, \omega_{cmd}, \omega_{tim} \geq 0$ – вагові коефіцієнти, що відображають відносну важливість відповідної метрики, причому

$$\omega_{pos} + \omega_{cmd} + \omega_{tim} = 1. \quad (2.25)$$

Описані вагові коефіцієнти інтерпретуються наступним чином:

- Якщо у місії головним є точне збереження формації, найбільшу вагу надають ω_{pos} .
- Якщо критичним є виконання однакових цільових команд, перевага надається ω_{cmd} .
- Якщо важливо підтримувати стабільний ритм обміну даними, зростає роль ω_{tim} .

Таким чином, вибір вагових коефіцієнтів дає змогу адаптувати метод під різні сценарії використання рою.

Відтак описаний вище інтегральний показник має наступні властивості:

- Нормалізація. Оскільки кожна з метрик належить до діапазону $[0,1]$, інтегральний показник також належить цьому діапазону.
- Гнучкість. Зміна ваг дозволяє швидко переналаштовувати метод без зміни самої структури формул.
- Чутливість. Якщо хоча б одна метрика має значно низьке значення, то і загальна оцінка s_i зменшується, сигналізуючи про потенційну проблему.

Отримане значення $s_i(t)$ є основою для розрахунку динамічної довіри. Воно відіграє роль «первинного сигналу», який у наступних кроках буде згладжено у часі та поєднано з результатами роботи нейронної мережі. Саме через інтегральний показник забезпечується баланс між різними аспектами поведінки дрона та дає змогу оцінити, наскільки поведінка дрона відповідає очікуваним нормам у конкретний момент часу. Однак одноразова оцінка є надто чутливою до випадкових відхилень. Наприклад, через шум каналу чи затримку передачі окреме повідомлення може вийти з помилкою, хоча загалом дрон діє правильно. Тому необхідно враховувати історію поведінки учасників.

2.2.4. Алгоритм розрахунку довіри та механізм карантину

Аби вирішити проблему одноразової оцінки застосовується ідея динамічної довіри. Динамічна довіра – це числовий показник, який відображає не тільки поточну поведінку дрона, але й накопичений досвід взаємодії з ним. На відміну від бінарних моделей («довіряю / не довіряю»), тут довіра безперервно змінюється у діапазоні $[0,1]$, відображаючи ступінь функціональної стабільності. Вона повинна:

- згладжувати випадкові шуми;
- швидко знижуватися за повторюваних відхилень;
- мати мінімальні обчислювальні витрати.

Оновлення довіри дрона i у момент часу t виконується за правилом експоненційного згладжування:

$$trust_i(t) = \alpha s_i(t) + (1 - \alpha) trust_i(t - 1), \alpha \in (0,1], \quad (2.26)$$

тут $s_i(t)$ – нове спостереження, $trust_i(t - 1)$ – попереднє значення довіри, α – коефіцієнт чутливості.

Для формалізації оновлення довіри було застосовано правило експоненційного згладжування (EWMA), що було досліджено автором у роботі [25] через його наступні переваги:

1. Адаптивність і «коротка пам'ять». Експоненційне згладжування підсилює вплив свіжих спостережень і «старить» давні. Це забезпечує швидку реакцію на зміни без надмірних коливань – базова перевага методу, що широко описана у підручниках з прогнозування часових рядів [87].
2. Чутливість до малих, але стійких відхилень. У статистичному керуванні процесами карти типу EWMA застосовують саме для раннього виявлення невеликих зсувів середнього рівня сигналу. Для рою це означає своєчасне зниження довіри до дрона, який «потроху» відхиляється від норми, але не карає за поодинокі шуми [41].
3. Перевірена практика у мережах реального часу. У телекомунікаціях експоненційне згладжування давно стало стандартом для стабілізації «шумних» метрик затримок, зокрема під час оцінювання часу обігу пакетів і налаштування таймера повторної передачі в TCP (офіційно закріплено в RFC 6298) [86]. Запропонований сценарій аналогічний: несталі спостереження згладжуються в режимі реального часу.
4. Мінімальні обчислювальні витрати. Оновлення потребує лише поточного спостереження та одного попереднього значення – пам'ять і обчислення сталого порядку. Для бортових обчислювачів це критично [2].
5. Узгодженість із довірчими системами в ad-hoc мережах. Репутаційні механізми для мобільних ad-hoc мереж практикують «старіння» свідчень: нові події важать більше за давні. Експоненційне згладжування дає такий ефект у строгій, просто реалізованій формі [88].

Коефіцієнт α що застосовується у формулі (2.26), несе в собі наступний сенс: при $\alpha \rightarrow 1$ – швидка реакція та вища чутливість до змін; α мала – більша стійкість до шуму та повільніша адаптація. Вибір α визначає баланс між оперативністю і стабільністю. Таким чином обрана формалізація оновлення довіри в часі має такі властивості:

1. Значення завжди у $[0, 1]$
2. Поступове «старіння» давніх подій.
3. Швидке накопичення ефекту за повторюваних відхилень.
4. Низькі вимоги до пам'яті й обчислень.

Відтак виникає потреба перевести безперервну довіру $trust_i(t) \in [0, 1]$ у керовані режими поведінки системи: зменшення ваги голосу «сумнівних» вузлів та повне відключення впливу «проблемних» вузлів. Це можна зробити через два пороги та лічильники тривалості відхилень. Введемо два пороги:

$\tau_{warn} \in (0, 1)$ – поріг попередження, нижче якого вплив дрона на рішення зменшується, але не зануляється;

$\tau_{quarantine} \in (0, \tau_{warn})$ – поріг карантину, нижче якого дрон вважається проблемним, і його голос має бути відсічений після підтвердження стійкості відхилення.

Щоб уникнути «миготіння» станів через одиничні збої, застосовуємо рахунок тривалості відхилення $c_i(t)$ та бінарний індикатор карантину $q_i(t) \in \{0, 1\}$:

$$c_i(t) = \begin{cases} c_i(t-1) + 1, & \text{якщо } trust_i(t) < \tau_{quarantine} \\ \max(0, c_i(t-1) - 1), & \text{інакше} \end{cases} \quad (2.27)$$

Карантин вмикається, якщо відхилення є стійким не менше ніж протягом m кроків:

$$q_i(t) = \begin{cases} 1, & \text{якщо } c_i(t) \geq m \\ 0, & \text{інакше} \end{cases} \quad (2.28)$$

Для повернення з карантину використовується умова, жорсткіша за вхід у карантин. Наприклад, вимагається, щоб протягом r послідовних кроків довіра була вищою за $\tau_{release}$, де $\tau_{release} > \tau_{quarantine}$:

$$(trust_i(t-k) \geq \tau_{release} \forall k = 0, \dots, r-1) \Rightarrow q_i(t) = 0. \quad (2.29)$$

Таке налаштування запобігає частим перемиканням режимів і стабілізує поведінку системи. Окрім бінарного карантину, потрібне плавне зменшення впливу тих, чия довіра впала нижче τ_{warn} , але ще не досягла зони карантину. Визначимо ефективну вагу голосу $\widetilde{\omega}_i(t)$ як композицію стану карантину та безперервної функції від довіри, як це було вперше запропоновано у [25]:

$$\tilde{\omega}_i(t) = (1 - q_i(t)) \cdot f(\text{trust}_i(t)), \quad (2.30)$$

де $f: [0, 1] \rightarrow [0, 1]$ – безперервна кусочно-лінійна функція з наступним означенням [25]:

$$f(x) = \begin{cases} 0, & x \leq \tau_{quarantine} \\ \frac{\tau_{warn}}{\tau_{warn} - \tau_{quarantine}}(x - \tau_{quarantine}), & \tau_{quarantine} < x < \tau_{warn} \\ x, & x \geq \tau_{warn} \end{cases} \quad (2.31)$$

Властивості $f(x)$:

1. Нульова вага у зоні глибокого рівня недовіри $x \leq \tau_{quarantine}$ – узгоджується з карантинною логікою.
2. Плавний перехід на ділянці $(\tau_{quarantine}, \tau_{warn})$ – таким чином уникаються різкі стрибки впливу.
3. Безперервність у точці τ_{warn} – при $x = \tau_{warn}$, $f(x) = \tau_{warn}$, далі вага зростає разом із довірою.

Таким чином, навіть без ввімкнення карантину, вузол зі зниженою довірою автоматично отримує меншу вагу в наступному пункті (прийняття рішення). Відтак така модель забезпечує 3 режими для голосуючих учасників розподіленої системи:

1. Нормальний режим: $\text{trust}_i(t) \geq \tau_{warn}$, $q_i(t) = 0$ – повна участь у голосуванні, $\tilde{\omega}_i(t) = \text{trust}_i(t)$.
2. Попереджений режим: $\tau_{quarantine} < \text{trust}_i(t) < \tau_{warn}$, $q_i(t) = 0$ – зменшена вага голосу за формулою 16, ведеться спостереження стійкості відхилення.
3. Карантин: тривале падіння $\text{trust}_i(t)$ нижче $\tau_{quarantine}$ (2.28) – $\tilde{\omega}_i(t) = 0$, голос вузла не береться до уваги до виконання умови виходу з карантину (2.29).

Як результат, обраний підхід приводить до наступних властивостей та наслідків:

1. Стійкість до сплесків: одиничні збої не переводять вузол у карантин завдяки лічильнику $c_i(t)$

2. Плавна деградація впливу: ще до карантину вага поступово зменшується – зменшується ризик домінування сумнівних пропозицій.
3. Локальність і простота: усі обчислення – локальні, складність на крок $O(1)$ на дрона.
4. Керований компроміс «чутливість – стабільність»: параметри $\tau_{warn}, \tau_{quarantine}, m, r, \tau_{release}$ дають змогу налаштувати поведінку під конкретну місію.

Наступним кроком постає задача перетворити індивідуальні пропозиції дронів на спільне рішення з урахуванням їхньої довіри та станів, визначених попередньо. Логіка така: до голосування допускаються лише вузли поза карантинном, їхні голоси зважуються за довірою, а підсумкове рішення обчислюється стійким способом, щоб зменшити вплив поодиноких крайніх значень.

Отже до формування рішення допускаються лише дрони, що не перебувають у карантині:

$$V'(t) = \{i \in V : q_i(t) = 0\}. \quad (2.32)$$

Для кожного допустимого дрона використовується ефективна вага $\tilde{\omega}_i(t)$. Щоб забезпечити коректне усереднення, нормуємо їх:

$$\omega_i^*(t) = \frac{\tilde{\omega}_i(t)}{\sum_{k \in V'(t)} \tilde{\omega}_k(t)}. \quad (2.33)$$

Властивості: $\omega_i^*(t) \geq 0$ для всіх $i \in V'(t)$ та $\sum_{i \in V'(t)} \omega_i^*(t) = 1$.

В залежності від дискретності чи неперервності множини потенційних рішень обираються різні алгоритми вибору лідера. Спочатку буде розглянуто порядок вибору рішення з неперервної множини (наприклад вибір координат цільової точки польоту). Нехай кожен дрон пропонує вектор керування $u_i(t) \in \mathbb{R}^d$. Базовий зважений центр:

$$u(t) = \sum_{i \in V'(t)} \omega_i^*(t) u_i(t). \quad (2.34)$$

Щоб зменшити вплив поодиноких «віддалених» пропозицій, застосовуємо стійке відсіювання за відстанню до центру $u(t)$. Нехай $r_i(t) = \|u_i(t) - u(t)\|$.

Визначимо підмножину індексів, що проходять відбір, як ті, чиї відстані не перевищують вибраний поріг $q_{1-k}(r(t))$ – верхній $(1 - k)$ квантиль множини $\{r_i(t)\}$ для заданої частки відсікання $k \in [0, 1)$:

$$S(t) = \{i \in V'(t) : r_i(t) \leq q_{1-k}(r(t))\}. \quad (2.35)$$

Після відсіювання обчислюємо фінальний вектор рішення:

$$u^{cons}(t) = \left(\sum_{i \in S(t)} \omega_i^*(t) u_i(t) \right) / \left(\sum_{i \in S(t)} \omega^*(t) \right). \quad (2.36)$$

Таким чином якщо поодинокі дрони пропонують значення, що різко відрізняються від більшості, вони не потрапляють у список відібраних. Тобто, навіть за ненульових ваг їхній вплив обмежується процедурою відсікання. Сам факт наявності обчисленого консенсусного вектору не є достатнім для прийняття рішення до виконання. Необхідно співставити рівень підтримки рішення з поточним станом системи. Для цього вводиться поняття «Рівень довіри» (Ψ_{req}), який визначає еталонну міру консолідованої репутації, необхідну для легітимізації дій у поточний момент часу. Умова верифікації має вигляд:

$$\sum_{i \in S(t)} w_i^*(t) \geq \Psi_{req}, \quad (2.37)$$

де $\sum w_i^*(t)$ – фактична сумарна репутація групи підтримки.

У цій моделі Ψ_{req} виступає не як зовнішнє обмеження, а як міра довіри, яку система готова надати колективному рішення. Якщо фактична підтримка відповідає цій мірі (Ψ_{req}), рішення набуває статусу валідного. Це дозволяє гарантувати, що в умовах невизначеності система оперує лише тими даними, які забезпечені реальним капіталом довіри.

Якщо рішення має дискретний характер (наприклад, вибір однієї з K цілей $A = \{a_1, \dots, a_K\}$, то кожна пропозиція $u_i(t)$ ототожнюється з елементом $\hat{a}_i(t) \in A$. Для кожного варіанта накопичується зважена підтримка:

$$S(a, t) = \sum_{i \in S(t)} \omega_i^*(t) \mathbb{1}_{\hat{a}_i(t) = a}. \quad (2.38)$$

Далі, обирається найпопулярніше рішення у розподіленій системі

(переможець голосування):

$$a^{cons}(t) = \arg \max_{a \in A} S(a, t). \quad (2.39)$$

Аналогічно до неперервного випадку, рішення-переможець перевіряється на відповідність необхідному рівню довіри (Ψ_{req}):

$$S(a^{cons}, t) \geq \Psi_{req}. \quad (2.40)$$

Параметр Ψ_{req} є ключовою характеристикою методу:

1. Високе значення Ψ_{req} свідчить про те, що для збереження стійкості система потребує максимальної консолідації «безпечних» (функціонально-стабільних) вузлів.
2. Помірне значення Ψ_{req} вказує на режим роботи, де вимоги до колективних рішень є стандартними.

Якщо умова виконується, рішення $a^{cons}(t)$ вважається таким, що досягло необхідного рівня довіри, і приймається до виконання. В іншому випадку система зберігає попередній стан, мінімізуючи ризики. Таким чином, якщо зміна поведінки не набирає достатньої підтримки, ці зміни не впроваджуються. За потреби можна застосувати просте відсіювання: не враховувати голоси з найменшими вагами. Такий підхід забезпечує узгодженість та стійкість системи завдяки наступним властивостям:

1. Зменшення впливу підозрілих вузлів. Ваги $\omega_i^*(t)$ напряду залежать від довіри: якщо дрон проявляє відхилення, його вага зменшується, і навіть без карантину вплив на підсумок падає.
2. Захист від крайніх значень. Відсіювання за формулою 20 прибирає поодинокі крайні пропозиції – у безперервному випадку це запобігає «тягненню» колективного рішення у бік одиничних аномалій.
3. Плавність рішень. Оскільки ваги змінюються поступово (через динамічну довіру), підсумкове рішення $u^{cons}(t)$ або $a^{cons}(t)$ не «стрибає» від кроку до кроку за відсутності істотних нових відхилень.
4. Місцева реалізація. Усі величини обчислюються з локально доступних даних (ваги, пропозиції сусідів, відстані до центру). Це зберігає простоту

та дозволяє працювати у режимі реального часу.

2.2.5. Розрахунок динамічного коефіцієнта довіри через LSTM

Після зведення індивідуальних пропозицій у колективне рішення виникає потреба підвищити чутливість системи до послідовних у часі відхилень поведінки окремих дронів, зберігаючи простоту реалізації та малу обчислювальну вартість. Щоб урахувати часові закономірності за межами статичних ознак, використано рекурентну нейронну мережу архітектури *LSTM* (*Long Short-Term Memory*) (рис. 2.2). На відміну від простих рекурентних мереж (*RNN*), *LSTM* здатна ефективно зберігати інформацію про давні події завдяки механізму шлюзів, що дозволяє їй відрізняти випадкові короточасні збої від системних, розтягнутих у часі маніпуляцій. Мережа працює на послідовностях нормованих ознак і повертає оцінку корисності потенційного рішення (коефіцієнт $\theta_i(t)$).

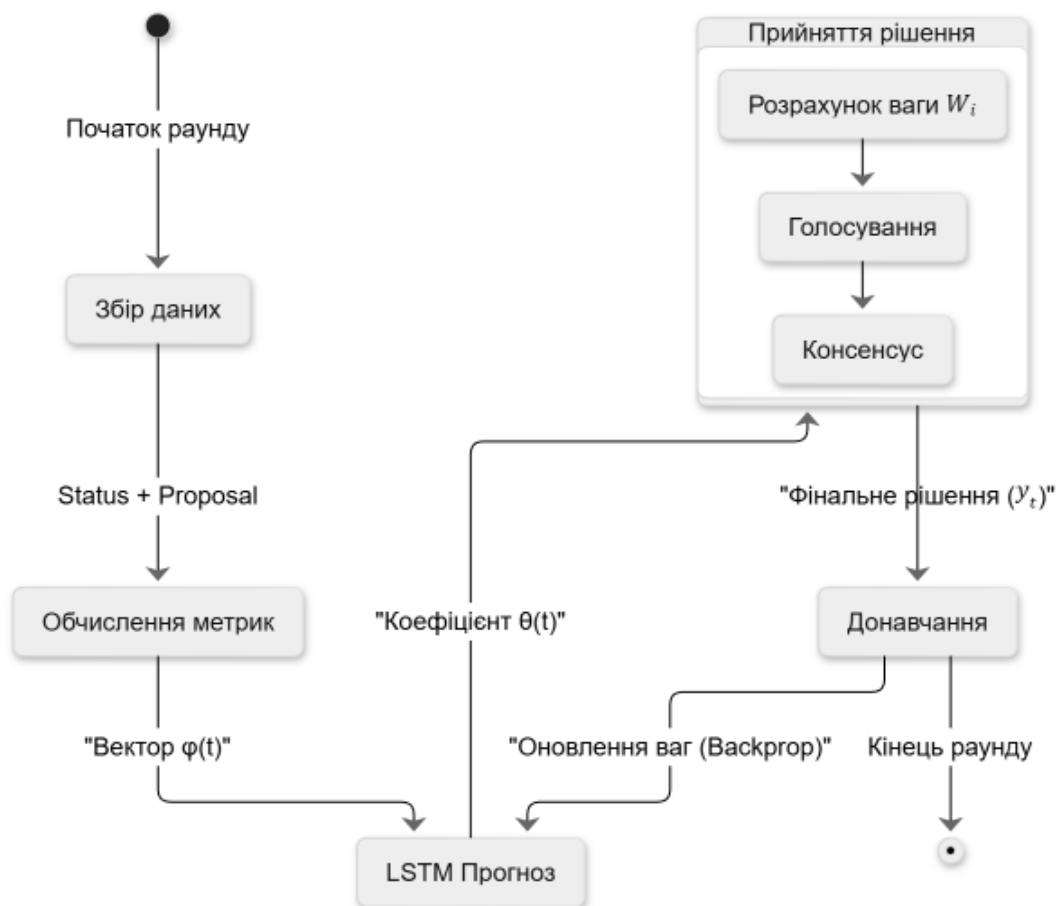


Рис. 2.2 Інтеграція в контур керування

Для кожного сусіда i на дискретному кроці часу t формується вектор ознак:

$$\phi_i(t) = [s_{pos}(i, t), s_{cmd}(i, t), s_{tim}(i, t), E_i(t), d_i(t)]^T \in [0, 1]^5, \quad (2.41)$$

тут $s_{pos}, s_{cmd}, s_{tim}$ – метрики узгодженості положень, команд і регулярності повідомлень; $E_i(t)$ – рівень заряду; $d_i(t)$ – нормалізований ступінь вузла. Усі компоненти вже належать діапазону $[0, 1]$, що спрощує навчання та інтерпретацію.

Архітектура *LSTM*-комірки описується системою рівнянь, що регулюють потік інформації через стан комірки $C_i(t)$ та прихований стан $h_i(t)$.

1. Шлюз забування (*Forget Gate*): Визначає, яку частину попередньої пам'яті слід відкинути:

$$f_i(t) = \sigma(W_f \cdot [\phi_i(t), h_i(t-1)] + b_f). \quad (2.42)$$

2. Шлюз входу (*Input Gate*): Визначає, яку нову інформацію слід записати в пам'ять:

$$k_i(t) = \sigma(W_k \cdot [\phi_i(t), h_i(t-1)] + b_k). \quad (2.43)$$

$$\tilde{C}_i(t) = \tanh(W_c \cdot [\phi_i(t), h_i(t-1)] + b_c). \quad (2.44)$$

3. Оновлення стану комірки: Комбінація збереженої частини старої пам'яті та нових кандидатурних значень:

$$C_i(t) = f_i(t) \cdot C_i(t-1) + k_i(t) \cdot \tilde{C}_i(t). \quad (2.45)$$

4. Шлюз виходу (*Output Gate*) та прихований стан: Визначає фінальний вихід комірки, що базується на відфільтрованому стані пам'яті:

$$o_i(t) = \sigma(W_o \cdot [\phi_i(t), h_i(t-1)] + b_o). \quad (2.46)$$

$$h_i(t) = o_i(t) \cdot \tanh(C_i(t)). \quad (2.47)$$

Тут σ – логістична сигмоїда, W та b – навчувані вагові матриці та вектори зміщення для відповідних шлюзів.

Внутрішній прихований стан трансформується у скалярне значення в інтервалі $[0, 1]$ (оцінка «нормальності» або нейромережевий коефіцієнт довіри) за допомогою вихідного шару:

$$\theta_i(t) = \sigma(W_\theta h_i(t) + b_\theta). \quad (2.48)$$

Ключова особливість запропонованого методу полягає у відмові від використання фіксованих констант для процедури валідації рішень. Замість

статичного порогу, неймережа керує глобальним параметром системи – необхідним динамічним рівнем довіри (Ψ_{req}). Зв'язок між виходом LSTM-мережі $\theta(t)$ та параметром валідації встановлюється наступним чином:

$$\Psi_{req}(t) = \theta(t). \quad (2.49)$$

Така заміна статичного значення на динамічний змінний параметр $\theta(t)$ забезпечує адаптивність системи прийняття рішень до поточних умов середовища:

1. У стабільному стані: Коли метрики узгодженості $\phi_i(t)$ є високими і стабільними, LSTM генерує помірні значення $\theta(t)$. Це знижує вимоги до $\Psi_{req}(t)$, дозволяючи системі швидко приймати рішення та підтримувати високу пропускну здатність, не блокуючи легітимні дії через надмірну обережність.
2. У стані атаки: При виявленні аномалій у часових рядах (наприклад, різка зміна геолокації або порушення таймінгів, характерні для скоординованих атак), значення $\theta(t)$ зростає, наближаючись до 1. Це автоматично підвищує планку $\Psi_{req}(t)$. В результаті система блокує виконання будь-яких рішень, які не мають абсолютної консолідованої підтримки, ефективно переходячи в режим нульової довіри.

Таким чином, параметр $\theta(t)$ виступає прямим регулятором «імунітету» розподіленої системи, роблячи її стійкою до динамічних загроз без необхідності ручного налаштування порогових значень.

Для покрокового навчання використовується «слабке» маркування на основі узгодженості пропозиції сусіда з колективним рішенням. Нехай $u_i(t)$ – індивідуальна векторна пропозиція керування, $u_{const}(t)$ – прийняте колективне векторне рішення; у дискретному випадку $a_i(t)$ та $a_{const}(t)$ – відповідно індивідуальний і колективний вибір варіанту. Тоді визначається мітка:

$$y_i(t) = \begin{cases} 1, & \text{якщо } \|u_i(t) - u^{cons}(t)\| \leq \varepsilon_u \text{ (безперервний випадок)} \\ & \text{або } \hat{a}_i(t) = a^{cons}(t) \text{ (дискретний)} \\ 0, & \text{інакше} \end{cases}. \quad (2.50)$$

Параметр $\varepsilon_u > 0$ задає допустиму близькість у просторі керувань. Відповідність оцінки $\theta_i(t)$ (або $s'_i(t)$) і мітці $y_i(t)$ оцінюється стандартною крос-

ентропійною функцією втрат (log-loss):

$$L_i(t) = -y_i(t) \ln(\theta_i(t)) - (1 - y_i(t)) \ln(1 - \theta_i(t)). \quad (2.51)$$

Крос-ентропійна функція втрат (log-loss) є негативним логарифмом бернуллівської правдоподібності й загальноприйнятим критерієм для бінарної класифікації. Така втрата забезпечує коректну інтерпретацію $\theta_i(t)$ як імовірнісної оцінки, має прості градієнти та добре працює в онлайн-сценаріях з малими кроками оновлення. Повний контур та загальна схема покрокової обробки даних із застосуванням нейромережі LSTM схематично зображені на рис. 2.3.

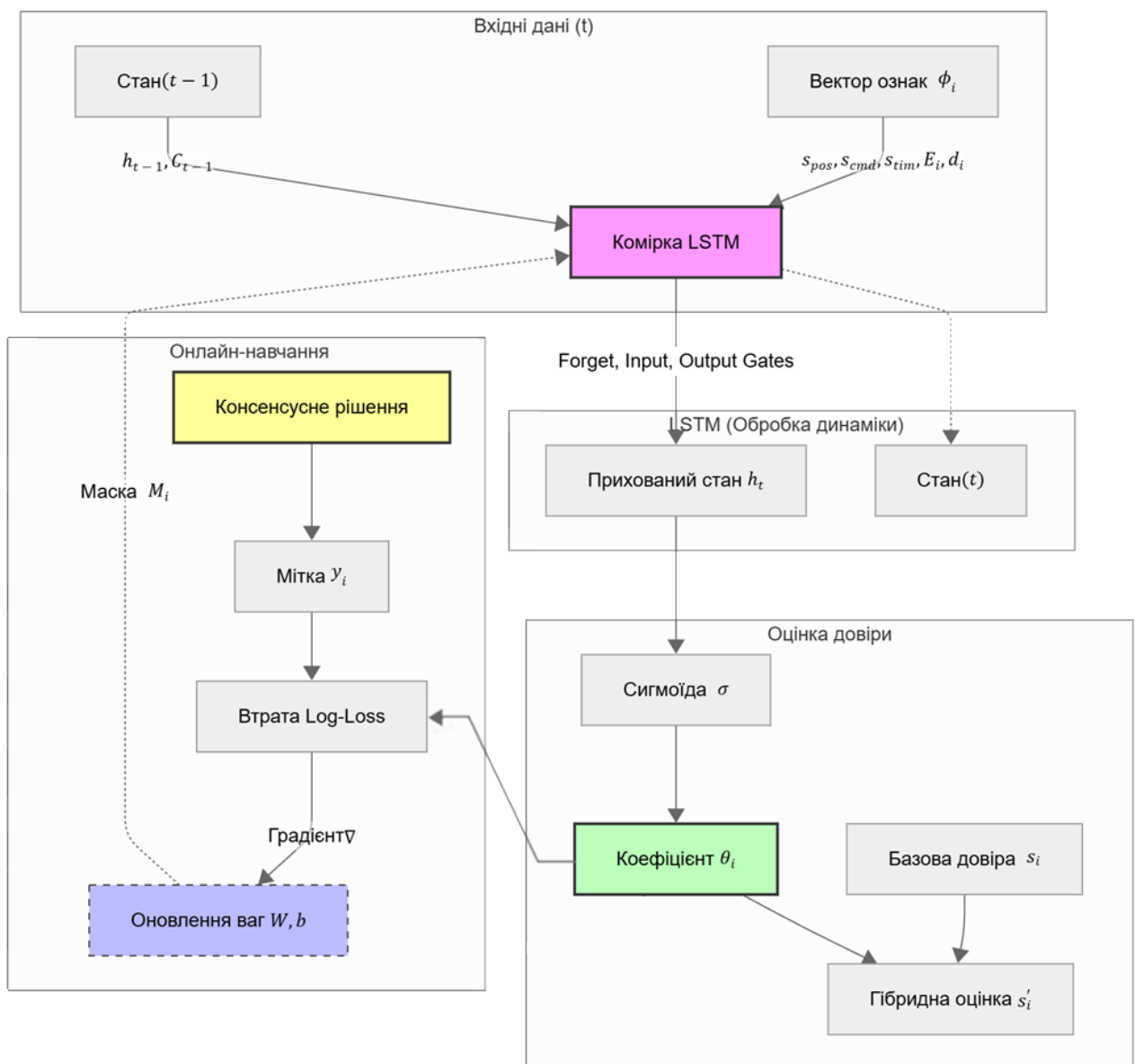


Рис. 2.3 Загальна схема обробки даних із застосуванням нейромережі LSTM

Параметри мережі (матриці W та вектори b) коригуються малими кроками градієнтного спуску зі сталою швидкістю навчання η , що гарантує плавність змін і стабільність виходу. Щоб запобігти отруєнню даними від підозрілих вузлів, оновлення дозволяється лише за умови достатньої довіри та відсутності карантину:

$$M_i(t) = I(\text{trust}_i(t) \geq \tau_{\text{warn}} \wedge q_i(t) = 0), \quad (2.52)$$

де $\text{trust}_i(t)$ – довіра, τ_{warn} – поріг попередження, $q_i(t)$ – індикатор карантину. Якщо $M_i(t) = 0$, параметри не оновлюються на кроці t ; якщо $M_i(t) = 1$, виконується стандартний малий крок навчання. Така маска узгоджується з вимогами безпеки та не дозволяє «навчати» модель даними із сумнівних джерел.

Після запровадження динамічного рівня довіри та вагового зведення варто зафіксувати рамкові умови, у яких метод коректно працює. Метод орієнтовано на сценарії з підтримуваною зв'язністю для обміну пропозиціями, помірною часткою відхилених учасників, стабільною моделлю затримок «кількість хопів $\rightarrow$ час» і можливістю формувати «слабкі» мітки узгодженості на основі колективного рішення. Застосовність знижується, якщо виникає велика узгоджена, але хибна коаліція (*Sybil attack*), яку не виявляє карантин і не прибирає відсікання.

Потенційні джерела викривлень включають: невдалий вибір порогів і частки відсікання; навчальні мітки низької якості; залежність від сценарію місії (покриття, щільність перешкод, інтенсивність втрат пакетів); чутливість до початкових умов (розташування, заряди).

2.3. Метод вибору лідера на основі інтегрованої довіри та нейромереж

2.3.1. Метрики ефективності комунікації

Припущення моделі зв'язку. Радіосигнал вважається таким, що поширюється миттєво в межах порогу R ; затримки обумовлені лише кількістю проміжних переходів (хопів) на маршруті між відправником та отримувачем. Така модель дозволяє параметризувати часову поведінку єдиною сталою $\beta > 0$ – середньою затримкою одного хопа (передача і підтвердження), яку буде використано у наступних формулах. Для формального означення маршрутів позначимо через $P_{i \rightarrow j}(t)$ множину всіх допустимих шляхів у графі G_t від вузла i до вузла j , а через

$H(p)$ – кількість хопів на шляху $p \in P_{i \rightarrow j}(t)$. Мінімальна кількість хопів між i та j – це стандартна графова відстань за числом ребер:

$$\delta_t(i, j) = \min_{p \in P_{i \rightarrow j}(t)} H(p). \quad (2.53)$$

Ця величина прямо входить у визначення часу доставки повідомлень що буде описано далі. Нарешті, для задачі вибору лідера важливо мати зв'язність графа в операційному стані: припускаємо, що підграф, індукований E_t , є зв'язним, тобто $\delta_t(i, j) < \infty$ для будь яких $i \neq j$. Це забезпечує існування маршруту доставки до майбутнього координатора та коректність агрегованих часових характеристик.

Ефективність комунікації в розподіленій системі рою безпілотних літальних апаратів визначається не лише структурою графа зв'язків G_t , а й тим, наскільки швидко повідомлення може передаватися між вузлами. Навіть якщо два дрони формально з'єднані маршрутом, практична користь такого зв'язку залежить від часу доставки повідомлень, який безпосередньо впливає на швидкість досягнення консенсусу.

Запропонована модель рою дронів ґрунтується на припущенні, що єдине джерело затримки – це час, витрачений на проходження кожного хопу маршруту. Кожен хоп складається з двох фаз:

1. передача пакета від одного дрона до іншого;
2. підтвердження отримання (АСК), що повертається назад.

Таким чином, середня затримка одного хопу $\beta > 0$ включає сумарний час необхідний для цих двох фаз. Фізично вона залежить від швидкості модуляції, довжини кадру, потужності передавача та обчислювальної здатності мікроконтролерів дронів, однак у моделі ці фактори агрегуються в єдину константу. Це дозволяє значно спростити аналітичний опис і працювати з абстрактним параметром β , який у подальших симуляціях можна задати як вимірюваний або емпірично оцінений.

Враховуючи вищеописане можна перейти до математичного представлення часу доставки маршрутом. Нехай p – маршрут у графі G_t , який сполучає дрона-відправника та дрона-отримувача. Якщо цей маршрут має $H(p)$ хопів, то затримка

визначається як

$$T(p) = H(p) \cdot \beta. \quad (2.54)$$

Ця залежність є лінійною: кожен додатковий хоп збільшує час доставки на величину β . У реальних системах можуть бути варіації внаслідок перешкод або повторних передач, але у запропонованій математичній моделі враховується лише середній ефект. Виходячи з цього можна виразити формулу часу доставки між двома дронами. Між будь-якою парою вузлів $i, j \in V$ може існувати кілька маршрутів $p \in P_{i \rightarrow j}(t)$. Прагнучи досягти раціональної та оптимальної маршрутизації, практична доставка здійснюватиметься найшвидшим маршрутом, тобто тим, де кількість хопів є мінімальною:

$$T_{ij}(t) = \beta \cdot \min_{p \in P_{i \rightarrow j}(t)} H(p) = \beta \cdot \delta_t(i, j), \quad (2.55)$$

де $\delta_t(i, j)$ – графова відстань між вузлами i та j , виміряна у хопах.

Таким чином, час доставки $T_{ij}(t)$ залежить не від геометричної відстані між дронами, а від структури топології у момент часу t . Два дрони, що знаходяться фізично близько, можуть мати великий T_{ij} , якщо між ними немає прямого каналу, і навпаки – далекі геометрично вузли можуть з'єднуватися через малу кількість хопів. Для аналізу поведінки окремих дронів та всього рою вводяться середні характеристики:

Середній час доставки для конкретного дрона:

$$T_i^{avg}(t) = \frac{1}{N-1} \sum_{j \neq i} T_{ij}(t). \quad (2.56)$$

Цей показник відображає, наскільки швидко дрон i у середньому може доставити повідомлення до решти учасників системи. Він є локальною характеристикою кандидата у лідери й напряму входить у подальші функції корисності.

Глобальний середній час доставки для всього рою:

$$\bar{T}(t) = \frac{1}{N(N-1)} \sum_{i \neq j} T_{ij}(t). \quad (2.57)$$

Це вже інтегральна характеристика, яка описує «середню швидкодiю»

комунікаційної мережі у момент часу t . Вона зручна для визначення опорних значень.

Граничний час доставки (діаметр рою за затримкою):

$$T^{diam}(t) = \max_{i,j \in V, i \neq j} T_{ij}(t). \quad (2.58)$$

Ця величина відповідає найгіршому випадку й вказує, скільки часу може знадобитися для доставки повідомлення у найбільш віддалену частину рою. Введені співвідношення дозволяють пояснити, чому час доставки є ключовим критерієм:

1. Формула (2.54) задає елементарну модель: кожен хоп збільшує затримку на β .
2. Формула (2.55) формалізує час доставки між будь-якими двома дронами через мінімальну кількість хопів.
3. Формула для (2.56) $T_i^{avg}(t)$ надає інтегральну оцінку «комунікаційної доступності» дрона i .
4. Глобальні показники $\bar{T}(t)$ та $T^{diam}(t)$ (2.57, 2.58) дозволяють аналізувати швидкодію рою як єдиного цілого та служать базою для нормалізації.

Отже, на цьому етапі можна зробити висновок: дрон, у якого значення T_i^{avg} є мінімальним серед усіх вузлів, забезпечує найменші середні затримки у розподіленій системі. Саме тому ця величина є природним кандидатом для включення до функції корисності при виборі лідера.

Наступним кроком після введення у модель часу доставки повідомлень між окремими вузлами $T_{ij}(t)$ та середнього часу для конкретного дрона $T_i^{avg}(t)$, постає завдання визначити такі інтегральні характеристики, які б дозволяли об'єктивно оцінювати кожного дрона як потенційного координатора.

Для дрона $i \in V$ визначимо дві основні величини:

$$T_i^{max}(t) = \max_{j \neq i} T_{ij}(t), \quad T_i^{avg}(t) = \frac{1}{N-1} \sum_{j \neq i} T_{ij}(t), \quad (2.59)$$

де T_i^{max} – максимальний час доставки повідомлення від дрона i до будь-якого іншого вузла системи. T_i^{avg} – середній час доставки повідомлення від дрона i до

решти учасників рою. Таким чином T_i^{avg} відображає «типову» затримку, яку створює дрон i при взаємодії з іншими. Чим менше значення, тим швидше цей дрон у середньому може поширювати інформацію. В свою чергу T_i^{max} показує «найгірший випадок»: наскільки довго повідомлення може йти від дрона i до найбільш віддаленого учасника. Це критично для консенсусу, адже узгодження вимагає досягнення всіх вузлів.

У поєднанні ці дві величини описують баланс між середньою ефективністю та найгіршими сценаріями. Між T_i^{avg} та T_i^{max} завжди виконується співвідношення

$$T_i^{avg}(t) \leq T_i^{max}(t). \quad (2.60)$$

причому рівність досягається лише у випадку, коли всі маршрути від дрона i до інших мають однакову довжину. У реальному рої таке трапляється рідко, тому зазвичай середнє значення менше за максимальне. Це означає, що дрон із малим середнім часом доставки ще не гарантує низький максимум, і навпаки. Тому для оцінки придатності кандидата варто враховувати обидва показники.

Для подальшого формування функції корисності ключовим є те, що:

- значення T_i^{avg} безпосередньо показує ефективність дрона як відправника інформації;
- значення T_i^{max} визначає, наскільки «вузьким місцем» стане цей дрон при найгірших маршрутах.

У системах, де критично важлива гарантована доставка до всіх учасників, більшого значення надається саме T_i^{max} . У випадках, коли акцент робиться на загальній швидкодії, головним критерієм стає T_i^{avg} .

Таким чином, введення агрегованих часових характеристик дозволяє перейти від локальних параметрів (наприклад, кількість сусідів чи рівень енергії) до метрик, які напряду описують часову ефективність усього рою при взаємодії через конкретного дрона. Використання середнього часу доставки як критерію мінімізації гарантує, що обраний лідер знаходитиметься у «комунікаційному центрі» групи. Це забезпечує найшвидше розповсюдження керуючих команд та збір телеметрії, що є критично важливим для підтримки аеродинамічної

стабільності строю та реакції на раптові перешкоди. З точки зору теорії графів, такий підхід мінімізує зважений діаметр мережі, оптимізуючи її продуктивність.

Однак, орієнтація виключно на фізичні параметри каналу зв'язку та енергетичний ресурс приховує в собі фундаментальну вразливість, пов'язану з проблемою безпеки. Висока швидкість доставки повідомлень є необхідною умовою ефективності, але вона не є гарантією достовірності даних, що передаються. У середовищі з можливістю візантійських відмов виникає ризик появи «швидкого зловмисника»: скомпрометований вузол може мати ідеальний канал зв'язку, повний заряд батареї та центральне розташування, що формально робить його найкращим кандидатом у лідери за критеріями якості обслуговування. Якщо алгоритм вибору базується лише на мінімізації затримок, такий вузол гарантовано отримає повноваження координатора і зможе використовувати свою швидкісну перевагу для максимально швидкого поширення дезінформації або хибних команд, що призведе до дезорганізації місії раніше, ніж система встигне зреагувати.

Ця ситуація демонструє обмеженість суто технократичного підходу до керування в умовах активної протидії. Оптимізація метрик якості обслуговування без перевірки семантичної коректності поведінки створює вектор атаки, де зловмисник може маніпулювати виборами, просто покращуючи свої комунікаційні параметри. Отже, для побудови стійкої системи керування необхідно доповнити метрики фізичної ефективності (енергія, час, топологія) метрикою функціональної стабільності, яка б характеризувала ступінь довіри до змісту дій кандидата. Це обґрунтовує необхідність модифікації функції корисності лідера шляхом введення динамічного коефіцієнта довіри, розрахованого у попередньому методі. Такий коефіцієнт має виступати не просто додатковим доданком, а мультиплікативним фільтром («гейтом»), здатним повністю заблокувати обрання технічно досконалого, але скомпрометованого вузла.

2.3.2. Формування комплексної вагової функції

У класичних підходах до вибору координатора у розподілених системах

розглядаються переважно енергетичні та топологічні характеристики дронів. Ідея полягає в тому, що лідером має стати вузол, який:

має найбільший запас енергії (щоб довше виконувати роль координатора);
знаходиться у «центрі» топології (тобто має найбільше сусідів у межах радіуса зв'язку).

Формалізуємо далі ці характеристики. Кількість сусідів дрона i у момент часу t позначається як $d_i(t)$. Щоб зробити цю величину порівнянною між різними дронами, її необхідно нормалізувати на максимальне значення в рої:

$$\tilde{d}_i = \frac{d_i}{d_{max}}, d_{max} = \max_{k \in V} d_k. \quad (2.61)$$

Таким чином, $\tilde{d}_i \in [0, 1]$, де 1 відповідає дрону з найбільшою кількістю сусідів, а значення, близькі до 0, вказують на «окраїну» топології.

У свою чергу, заряд батареї дрона позначається як $E_i \in [0, 1]$, де 1 – повністю заряджена батарея, а 0 – повне вичерпання ресурсу. Це дозволяє порівнювати дрони незалежно від їхньої фактичної ємності батарей.

Поряд із фізичними характеристиками, критично важливим для безпечного функціонування рою є урахування рівня функціональної стабільності вузла, який не може бути виміряний прямими апаратними сенсорами, але обчислюється аналітично. Відповідно до розробленого у підрозділі 2.2 методу узгодження рішень, кожному агенту i у момент часу t присвоюється значення динамічного коефіцієнта довіри:

$$Trust_i^{final}(t) \in [0, 1]. \quad (2.62)$$

Ця величина є інтегральною оцінкою, що агрегує історію узгодженості команд, просторову поведінку та вердикт нейромережевого регулятора. Значення 1 свідчить про повну передбачуваність та доброчесність дій дрона, тоді як значення, що прямують до 0, сигналізують про виявлені аномалії або ідентифікацію вузла як джерела дезінформації. Введення цього параметру у вхідний вектор моделі дозволяє перейти від суто технократичної оцінки кандидата (за ресурсами) до комплексної, що включає безпекову складову.

У класичних підходах до вибору координатора функція корисності

традиційно будується як лінійна комбінація параметрів. Базовий критерій вибору лідера поєднує енергетичний та топологічний параметри у такій функції:

$$U_i^{base} = \omega_1 E_i + \omega_2 d_i, \text{де } \omega_1 + \omega_2 = 1, \omega_k \geq 0, \quad (2.63)$$

тут E_i – нормалізований енергетичний внесок, d_i – топологічний внесок, а вагові коефіцієнти ω_1, ω_2 задають пріоритети місії. Їхнє співвідношення інтерпретується наступним чином:

1. Якщо $\omega_1 > \omega_2$, вибір координатора фокусується на довговічності: дрон із найбільшою батареєю отримає перевагу навіть за відносно слабкої позиції в мережі.
2. Якщо $\omega_2 > \omega_1$, вирішальною стає комунікаційна «центральність»: лідером обирається той, хто має найбільше сусідів.
3. При $\omega_1 = \omega_2 = 0.5$ параметри є рівнозначними.

Однак, як було показано вище, ігнорування часової складової може призвести до обрання «повільного» лідера. В рамках удосконалення методу вибору лідера рою [23], для створення базису, що дозволить у майбутньому ізолювати підозрілі вузли без порушення загальної координації системи, базова функція була модифікована шляхом введення метрики середнього часу доставки.. Для цього визначається опорне значення $T_{ref}(t)$ як середнє арифметичне часів доставки всіх вузлів у рої. Вдосконалена адитивна функція має вигляд:

$$U_i^{ref} = \omega_1 E_i + \omega_2 d_i - \omega_3 \frac{T_i^{avg}}{T_{ref}}, \sum \omega_k = 1, \quad (2.64)$$

тут доданок $-\omega_3 \frac{T_i^{avg}}{T_{ref}}$ є штрафом за затримку: дрон, який повільніший за середній рівень по рою ($T_i^{avg} > T_{ref}$), втрачає бали корисності. Це дозволяє балансувати між енергією, зв'язністю та швидкістю.

Проте, в умовах функціонування у ворожому середовищі, адитивна природа обох наведених функцій (U_i^{base} та U_i^{ref}) створює критичну вразливість. Лінійна комбінація дозволяє здійснювати компенсацію низьких значень одних параметрів за рахунок високих значень інших. Скомпрометований вузол (наприклад, захоплений зловмисником) може мати повний заряд батареї ($E_i \approx 1$) та

знаходиться у центрі групи ($d_i \approx 1$). Навіть якщо система має механізм оцінки довіри і ми спробуємо просто додати її як четвертий доданок ($+\omega_4 Trust_i$), високі фізичні показники «зрадника» можуть математично перекрити штраф за низьку довіру.

Для усунення цього ризику необхідно перейти від адитивної логіки до мультиплікативної, де параметр довіри $Trust_i^{final}(t)$, обчислений у попередньому підрозділі, виступає не як доданок, а як фільтруючий множник (гейт). Вперше математичну модель такого мультиплікативного підходу було запропоновано та досліджено здобувачем у роботі [24]. Відповідно до неї, комплексна функція ваги набуває вигляду:

$$W_i(t) = \left(\omega_1 E_i(t) + \omega_2 d_i(t) + \omega_3 \left(1 - \frac{T_i^{avg}(t)}{T_{max}(t)} \right) \right) \cdot \left(Trust_i^{final}(t) \right)^\gamma, \quad (2.65)$$

де $\gamma \geq 1$ – адаптивний параметр суворості фільтрації. У такій моделі довіра визначає валідність фізичних характеристик кандидата. Якщо рівень довіри до вузла наближається до нуля, то весь вираз $W_i(t)$ також прямує до нуля, незалежно від того, наскільки ідеальними є заряд чи якість каналу. Це реалізує принцип пріоритету безпеки: технічна ефективність враховується лише для тих учасників, які пройшли перевірку на доброчесність. У запропонованому виразі перший множник, окремі аспекти якого висвітлені автором у [23] відповідає за фізичну ефективність (QoS) кандидата. Він агрегує енергетичні та комунікаційні характеристики, забезпечуючи вибір найкращого серед «чесних». Ключовою особливістю тут є інверсія часової метрики для узгодження напрямку оптимізації (максимізація):

$$QoS_i(t) = \omega_1 E_i(t) + \omega_2 d_i(t) + \omega_3 \left(1 - \frac{T_i^{avg}(t)}{T_{max}(t)} \right), \quad (2.66)$$

де:

$E_i(t) \in [0,1]$ – нормалізований заряд батареї;

$d_i(t) \in [0,1]$ – нормалізований ступінь вузла (показник центральності);

$\left(1 - \frac{T_i^{avg}(t)}{T_{max}(t)} \right)$ – показник відносної швидкодії: чим менша середня затримка

вузла порівняно з найгіршою в рої (T_{max}), тим ближче цей доданок до 1.

$\omega_1, \omega_2, \omega_3$ – вагові коефіцієнти, що відображають пріоритети місії ($\sum \omega_k = 1$).

Другий множник – це функція штрафу, що базується на довірі. Як було обґрунтовано автором у праці [24], з метою ситуативного посилення ваги репутації вузлів під час взаємодії. Ключовою інновацією методу є динамічна адаптація експоненти γ . Вона визначається як лінійна функція від виходу рекурентної нейронної мережі:

$$\gamma(t) = a \cdot \theta(t), \quad (2.67)$$

де:

a – константа масштабування, що задає максимальну «жорсткість» системи;

$\theta(t)$ – оцінка нормальності поведінки мережі, що обчислюється *LSTM*-мережею відповідно до формули (2.46).

Такий зв'язок дозволяє системі змінювати чутливість до підозрілої поведінки в реальному часі. При стабільному стані мережі ($\theta(t)$ помірно, $\gamma \approx 1$) залежність є близькою до лінійної, що дозволяє ефективним вузлам ставати лідерами. При виявленні аномалій ($\theta(t)$ зростає, $\gamma > 1$) функція штрафу стає суперадитивною. Це забезпечує різку реакцію на найменше відхилення довіри., що робить вузол неконкурентоспроможним порівняно з агентами, які мають довіру ≈ 1.0 , навіть якщо їхні фізичні параметри дещо гірші. Як приклад розглянемо вузол із незначним зниженням репутації до рівня 0.8 (підозра на компрометацію). Якщо нейромережа детектує загрозу і підвищує $\theta(t)$ так, що $\gamma = 3$, множник довіри складе $0.8^3 = 0.512$. Таке падіння фінальної ваги майже вдвічі робить даний вузол неконкурентоспроможним у порівнянні з агентами, які мають довіру $Trust \approx 1.0$, навіть якщо їхні фізичні параметри є об'єктивно гіршими. Це дозволяє превентивно виключати підозрілі вузли з процесу виборів ще до моменту, коли їхня шкода стане критичною.

Окремо формалізується умова жорсткого блокування (*Hard Gate*), яка пов'язує цей метод із механізмом карантину з попереднього підрозділу:

$$W_i(t) = \begin{cases} -\infty, & \text{якщо } q_i(t) = 1 \text{ (карантин)} \\ QoS_i(t) \cdot (Trust_i^{final}(t))^\gamma, & \text{інакше} \end{cases}. \quad (2.68)$$

Така структура гарантує, що вузол, ідентифікований нейромережею як зловмисний ($q_i = 1$), виключається з процесу виборів миттєво, а вузол із «підмоченою» репутацією (низький *Trust*) отримує суттєвий штраф, який неможливо компенсувати повним зарядом батареї. Таким чином, розроблена мультиплікативна функція $W_i(t)$ забезпечує математично гарантований пріоритет безпеки над ефективністю. Введена структура діє як дворівневий фільтр:

1. Жорсткий фільтр (*Hard Gate*): Виключає з множини кандидатів вузли, що перебувають у карантині ($W_i \rightarrow -\infty$), унеможливаючи обрання відверто деструктивних агентів.
2. М'який фільтр (*Soft Gate*): Через степеневу залежність від довіри $Trust^V$ суттєво знижує конкурентоспроможність вузлів із нестабільною поведінкою, навіть якщо вони мають високий заряд батареї та ідеальний канал зв'язку.

Це дозволяє формалізувати задачу вибору лідера як пошук оптимуму в просторі «довірених» рішень, де метрики якості обслуговування (QoS) відіграють роль ранжування лише серед функціонально-стабільних учасників. Отримане значення $W_i(t)$ стає єдиним скалярним критерієм для наступного етапу – процедури розподіленого голосування.

2.3.3. Процедура виборів та стабільність кластера

Після обчислення комплексної функції ваги $W_i(t)$ кожним учасником рою, система переходить до фази узгодження єдиного координатора. Оскільки розроблений метод орієнтований на роботу в умовах нестабільного зв'язку та можливих зловмисних дій, процедура виборів будується за принципом децентралізованого порівняння з механізмом стримування (гістерезисом) для запобігання частій зміні лідера («миготінню»).

Інформація про поточне значення ваги $W_i(t)$ додається до періодичних службових повідомлень (*Heartbeat*), які дрони транслюють у фазі розсилки станів. Це дозволяє уникнути генерації окремого трафіку для виборів. Кожен вузол i підтримує локальну таблицю ваг сусідів.

$$\mathcal{W}_i(t) = \{(j, W_j(t)) \mid j \in N_i(t) \cup \{i\}\}. \quad (2.69)$$

Ваги, отримані від вузлів, що перебувають у карантині, ігноруються або замінюються на $-\infty$. У стаціонарному режимі (коли лідер ще не обраний або втрачений), лідером вважається вузол з максимальною задекларованою вагою серед доступних сусідів:

$$Leader(t) = \operatorname{argmax}_{j \in \mathcal{W}_i(t)} W_j(t). \quad (2.70)$$

Для забезпечення детермінованості у випадку рівності ваг ($W_j = W_k$) пріоритет надається вузлу з меншим ідентифікатором (ID).

В умовах високої динаміки значення $W_i(t)$ можуть флуктувати через незначні зміни заряду батареї або переміщення дронів. Якщо застосовувати правило простого максимуму, це призведе до ефекту «ring-pong», коли роль лідера постійно переходить від одного дрона до іншого, що дестабілізує керування. Щоб запобігти цьому, вводиться умова перемикавання з порогом гістерезису $\Delta W > 0$. Нехай $L(t-1)$ – поточний лідер. Вузол i визнає нового лідера k ($k \neq L$) тільки за виконання умови:

$$W_k(t) > W_{L(t-1)}(t) + \Delta W, \quad (2.71)$$

де ΔW – параметр інерційності системи. Ця нерівність створює «зону комфорту» для чинного лідера: він зберігає свої повноваження, навіть якщо його показники трохи погіршилися або з'явився кандидат із незначною перевагою. Зміна координатора відбувається лише тоді, коли перевага нового кандидата стає суттєвою (наприклад, перевищує 5-10% від поточного максимуму) або коли довіра до поточного лідера падає настільки, що його вага W_L різко зменшується (через мультиплікативний множник довіри). Такий підхід забезпечує стійкість кластера: система реагує на серйозні зміни (втрата довіри, критичний розряд, вихід із ладу), але ігнорує дрібні шумові флуктуації метрик.

2.4. Оцінка ресурсних витрат та стійкості

2.4.1. Аналіз обчислювальної складності

Критичною вимогою до алгоритмів керування роєм БПЛА є можливість їх

виконання на бортових обчислювачах з обмеженими ресурсами (мікроконтролерах класу *STM32* або одноплатних комп'ютерах типу *Raspberry Pi*) у реальному часі. Для підтвердження відповідності розробленого методу цим вимогам проведемо оцінку асимптотичної часової складності (*Time Complexity*) виконання однієї ітерації алгоритму для окремого вузла i .

Нехай N – загальна кількість вузлів у рої, а $k_i = |N_i(t)|$ – кількість безпосередніх сусідів вузла i (ступінь вузла). У щільних роях зазвичай $k_i \ll N$.

1. Оцінка складності Методу узгодження рішень (Блок довіри)

Ядром першого методу є рекурентна нейронна мережа (*LSTM*), яка обчислює коефіцієнт довіри для кожного сусіда.

Вхідні дані: Вектор ознак $\phi_j(t)$ має фіксовану розмірність $d = 5$ (узгодженість позиції, команд, часу, заряд, ступінь).

Оскільки розмірність вхідного вектора та параметри моделі є фіксованими величинами, які не залежать від розміру рою, складність обробки даних одного сусіда є константою $O(1)$. Відповідно, для оновлення довіри до всіх k_i сусідів складність лінійно залежить від їх кількості:

$$T_{trust} = O(k_i). \quad (2.72)$$

У всебічному огляді архітектур *LSTM* [47] підтверджують, що обчислювальна вартість прямого проходу мережі є детермінованою і залежить лише від кількості параметрів. Для задач навчання на граничних пристроях (*Edge Learning*), як зазначають [17], використання оптимізованих моделей дозволяє виконувати інференс за константний час, що є критичним для забезпечення роботи в реальному часі на автономних платформах. Це вигідно відрізняє запропонований підхід від складних імовірнісних моделей, де час розрахунку може суттєво зростати зі збільшенням історії спостережень.

2. Оцінка складності Методу вибору лідера

Другий метод базується на модифікованому алгоритмі зваженої кластеризації. Процедура складається з трьох кроків:

1. Обчислення власної ваги W_i : Включає арифметичні операції над локальними параметрами (заряд, ступінь) та агрегування метрик часу

доставки. Складність: $O(1)$.

2. Обмін та збір ваг: Вузол отримує повідомлення від k_i сусідів. Десеріалізація та збереження в локальну таблицю має складність $O(k_i)$.
3. Визначення переможця: Пошук максимуму в масиві ваг розміром $k_i + 1$ (власна + сусідів). Складність лінійного пошуку: $O(k_i)$.

Дослідження [8], присвячене аналізу живучості мереж БПЛА на основі динамічної кластеризації, підтверджує, що введення додаткових вагових коефіцієнтів (у нашому випадку – мультиплікатора довіри) додає лише фіксовану кількість арифметичних операцій. Це не змінює клас складності алгоритму, який залишається лінійним відносно щільності сусідів:

$$T_{elect} = O(k_i). \quad (2.73)$$

Підсумкова оцінка

Сумарна обчислювальна складність однієї ітерації комплексного методу для вузла i визначається як:

$$T_{total} \approx T_{trust} + T_{elect} = O(k_i) + O(k_i) = O(k_i). \quad (2.74)$$

Таблиця 2.1

Підсумкова оцінка складності алгоритму

Тип ресурсу	Залежність від N	Пояснення
CPU (одного дрона)	Лінійна $O(N)$	Обробка $N - 1$ вхідних повідомлень.
Radio (завантаження ефіру)	Лінійна $O(N)$	Кожен дрон транслює лише 1 пакет за раунд.
Пам'ять (RAM)	Лінійна $O(N)$	Зберігання станів LSTM (h_t, C_t) для $N - 1$ сусідів.

Таким чином, алгоритм має лінійну складність від кількості сусідів. У найгіршому теоретичному випадку (повний граф, $k_i \approx N$) складність становить $O(N)$. Це є суттєвою перевагою над класичними протоколами візантійського консенсусу (наприклад, *PBFT*), які, як зазначають [10] у роботі над легковаговим консенсусом *LAP-BFT*, вимагають інтенсивного обміну повідомленнями, що призводить до квадратичної складності $O(N^2)$ та перевантаження каналу. Запропонований метод забезпечує масштабованість, необхідну для функціонування роїв середнього та великого розміру без деградації

продуктивності бортових систем.

2.4.2. Аналіз комунікаційного навантаження

У бездротових мережах *FANET*, що використовують спільне середовище передачі, критичним параметром ефективності є не лише обсяг переданих даних, а й частота доступу до каналу. Кожна спроба передачі пакету вимагає виконання процедури змагання за канал або очікування часового слоту. Тому порівняння методів має базуватися на оцінці кількості фізичних передач та ефективності використання пакетів.

Для забезпечення надійності в умовах візантійських відмов розроблений метод реалізує трифазний протокол узгодження в кожному раунді керування:

1. Фаза 1 (*Proposal*): Розсилка пропозицій (команд) та векторів стану.
2. Фаза 2 (*Decision*): Розсилка локально прийнятих рішень (голосування) для перевірки кворуму.
3. Фаза 3 (*Commit*): Фіксація спільного рішення та перехід до виконання.

Службова інформація додається до стандартних періодичних повідомлень телеметрії, що дозволяє уникнути генерації «порожніх» пакетів керування.

Розглянемо структуру корисного навантаження пакетів на різних фазах:

1. Пакет фази 1 (*Proposal*):

- L_{base} (Телеметрія): Ідентифікатор, координати, вектор швидкості, орієнтація, заряд (≈ 52 байти).
- L_{obs} (Спостереження): Список ідентифікованих особливостей середовища. При передачі ID та координат $K = 5$ об'єктів (≈ 80 байт).
- L_{cmd} (Команда): Вектор пропонованої дії (≈ 12 байт).
- Сума: ≈ 144 байти.

2. Пакет фази 2 (*Decision/Vote*):

- L_{vote} (Голос): ID підтриманої пропозиції або вектор локального рішення, що свідчить про вибір вузла серед альтернатив (≈ 4 байти).
- Сума: ≈ 4 байти (плюс заголовок).

3. Пакет фази 3 (*Commit*):

- L_{commit} (Фіксація): Підтвердження переходу до виконання дії (≈ 1 байт).
- Сума: ≈ 1 байт (плюс заголовок).

Сумарний обсяг даних, що генерується одним БПЛА за раунд, становить приблизно 149 байт (без урахування заголовків протоколів нижнього рівня). Такий обсяг є незначним для пропускну здатності сучасних стандартів зв'язку (*IEEE 802.11*, *LoRa*), що гарантує відсутність фрагментації пакетів та мінімальні затримки на передачу.

З точки зору навантаження на мережу, метод використовує переваги фізичного широкомовлення (*Broadcast*), характерного для бездротового середовища. Кожен пакет, відправлений вузлом, приймається всіма його сусідами одночасно. Це дозволяє уникнути необхідності встановлення окремих з'єднань «точка-точка», характерних для дротових мереж. Відповідно, для мережі з N вузлів сумарна кількість фізичних передач за один раунд становить $3 \times N$.

Таким чином, комунікаційна складність методу є лінійною $O(N)$, що відповідає показникам ефективності сучасних легковагових протоколів консенсусу для мобільних мереж і забезпечує необхідну масштабованість для керування роєм.

Висновки до розділу 2

У межах другого розділу розроблено та теоретично обґрунтовано комплексний підхід до адаптивного керування роєм безпілотних апаратів, який дозволяє знайти ефективний компроміс між вимогою мінімізації затримок та необхідністю захисту від внутрішніх загроз. Сформульовано математичну модель системи, що формалізує простір станів агентів та модель загроз типу «візантійська фальсифікація», де зловмисник маніпулює семантичним змістом команд (ін'єкція хибних цілей, саботаж) або параметрами протоколу, залишаючись технічно справним вузлом мережі.

Першою складовою розробленого комплексу є метод узгодження рішень на основі динамічної оцінки довіри. Його ключова наукова новизна полягає у відмові від статичних порогових значень (констант) на користь адаптивного параметра – динамічного рівня довіри (Ψ_{req}). Використання компактної рекурентної мережі (LSTM) дозволяє аналізувати часові ряди поведінкових метрик та автоматично регулювати цей рівень в режимі реального часу. На відміну від існуючих рішень, які часто обмежуються бінарною класифікацією загроз («атака»/«норма») або статичними евристичними правилами (як зазначено у роботі [7]), запропонований метод впроваджує механізм неперервного регулювання «вимогливості» системи. Це забезпечує здатність рою автоматично переходити від режиму високої ефективності до режиму нульової довіри при виявленні аномалій, адаптуючись до нових стратегій атак без оновлення програмного забезпечення.

Другою складовою є метод вибору лідера, побудований на модифікації алгоритму зваженої кластеризації. Новизна методу полягає в інтеграції виходу нейромережі як керуючого параметра для мультиплікативного фільтра («гейта») у функції корисності. На відміну від відомих мультиплікативних моделей ранжування (які використовуються, наприклад, у нечітких системах або імовірнісних протоколах [8]), де вагові коефіцієнти є фіксованими або залежать від інертної статистики, запропонований підхід використовує адаптивну оцінку контексту. Це дозволяє системі змінювати суворість відбору кандидатів динамічно: у безпечному середовищі пріоритет надається фізичній ефективності,

а в умовах загрози – бездоганній репутації. Такий підхід нівелює вплив скомпрометованих вузлів значно ефективніше, ніж традиційні адитивні або статичні мультиплікативні методи.

Виконаний теоретичний аналіз підтвердив, що попри застосування компонентів штучного інтелекту, обчислювальна та комунікаційна складність запропонованих алгоритмів залишається лінійною $O(N)$ відносно кількості сусідів. Це досягається завдяки використанню оптимізованих архітектур нейромереж для граничних обчислень, ефективність яких обґрунтована у роботі [17]. Отримані характеристики відповідають вимогам до сучасних легковагових протоколів керування для БПЛА, розглянутих у роботі [10], забезпечуючи необхідну масштабованість та ефективність використання радіоканалу при значно вищому рівні адаптивності до загроз.

Таким чином, створене математичне забезпечення створює необхідний базис для програмної реалізації та експериментальної перевірки системи керування, що буде виконано у наступному розділі.

РОЗДІЛ 3

ЗАСОБИ УЗГОДЖЕННЯ РІШЕНЬ ІЗ ДИНАМІЧНОЮ ДОВІРОЮ ТА ВИБОРУ ЛІДЕРА У РОЗПОДІЛЕНИХ СИСТЕМАХ

3.1. Вибір технологічного стеку та середовища моделювання

Метою розділу є експериментальна перевірка та програмна верифікація двох взаємодоповнюючих компонентів системи керування роєм: засобу вибору лідера на основі інтеграції динамічної довіри та засобу узгодження рішень з динамічною оцінкою довіри та стійким консенсусом із застосуванням нейромереж. Зважаючи на стохастичну природу фізичної симуляції (вплив асинхронності потоків, недетермінованість мережових затримок у середовищі *Gazebo*), пряме порівняння ефективності різних алгоритмів у серії послідовних запусків може містити похибку, викликану зміною початкових умов. Для забезпечення наукової чистоти експерименту («за інших рівних умов») реалізація виконана за модульним принципом із використанням підсистеми логування даних. Це дозволяє зафіксувати еталонні траси польоту та профілі атак, на яких згодом проводиться апостеріорна валідація роботи нейромережевого модуля та розрахунок метрик ефективності вибору лідера. Такий підхід гарантує, що виявлені покращення показників є результатом роботи запропонованих методів, а не випадкових флуктуацій симуляційного середовища.

Симуляційне середовище реалізовано на зв'язці *ROS 2* та *Gazebo* з автоматизованим сценарним запуском через *tmuxinator*; доменно-специфічні функції винесено у власний плагін *Gazebo* та вузол *drone_controller*. Зв'язка *ROS 2* + *Gazebo* є загальноприйнятим стандартом у робототехнічній спільноті для моделювання та відлагодження алгоритмів, що підтверджується офіційною документацією та навчальними матеріалами інтеграції *ROS 2* і *Gazebo* [4], [89]. Спеціалізований фреймворк *Aerostack2* також розглядався, однак з огляду на ресурсні обмеження перевагу надано полегшеній конфігурації *ROS 2* + *Gazebo* з цільовим плагіном. У науково-інженерній практиці рої БПЛА регулярно симулюються саме у середовищі *ROS/ROS 2* + *Gazebo* – як у відкритих

фреймворках, так і в дослідницьких роботах. Керування багатопроцесним запуском забезпечує *tmuxinator*, що спрощує відтворюваність прогонів.

Щоб забезпечити відтворюваність і керованість експериментів, використано мінімальну, але розширювану конфігурацію: *ROS 2* як транспорт подій і типізованих повідомлень, *Gazebo* як кінематико-фізичний рушій сцени з власним плагіном для доменної логіки, спеціалізований вузол *drone_controller* для раундового обміну («пропозиції»/»рішення»/»звіт сенсора») та *tmuxinator* як диспетчер багатопроцесних сесій (сценарний запуск і стабільна структура логів). Такий розподіл дає чіткі межі відповідальності: *ROS2* відповідає за теми, профілі якості обслуговування (*QoS*) та синхронізацію старту; *Gazebo* – за динаміку сцени і доступ до датчиків через плагін; *drone_controller* – за алгоритмічні цикли; *tmuxinator* – за «одноклавішний» підйом усіх процесів і їх коректне завершення [4], [89].

Організація запуску стандартизована: кожен сценарій описується *YAML*-файлом *tmuxinator* із окремими вікнами для симулятора, службових вузлів *ROS 2* та множини екземплярів *drone_controller*. Синхронізація початку виконується через службове повідомлення зі зсунутим часом старту; далі обмін відбувається періодично з коротким фіксованим інтервалом. Для кожного прогону фіксуються версії пакунків, зерно випадковості, профілі *QoS* і шляхи вивантаження артефактів (журнали подій, таблиці метрик, за потреби – збережені стани навчальної моделі). Це забезпечує повторюваність і дає змогу автоматично збирати дані для аналізу без зміни коду експериментів.

Структура процесів і життєвий цикл запуску:

1. Ініціалізується сесія *tmuxinator* із параметрами: кількість апаратів, радіус взаємодії R , константа затримки β на хоп, обмеження швидкості, насіння генератора випадкових чисел, ідентифікатор сценарію та шляхи до конфігурацій.
2. Піднімається *Gazebo* зі світом і підключеним плагіном; перевіряється готовність тем *ROS 2* до обміну [4], [89].
3. Запускаються вузли *ROS 2*, зокрема *drone_controller* для кожного апарата

(або групи): публікація цільових точок руху, збір телеметрії; у сценаріях методу динамічної оцінки довіри та стійкого консенсусу із застосуванням нейромереж – вузли оцінювання узгодженості, оновлення довіри та зведення рішень.

4. Стартує експериментальна сесія з фіксованою тривалістю або з умовою завершення (проходження контрольних точок чи досягнення цільових подій).
5. Виконується згортання: зупинка вузлів, збереження агрегованих метрик і архівація журналів у папку прогону з уніфікованою схемою іменування.

Розподіл відповідальностей: плагін *Gazebo* та вузол *drone_controller*.

– Плагін *Gazebo* реалізує низькорівневе переміщення: отримує від *drone_controller* поточну цільову точку та рухає дрон у напрямку цієї точки нормалізованим вектором, помноженим на швидкість; додатково забезпечується підтримання висоти з компенсацією прискорення 9,8. Плагін не формує траєкторій і не керує формаціями; він виконує команди переміщення та публікує мінімальну телеметрію в теми *ROS 2*.

– Вузол *drone_controller* відповідає за логіку траєкторій і формацій: формування маршруту на основі списків контрольних точок, профілювання швидкості на поворотах траєкторії центроїди для компенсації різної довжини дуг у крайніх апаратів, підтримання формації та вибір наступної цільової точки; у сценаріях методу динамічної оцінки довіри – оброблення подій «виявлення» від детектора наближення до заданих координат і взаємодія з вузлами оцінювання узгодженості/довіри та зведення рішень. Передавання цільових точок плагіну і приймання телеметрії виконуються через узгоджені теми *ROS 2*.

Комбінація *ROS 2* – *Gazebo* – *tmuxinator* забезпечує баланс між контрольованістю експериментів, відтворюваністю та обчислювальною помірністю. Розподіл ролей між плагіном *Gazebo* (нижній рівень руху) і вузлом *drone_controller* (траєкторії, формація, події) узгоджується з типовими рекомендаціями інтеграції *ROS 2* та *Gazebo* і підтверджує доцільність обраного підходу для задач цього розділу.

ROS 2 – це децентралізована робототехнічна система, в якій складні проекти розбиваються на безліч модульних вузлів. Кожен вузол відповідає за окрему функціональність, а обмін даними між вузлами здійснюється через топіки – іменовані канали для передачі повідомлень у моделі публікації/підписки. *ROS2* використовує стандарт *DDS (Data Distribution Service)* – розподілене проміжне програмне забезпечення, яке забезпечує децентралізоване виявлення вузлів і обмін повідомленнями. *DDS* надає топік-орієнтовану модель *publish/subscribe* (рис. 3.1) та підтримує налаштовувані політики *Quality of Service (QoS)*. Політики *QoS* дозволяють тонко регулювати характеристики комунікації (доступність даних, надійність доставки, обсяг історії повідомлень, тривалість зберігання тощо), підлаштовуючи систему під вимоги мережі і реального часу. Наприклад, з правильним профілем *QoS ROS 2* може працювати як із гарантованою доставкою повідомлень на рівні *TCP*, так і в режимі «*best-effort*» (аналог *UDP*) для мінімізації затримок – і з багатьма проміжними варіантами. Така гнучкість стала можливою завдяки *DDS* і є суттєвою перевагою при роботі в ненадійних бездротових мережах або системах з жорсткими часовими обмеженнями, де відповідний профіль *QoS* допомагає забезпечити необхідну якість обслуговування передачі даних

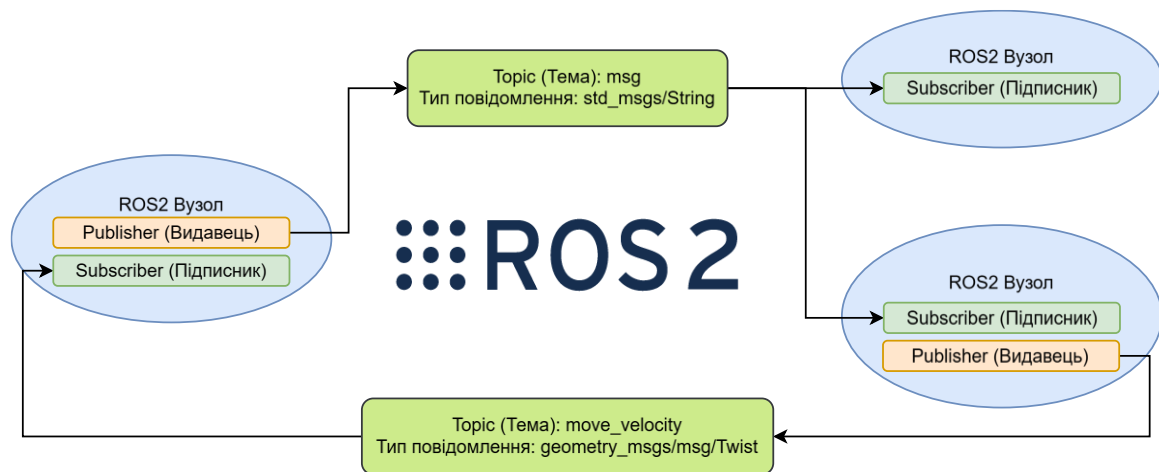


Рис. 3.1 Схеми роботи розподілених вузлів у *ROS2*

Для візуалізації та симуляції *ROS 2* взаємодіє із *Gazebo* – вільнодоступним симулятором робототехнічних систем, який надає тривимірне середовище з

фізичним рушієм для моделювання руху роботів, взаємодії з об'єктами сцени та роботи сенсорів. Основою опису сцени є файл світу (*drone_swarm.world*) у форматі *SDF*, де задаються гравітація, часові параметри симуляції, склад сцени (моделі), а також підключаються плагіни для розширення поведінки об'єктів [4]. Моделі можуть містити ланки та з'єднання, сенсори (наприклад, «лідар», камера, ІНС) і виконавчі механізми, що дозволяє відтворювати як кінематику, так і базову динаміку системи [4].

Архітектурно *Gazebo* підтримує кілька рівнів розширення: плагіни світу, моделей і системні плагіни. Плагіни під'єднуються до життєвого циклу симуляції та можуть зчитувати стан об'єктів, генерувати керувальні впливи, публікувати телеметрію або іншим чином взаємодіяти з середовищем симулятора [89]. У типовій інтеграції з *ROS 2* застосовується міст *ros_gz*, який відображає внутрішні топіки *Gazebo* на топіки *ROS 2* і навпаки, що дає змогу використовувати вузли *ROS 2* для керування моделями та читання сенсорних даних без прямої прив'язки до внутрішнього транспорту *Gazebo* [89].

У цій роботі файл конфігурації симуляції *Gazebo* називається *drone_swarm.world* і мінімальний за складом – містить пласку поверхню (*ground plane*) та декілька дронів. Кожен дрон описаний у *SDF* як окремий рухомий об'єкт зі своїми фізичними параметрами (масою, розмірами, моделлю пропелерів тощо) та сенсорами, що дозволяє *Gazebo* прораховувати динаміку польоту і відтворювати дані датчиків віртуального дрона. Низькорівневий рух реалізовано в окремому плагіні *Gazebo drone_driver*. Для взаємодії *ROS 2* з симулятором *Gazebo* використовується спеціальний програмний міст, наприклад пакет *ros_gz_bridge*. Він створює двосторонній канал обміну повідомленнями між екосистемою *ROS 2* та внутрішньою транспортною системою *Gazebo*. Симулятор *Gazebo* оперує власним проміжним програмним забезпеченням (*Gazebo Transport*), що публікує дані про стан симуляції на топіках *Gazebo* (за структурою подібних до *ROS*-топиків) і надає сервіси керування. Міст *ros_gz_bridge* транслює ці повідомлення в повідомлення *ROS 2* типів, сумісних з форматами *Gazebo*, і навпаки, виконуючи необхідне перетворення форматів даних. Таким чином досягається прозора

інтеграція: *ROS 2*-вузли можуть підписуватися на “симуляторні” дані (наприклад, отримувати від *Gazebo* дані віддаленості або координати дронів як стандартні повідомлення *ROS*), а також публікувати команди керування, які через міст потрапляють у *Gazebo* для впливу на моделі дронів (наприклад, рух або зміна положення). Варто зауважити, що міст підтримує обмежений набір типів повідомлень, узгоджених між *ROS 2* та *Gazebo*, і його можна налаштувати на передачу даних у потрібному напрямку (від *Gazebo* до *ROS*, від *ROS* до *Gazebo* або двосторонньо) залежно від вимог симуляції. Це рішення поєднує переваги обох систем: реалістичну фізичну симуляцію *Gazebo* та розподілену архітектуру *ROS 2* з її засобами для побудови рою роботів, забезпечуючи єдину узгоджену платформу для моделювання та керування роєм дронів.

Формування траєкторій, профілювання швидкості на поворотах і підтримання формації виконується у вузлі *drone_controller* (*ROS 2*), який через *ros_gz* передає цільові точки плагіну та отримує телеметрію назад [4], [89] (рис. 3.2). Такий поділ забезпечує відтворюваність експериментів і відповідає рекомендованим сценаріям інтеграції *Gazebo* ↔ *ROS 2* [4], [89].

Запуски симуляції організовано через *tmuxinator*, що слугує сценарним диспетчером багатопроцесних сесій. Такий підхід значно полегшує організацію запуску симуляції, оскільки на кожен дрон треба запустити декілька програм. У конфігурації задається сесія з окремими вікнами для *Gazebo*, мостів *ROS 2* ↔ *Gazebo* та *ROS 2* вузлів. Повний *YAML*-файл конфігурації зберігається в репозиторії; у тексті наведено структурований опис без вставки коду.

У профілі використовуються змінні налаштувань, зокрема *drone_namespaces* – список ідентифікаторів моделей дронів, поданий через кому. На основі цього списку автоматично генеруються вікна/панелі для кожного апарата, що забезпечує масштабованість запусків без ручного редагування файлів. Такий підхід забезпечує наступну структуру запуску симуляції для перевірки методу вибору лідера – у терміналі створюється вікно *gazebo* і *nav_<drone>* (для кожного дрона). Вікно *gazebo* відповідає за запуск симуляції із відповідним файлом конфігурації та за її налаштування, а у вікнах *nav_<drone>* створюються мости для підтримки

комунікації між *ROS2* та *Gazebo* (рис. 3.3).

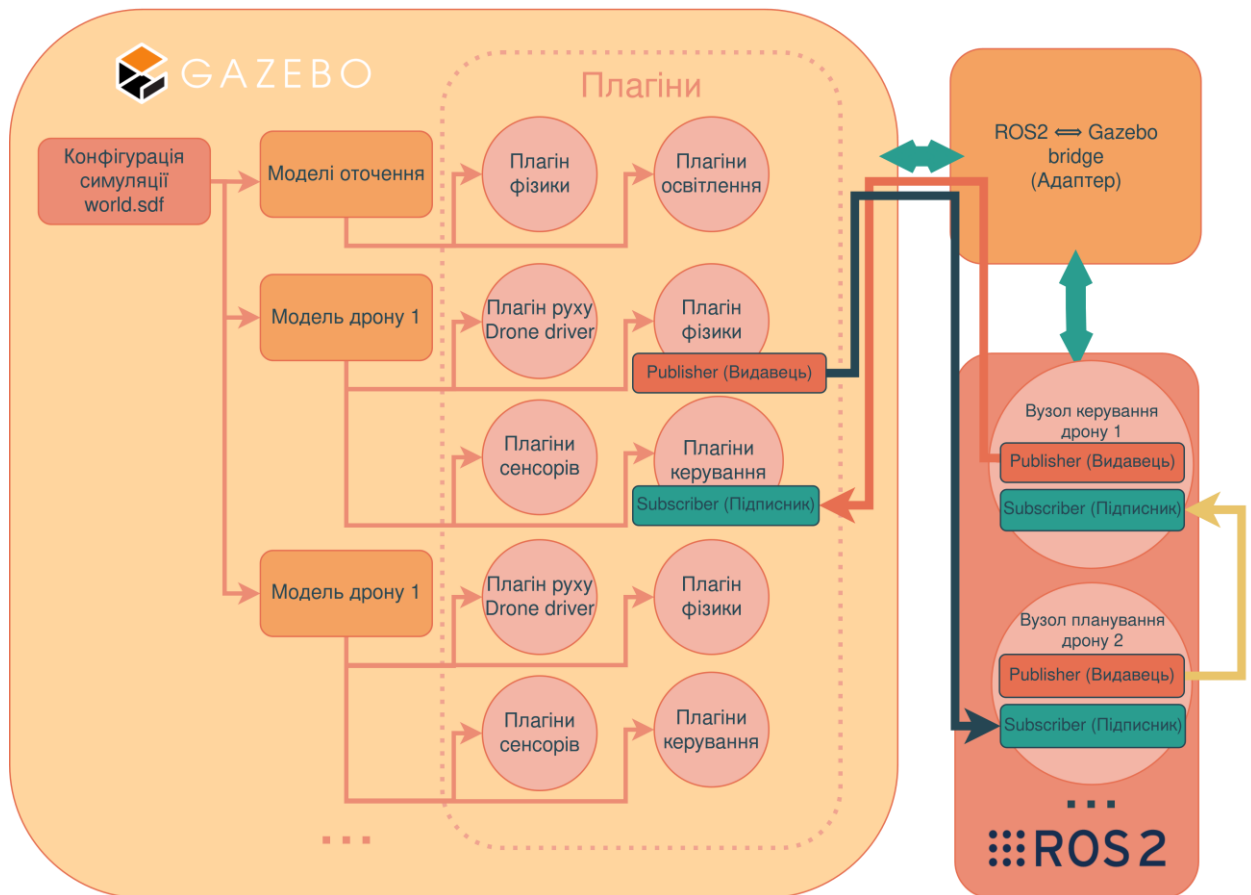


Рис. 3.2 Схема взаємодії симулятора *Gazebo* із *ROS2*

Для методу динамічної оцінки довіри та стійкого консенсусу із застосуванням нейромереж конфігурація розширена додатковими вікнами/панелями під вузли міждронового обміну, оцінювання узгодженості, динамічної довіри та зведення рішень (мости для відповідних топіків, вузли *heartbeat*/пропозицій/рішень). Це дозволяє зберегти єдину схему запусків і відтворюваність серій прогонів. На цьому тлі доцільно зафіксувати вибір мов реалізації модулів і межі відповідальності між ними, оскільки саме ці рішення визначають стабільність часових характеристик та швидкодію симуляції у подальших експериментах.

У проєкті застосовано дві мови: *C++* для плагіна *Gazebo* та *Python* для вузлів *ROS 2*. Такий поділ впливає з природи інструментів. Екосистема *ROS 2* офіційно підтримує клієнтські бібліотеки як для *C++*, так і для *Python*, що дозволяє обирати

мову відповідно до ролі компонента в системі та вимог до часу виконання [4]. Плагін *Gazebo* працює в тілі циклу оновлення світу, взаємодіє з фізичним рушієм і повинен реагувати з фіксованою частотою без додаткових накладних витрат; офіційні інтерфейси плагінів історично надаються насамперед для C++, а довідкові матеріали орієнтовані саме на цей стек. Тому низькорівневе переміщення дрона до поточної цілі (рух уздовж нормованого вектора, помноженого на швидкість, з підтриманням висоти) реалізовано на C++, що забезпечує передбачуваність часу виконання та повну сумісність із життєвим циклом *Gazebo* [89], а також відповідає рекомендованим практикам розроблення плагінів і підключення їх до сцени світу.

```

ros2 run ros_gz_bridge parameter_bridge /model/drone1/cmd_move@geometry_msgs/msg/Quaternion@ignition.msgs.Quaternion
mmelench@mmelench-PC:~/Documents/phd/dissertation/simple_swarm/swarm_world$ ros2 run ros_gz_bridge parameter_bridge /mo
del/drone1/cmd_move@geometry_msgs/msg/Quaternion@ignition.msgs.Quaternion
[INFO] [1757089569.179436245] [ros_gz_bridge]: Creating GZ->ROS Bridge: [/model/drone1/cmd_move (ignition.msgs.Quaterni
on) -> /model/drone1/cmd_move (geometry_msgs/msg/Quaternion)] (Lazy 0)
[INFO] [1757089569.189531430] [ros_gz_bridge]: Creating ROS->GZ Bridge: [/model/drone1/cmd_move (geometry_msgs/msg/Quat
ernion) -> /model/drone1/cmd_move (ignition.msgs.Quaternion)] (Lazy 0)
|

ros2 run ros_gz_bridge parameter_bridge /model/drone1/pos@geometry_msgs/msg/Pose@ignition.msgs.Pose
mmelench@mmelench-PC:~/Documents/phd/dissertation/simple_swarm/swarm_world$ ros2 run ros_gz_bridge parameter_bridge /mo
del/drone1/pos@geometry_msgs/msg/Pose@ignition.msgs.Pose
[INFO] [1757089569.262703152] [ros_gz_bridge]: Creating GZ->ROS Bridge: [/model/drone1/pos (ignition.msgs.Pose) -> /mod
el/drone1/pos (geometry_msgs/msg/Pose)] (Lazy 0)
[INFO] [1757089569.267367727] [ros_gz_bridge]: Creating ROS->GZ Bridge: [/model/drone1/pos (geometry_msgs/msg/Pose) ->
/model/drone1/pos (ignition.msgs.Pose)] (Lazy 0)

[swarm dro0:gazebo- 1:nav drone1* 2:nav drone2 3:nav drone3 4:nav drone4 5:nav drone5 >"mmelench-PC" 19:26 05-Sep-25

```

Рис. 3.3 Запуск вузлів симулятора для дослідження методу вибору лідера

Вузли в графі *ROS 2* виконують логіку вищого рівня – формування траєкторій у *drone_controller*, підтримання формації, диспетчеризацію подій і журналювання. Для цих завдань обрано *Python*, оскільки він пришвидшує моделювання та зміну сценаріїв, полегшує роботу з параметрами, дає змогу швидко додавати/змінювати топіки і скорочує час між ітераціями експериментів без торкання критичних по

продуктивності ділянок [4]. Такий вибір узгоджено з мостом між *ROS 2* та *Gazebo*: пакет *ros_gz_bridge* підтримує відображення необхідних типів повідомлень і дозволяє прозоро обмінюватися командами руху та телеметрією між вузлами *Python* і плагіном *C++* без ручної роботи з внутрішнім транспортом *Gazebo* [89], тоді як низькорівневі операції залишаються у швидкій *C++*-частині. Сценарна організація запусків через *tmuxinator* з фіксацією версій і параметрів додатково підсилює відтворюваність і не залежить від обраної мови кожного модуля.

Загалом, обраний поділ ролей мінімізує ризик зависань у фізичному циклі симуляції, залишаючи часочутливі обчислення у *C++*, і водночас зберігає високу гнучкість експериментів завдяки використанню *Python* у керувальних вузлах *ROS 2*. Така архітектура відповідає практиці спільноти та забезпечує потрібний баланс між стабільністю виконання й швидкістю наукових ітерацій [4], [89].

3.2 Структура засобу динамічної оцінки довіри та стійкого консенсусу

Засіб реалізації методу динамічної оцінки довіри та стійкого консенсусу побудовано на основі запропонованої інфраструктури. Додано циклічний обмін пропозиціями, локальне зважування на основі довіри з карантинном та зведення рішень за правилом супербільшості дві третини.

Робота відбувається в коротких синхронізованих циклах. Кожен апарат формує пропозицію зі спостережених подій у межах чутливості сенсора і розсилає її всім учасникам (рис.3.4). Після отримання повного набору пропозицій локальне ядро прийняття рішень обчислює власне рішення з урахуванням динамічної довіри, що оновлюється експоненційним ковзним середнім за ознаками узгодженості з більшістю попередніх циклів, стабільності переліку подій, регулярності участі та відсутності надмірно великих повідомлень. Для джерел із стало низькими показниками застосовується карантин із гістерезисом, який тимчасово виключає їхній внесок і повертає лише після послідовного покращення.

Для експериментальної верифікації розробленого засобу та перевірки стійкості механізмів довіри було обрано модельний сценарій патрулювання реперного маршруту групою БПЛА. У цьому сценарії рій дронів виконує рух у

заданій геометричній формації (наприклад, «клин», «лінія» або «колона») уздовж траєкторії, насиченої віртуальними контрольними точками – мітками (*landmarks*).

Програмний модуль сенсора (*MarkSensor*) налаштований таким чином, щоб імітувати роботу реальної системи комп'ютерного зору або радіопеленгації: виявлення мітки відбувається лише за умови входження дрона в зону впевненого розпізнавання, радіус якої є обмеженим і співмірним з дистанцією між сусідніми апаратами в строю. Через таку конфігурацію в процесі польоту виникають ситуації часткової спостережуваності: дрони, що знаходяться на фланзі формації, можуть зафіксувати об'єкт, який залишається невидимим для центральних апаратів, і навпаки (рис. 3.5). Це створює природну невизначеність («туман війни»), на тлі якої розгортається взаємодія агентів.

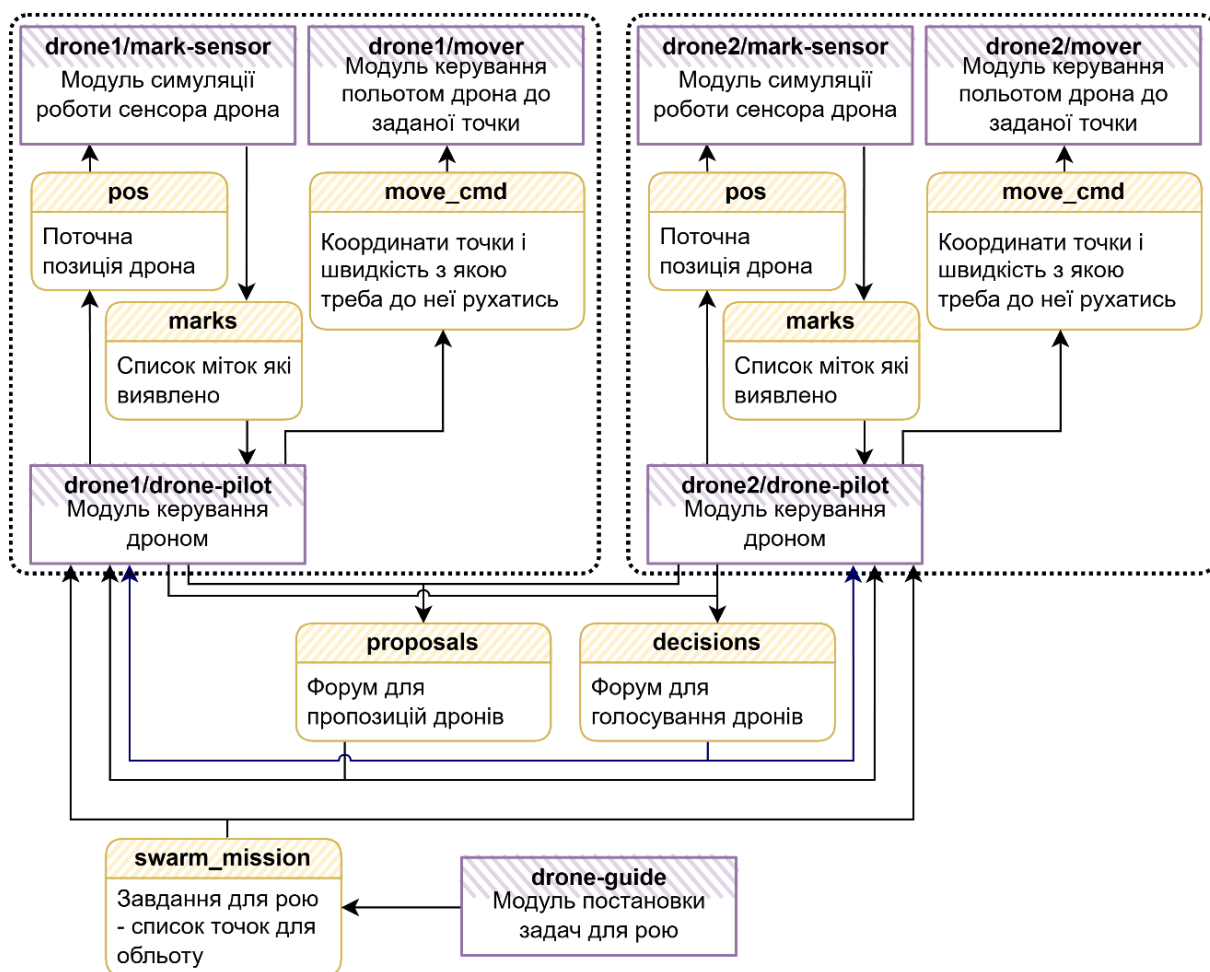


Рис. 3.4 Діаграма раундової взаємодії дронів рою

Модельовання візантійської поведінки реалізовано шляхом програмної ін'єкції

спотворень у модуль формування пропозицій (*_handle_proposal_round*) для визначеної підмножини дронів-зловмисників (рис. 3.6).

Реалізовано два базові типи атак на дані:

- Ігнорування (*False Negative*): Зловмисний вузол, фактично знаходячись у зоні видимості реальної мітки, навмисно не включає її до свого звіту, імітуючи «сліпоту» сенсора або саботаж розвідки.
- Галюцинація (*False Positive*): Зловмисник повідомляє про виявлення неіснуючої мітки у своїй зоні відповідальності, намагаючись спровокувати рій на хибний маневр або відхилення від маршруту.

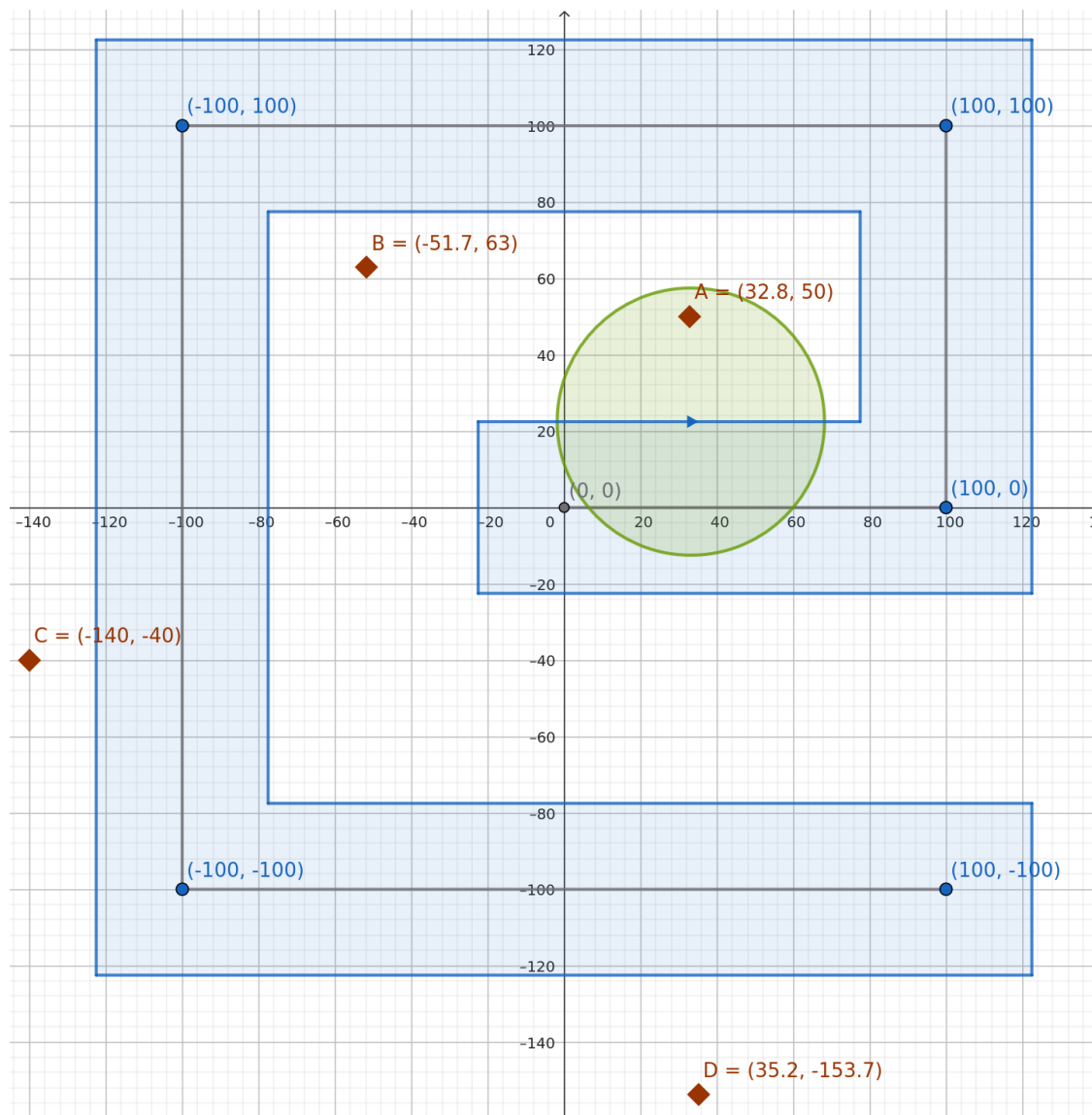


Рис. 3.5 Приклад реперного маршруту для рою дронів у якому мітку A бачать тільки дрони лівого флангу

Ключова задача розробленого засобу в цих умовах полягає у коректному розв'язанні конфліктів свідчень (коли одні сусіди повідомляють «бачу», а інші – «не бачу»). Алгоритм повинен, спираючись на історію поведінки та перехресні перевірки зон видимості, відрізнити ситуативну відсутність контакту (через дистанцію) від цілеспрямованої дезінформації та прийняти єдине правильне колективне рішення: виконати команду, пов'язану з міткою, або продовжити рух за маршрутом.

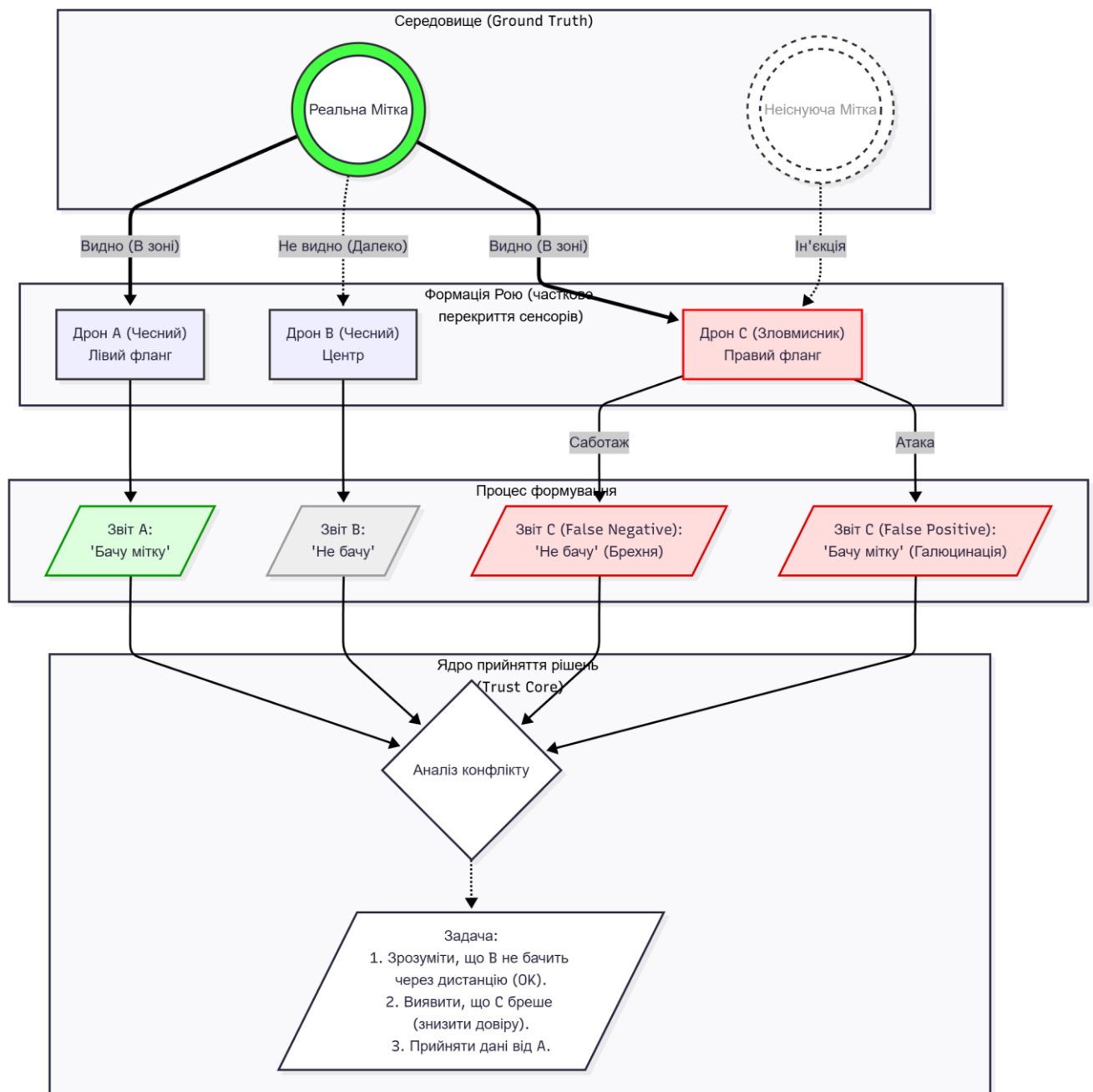


Рис. 3.6 Схеми конфлікту свідчень у рої БПЛА

Зведення до спільного результату ґрунтується на простому правилі: якщо не менше ніж дві третини апаратів оголосили ідентичне рішення «летіти до події з координатами μ », уся група приймає це рішення; інакше всі продовжують рух базовим маршрутом. На етапі кворуму не виділяються спеціальні «якірні» джерела, оскільки під час роботи невідомо, які апарати є достовірними. Довіра впливає лише на те, як окремих апарат формує власне попереднє рішення перед голосуванням (які події він підтримує), але не змінює сам критерій двох третин у кворумі. Конфлікти між кількома подіями з однаковою підтримкою розв'язуються детерміновано за сталою лексикографічною перевагою координат, що гарантує однаковий результат на всіх вузлах.

Поріг θ , який використовується на локальному етапі (перед кворумом) для відбору «достатньо підкріплених» подій, адаптується у двох конфігураціях, що відрізняються лише способом обчислення, але мають спільний набір ознак стану:

- Детермінована конфігурація: θ обчислюється за правилом, похідним від довірчих оцінок і частки підтримки. Базове значення θ_0 коригується обмеженою функцією від тих самих ознак, що входять у вектор ознак ситуації (кількість потенційних спостерігачів, кількість звітуючих, запас до базового порога за сумою довір, співвідношення «за/проти», тощо). Корекція монотонно зростає зі зростанням підтвердження і не зменшує поріг за відсутності достатніх підстав (захисні умови), при цьому θ завжди належить до заздалегідь визначеного безпечного інтервалу.
- Нейромережева конфігурація: ті самі ознаки подаються на вхід малої рекурентної моделі типу *LSTM* (довга-короткочасна пам'ять). Модель обчислює коефіцієнт довіри на основі таких же параметрів що і у детермінованій конфігурації. Використано центрювання за ковзним середнім прогнозів, щоб уникати дрейфу, та захисні умови, які не дозволяють знижувати θ поза ситуаціями з достатньою підтримкою більшості та повторюваними «втраченими можливостями». Як і в детермінованій конфігурації, θ обмежено однаковими нижньою та верхньою межами.

В обох конфігураціях формується один і той самий вектор ознак ситуації (зокрема кількість потенційних спостерігачів, кількість звітуючих, запас за сумою довір відносно базового порога, відношення «за/проти», міра «свіжості» та стабільності), після чого обчислений θ застосовується для локального відбору кандидатів. Прийняте спільне рішення виконує модуль руху: він вставляє відгалуження до події в поточний маршрут з урахуванням позиції та швидкості, після відвідування повертає апарат на базову траєкторію й реєструє подію у локальному сховищі. Записи про відвідування використовуються для корекції довіри до учасників, які підтримували рішення, а також як навчальна інформація для нейромережевої конфігурації. Логи та метрики збираються у стандартизованому вигляді, що дає змогу зіставляти роботу детермінованої та нейромережевої конфігурацій без зміни решти компонентів засобу.

Щоб застосувати обчислений поріг θ у роботі рою, вузол керування підтримує сталий цикл раундів (рис. 3.7):

1. Спостереження подій
2. Формування пропозиції
3. Локальний відбір кандидата за θ
4. Обмін локальними рішеннями
5. Визначення спільного рішення за правилом кворуму
6. Виконання обраної дії та фіксація результатів у журналах і метриках.

Локальна пам'ять організована за ідентифікатором раунду та включає три колекції:

1. Власні спостереження («раунд $\rightarrow$ перелік подій»)
2. Пропозиції інших учасників («раунд $\rightarrow$ перелік отриманих пропозицій»)
3. Локальні рішення рою («раунд $\rightarrow$ перелік отриманих рішень»).

Дані від датчика у межах одного раунду дедуплікуються, тож кожна подія враховується один раз. Власна пропозиція містить номер раунду, ім'я джерела, поточну позицію та перелік зафіксованих подій і публікується у спільний канал рою.

Після накопичення пропозицій на поточний раунд виконується локальний

відбір «наступного завдання» за порогом θ . Для кожної події розраховується підтримка з урахуванням кількості потенційних спостерігачів, кількості тих, хто реально звітує, суми довір до звітуючих і частки підтримки серед усіх, хто міг спостерігати. Далі формують локальне рішення: або відвідати конкретну подію (координати цілі), або продовжити рух базовим маршрутом. Це рішення публікується у спільний канал рою.

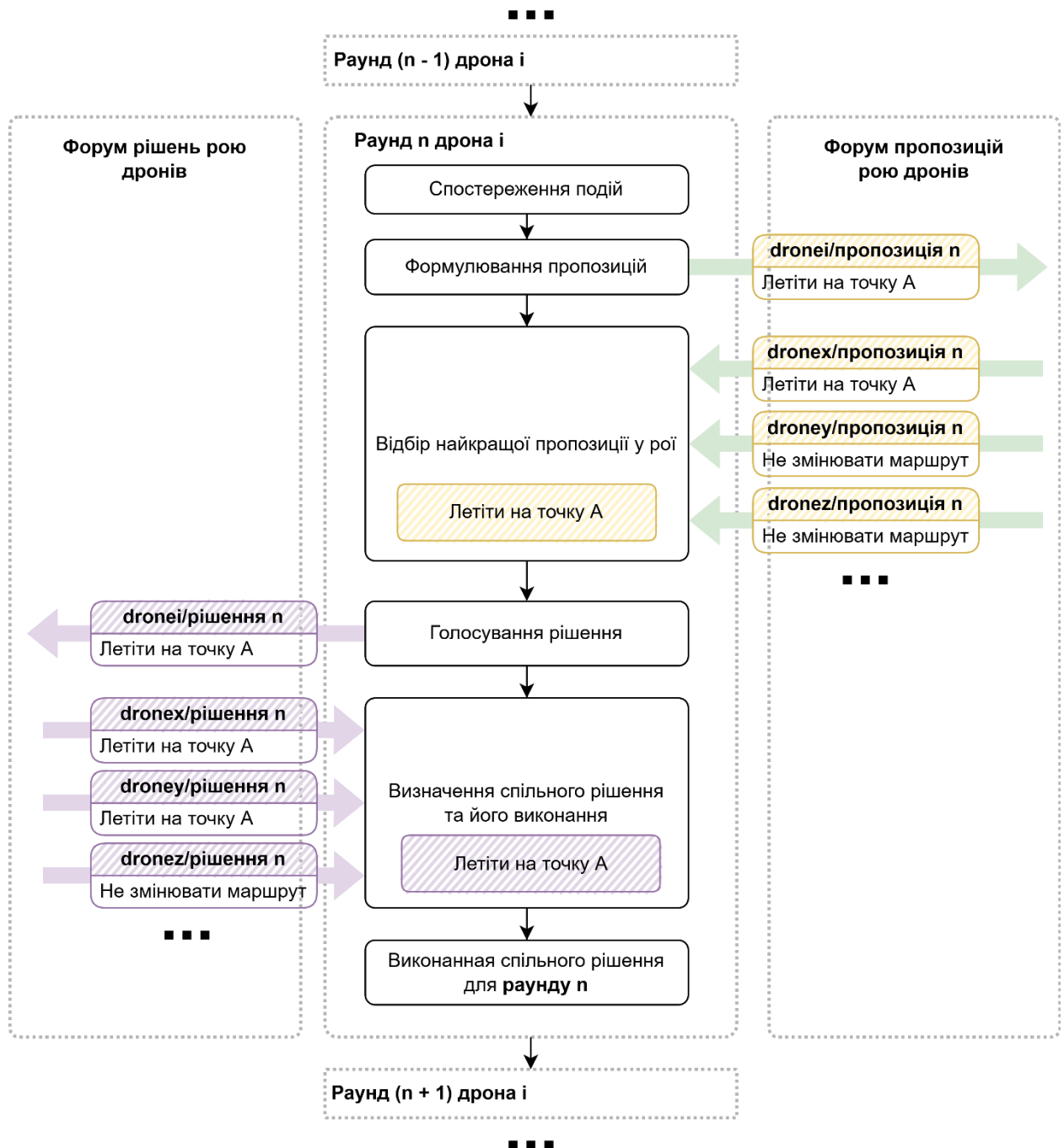


Рис. 3.7 Алгоритм дій дрону у раунді

Після одержання локальних рішень від усіх учасників обчислюється спільне рішення раунду за правилом кворуму більшості: вибір на користь відвідування події визнається, якщо її підтримала не менше ніж дві третини тих, хто міг її спостерігати, а зважена (довірою) сума підтримки перевищує поріг. У разі позитивного результату в траєкторію вставляється відгалуження до цілі із зазначенням сегмента вставки та частки шляху до точки відгалуження; після відвідування апарат повертається на базовий маршрут. Подія відвідування реєструється у локальному сховищі.

Зареєстровані відвідування використовуються для корекції довіри до учасників, які підтримували відповідні рішення, та як навчальна інформація у конфігурації з рекурентною моделлю. Оновлення малої моделі здійснюється без зупинки робочого циклу вузла, із формуванням мікрівибірок і вагових коефіцієнтів, щоб обмежити обчислювальні витрати.

Журнали та метрики збираються у стандартизованому вигляді для подальшого порівняння конфігурацій з детермінованим і нейромережовим обчисленням порога. На рівні рішень за раунди обчислюються кількість раундів і відвідувань, частоти істинних і хибних виборів, точність, повнота, $F1$ -міра, частка раундів із відвідуванням, а також мінімальне, максимальне та середнє значення застосованого θ . На рівні фактичних відвідувань підсумовуються кількість відвідувань, кількість унікальних істинних і хибних відвідувань, повторні відвідування, частка хибних відвідувань і частка покриття істинних подій; додатково наводиться $F1$ -міра покриття. Єдині формати журналів і правил підрахунку забезпечують коректне зіставлення результатів двох конфігурацій без змін інших компонентів засобу.

Обмін службовими повідомленнями розділено за роллю каналів: параметри місії доставляються з буферизацією останнього повідомлення (щоб нові підписники гарантовано отримали стартові дані), тоді як потоки пропозицій і рішень працюють у «летючому» режимі з надійним доставленням. Така конфігурація гарантує надійний старт і водночас мінімізує затримки під час раундів.

Центральним елементом архітектури є клас *DroneController* – вузол *ROS 2*, що забезпечує інтерфейс між мережею та логікою керування. Він підписується на топіки місії (*SwarmMission*), телеметрії (*Pose*), сенсорів (*SensorReport*) та обміну повідомленнями (*Proposal*, *Decision*). Отримані дані передаються до екземпляра класу *DroneStorage*, який виконує функцію локальної бази даних, накопичуючи історію пропозицій та рішень для кожного раунду.

Логіка прийняття рішень інкапсульована у класі *DecisionCore* (рис. 3.8). Він ініціалізується об'єктом *DecisionConfig*, що містить набори параметрів для різних аспектів роботи (довіра, карантин, кворум, нейромережа).

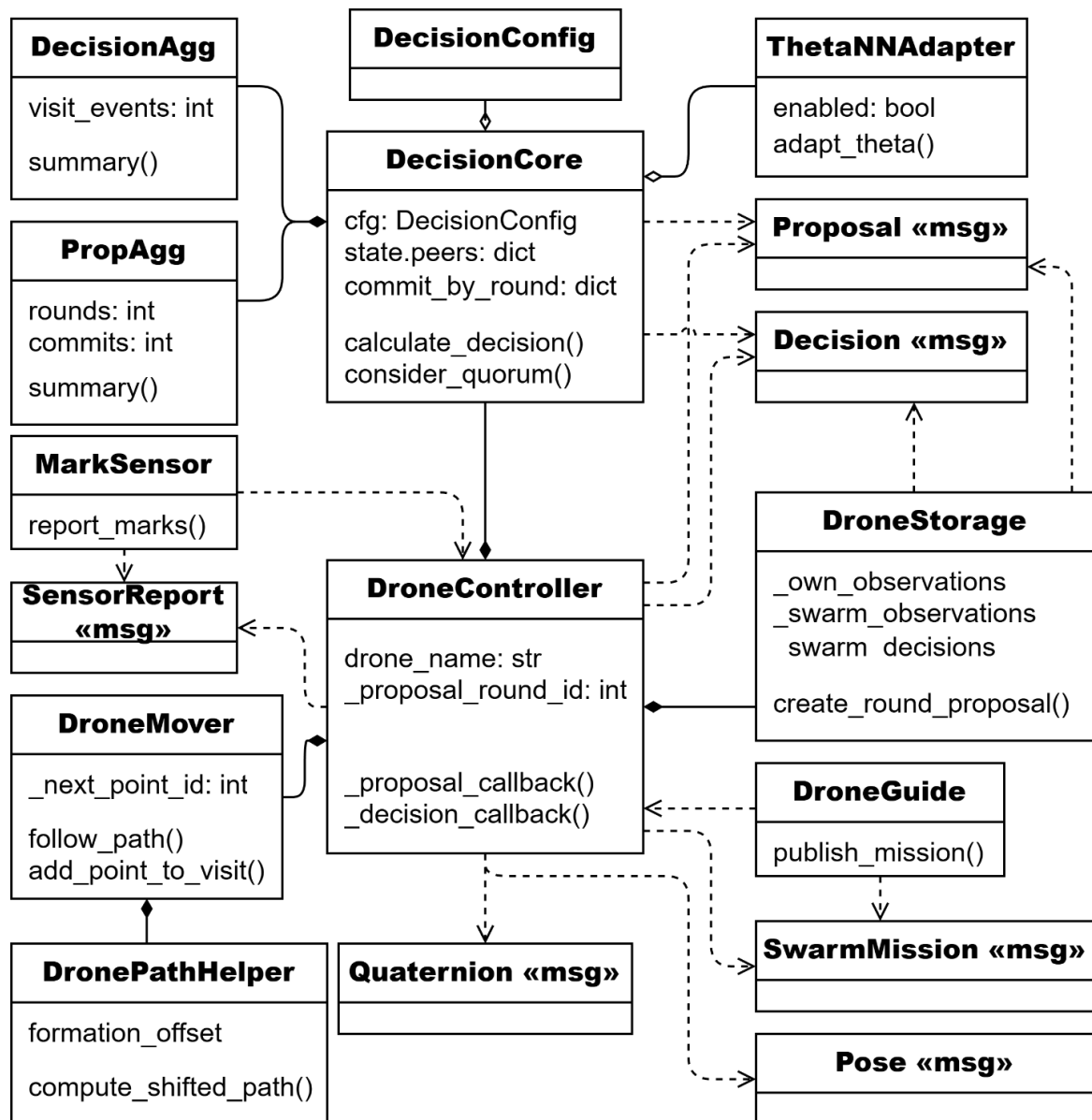


Рис. 3.8 Діаграма класів дрону

DecisionCore взаємодіє з модулем *ThetaNNAdapter*, який відповідає за неймережеву адаптацію порогу довіри. Адаптер використовує модель *TensorFlow* для передбачення коефіцієнта θ на основі вектора ознак, що формується з метрик узгодженості (агрегованих у класах *PropAgg* та *CommitAgg*).

Процес формування пропозиції ініціюється таймером у *DroneController*. Він звертається до *DroneStorage* для створення повідомлення типу *Proposal*, яке містить поточний статус та список виявлених міток. Отримані пропозиції від сусідів обробляються в *DecisionCore*: розраховується локальна довіра, перевіряється статус карантину, і формується локальне рішення (*Decision*).

Взаємодія з фізичним середовищем здійснюється через клас *DroneMover*, який використовує допоміжний клас *DronePathHelper* для розрахунку траєкторій та формацій. Команди руху публікуються у вигляді повідомлень *Quaternion*, а зворотний зв'язок від сенсорів надходить через *MarkSensor* у вигляді повідомлень *SensorReport*. Загальну координацію місії забезпечує зовнішній вузол *DroneGuide*, що публікує параметри завдання.

Основне обчислювальне навантаження покладено на клас *DecisionCore*, який реалізує «чисту» бізнес-логіку методу без прив'язки до транспортного шару *ROS*. Центральним алгоритмом є метод *compute_local_tau*, який виконує перерахунок коефіцієнтів довіри (τ) для кожного сусіда на основі аналізу його пропозицій.

Метод *compute_local_tau* приймає на вхід список отриманих пропозицій (*proposals*) та, опціонально, рішення попереднього раунду (*committed_prev*). Алгоритм розпочинається з агрегації всіх отриманих міток для визначення «колективної картини» раунду. Створюється частотна мапа (*counts*), де для кожної координати мітки підраховується кількість дронів, які про неї повідомили. Це дозволяє визначити рівень підтримки кожної заявленої події.

Далі для кожного дрона-відправника обчислюється вектор метрик узгодженості, що характеризують різні аспекти його поведінки (рис. 3.10):

1. Підтвердження групою (*u*): Ця метрика відображає ступінь консенсусу щодо міток, про які повідомляє конкретний дрон. Вона розраховується як частка міток у пропозиції сусіда, які також бачать інші учасники рою

(кількість підтверджень $\geq s_{min}$). Високе значення u свідчить про те, що дрон бачить ті ж об'єкти, що і більшість, що є сильною ознакою функціональної стабільності. Формально, якщо дрон повідомив про множину міток M , то $u = \frac{|\{m \in M: counts(m) \geq s_{min}\}|}{|M|}$.

2. Унікальність (v): Ця метрика є індикатором потенційних галюцинацій або спроб ін'єкції хибних даних. Вона визначається як частка «соло»-міток, про які повідомляє лише цей дрон ($counts(m) = 1$). Хоча в умовах розрідженого рою можливі ситуації, коли дрон дійсно єдиний бачить мітку, систематично високе значення v (особливо при наявності сусідів поруч) є підозрілим. $v = \frac{|\{m \in M: counts(m) = 1\}|}{|M|}$.
3. Радіус дії / Дальність (r): Ця метрика (у спрощеному вигляді в коді реалізована як бінарний прапорець наявності міток) може бути розширена для перевірки фізичної можливості спостереження. Вона оцінює, чи знаходяться заявлені мітки в межах радіуса дії сенсорів дрона відносно його поточної позиції. Це дозволяє відсіювати «неможливі» спостереження.
4. Стабільність (s): Ця метрика оцінює часову узгодженість спостережень дрона. Вона розраховується як коефіцієнт Жаккара (*Jaccard index*) між поточним набором міток M_t та набором міток з попереднього раунду M_{t-1} . $s = \frac{|M_t \cap M_{t-1}|}{|M_t \cup M_{t-1}|}$. Різкі зміни у списку виявлених міток без відповідної зміни позиції дрона (низьке s) можуть вказувати на нестабільність роботи сенсорів або маніпуляції (*on-off attacks*).
5. Анти-спам нормування (q): Метрика, що запобігає атакам типу «flooding» кількістю міток. Вона зменшує вагу довіри, якщо дрон повідомляє про аномально велику кількість об'єктів, що перевищує розумний максимум k_{max} .
6. Узгодженість з історією (a): Перевірка, чи відповідає поточна пропозиція рішення, прийнятому в попередньому раунді (*Commit*). Якщо рій вирішив

летіти до мітки X , а дрон у наступному раунді продовжує пропонувати іншу дію або ігнорує X , метрика a знижується. Цей параметр дозволяє виявити дронів, які саботують колективні рішення.

Оновлення значення довіри реалізовано через логістичну функцію (сигмоїду) від зваженої суми цих метрик ($z = w^T x + b$) з подальшим згладжуванням за формулою експоненційного ковзного середнього (EWMA):

$$\tau_t = (1 - \gamma) \cdot \tau_{t-1} + \gamma \cdot \hat{\tau}_t, \quad (3.1)$$

де γ – коефіцієнт інерційності, а $\hat{\tau}_t$ – миттєва оцінка раунду (результат сигмоїди).

Використання EWMA дозволяє згладити випадкові флуктуації метрик (наприклад, через короточасну втрату видимості) та забезпечити стабільність оцінки довіри, водночас зберігаючи чутливість до систематичних змін у поведінці. Коефіцієнти ваг *weights* у лінійній комбінації дозволяють налаштовувати пріоритетність метрик (наприклад, надати більшу вагу колективному підтвердженню u або стабільності s).

Важливою складовою є програмна реалізація Менеджера ізоляції (*Quarantine Manager*) у методі *compute_quarantine*. Запропоноване програмне рішення базується на методі динамічного оцінювання [25] та використовує механізм гістерезису на основі лічильників:

- *q_streak*: Кількість послідовних раундів з низькою довірою. При досягненні *min_quarantine_rounds* вузол отримує статус *quarantined = True*. Це запобігає миттєвому блокуванню вузла через одиничний збій.
- *r_streak*: Кількість раундів з високою довірою, необхідна для виходу з карантину (*release_streak*). Цей механізм забезпечує «випробувальний термін» для вузлів, що повертаються до нормальної роботи.

Статус карантину діє як «жорсткий фільтр» (*Hard Gate*): у методах агрегації *_fuse_local_decision* пропозиції від ізольованих вузлів ігноруються програмно, що унеможливорює їхній вплив на вибір лідера або цілі. Це критичний механізм захисту, який дозволяє системі функціонувати навіть за наявності скомпрометованих вузлів, ізолюючи їх від процесу прийняття рішень.

Такі заходи дозволяють швидко та ефективно протидіяти атакам типу On-Off,

під час яких зловмисний вузол навмисно коливає свій рейтинг довіри навколо межі відключення. Застосування карантинного гістерезису повністю усуває дестабілізуючий ефект «миготіння» (flapping) станів мережі та попереджує лавиноподібне зростання службового трафіку [25]. Для верифікації цього механізму було проведено ізольоване моделювання, результати якого (рис. 3.9) доводять здатність системи надійно блокувати маніпуляції.

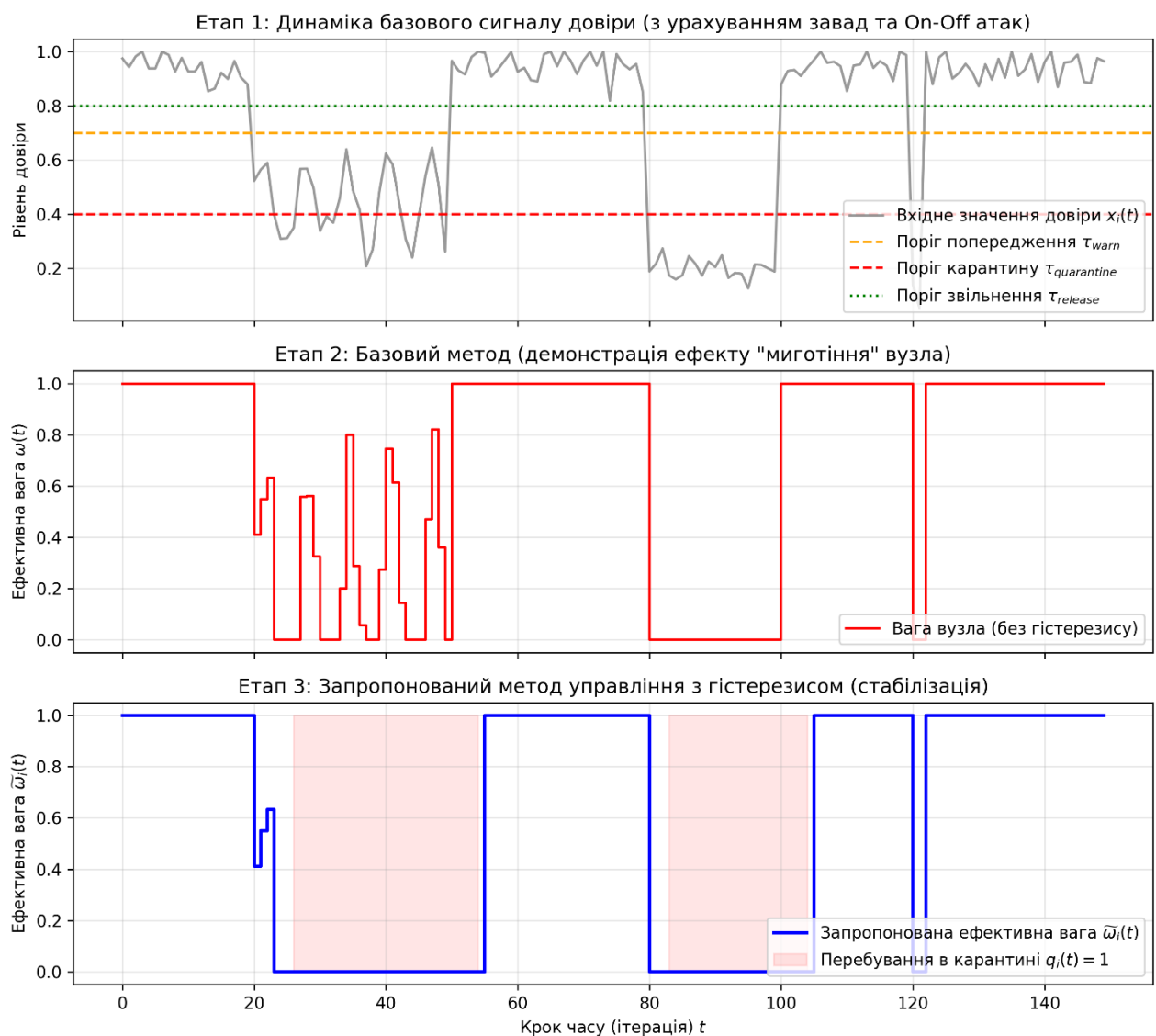


Рис. 3.9 Результати імітаційного моделювання: динаміка ваги вузла при On-Off атаці та стабілізація модифікованим гістерезисом

Після оновлення рівнів довіри та застосування карантину, система переходить до етапу прийняття рішення, реалізованого у методі `_fuse_local_decision`. Цей процес

базується на кластеризації отриманих пропозицій за координатами цілей (*_cluster_counts*) та подальшій багаторівневій фільтрації кандидатів. Для кожного кластера (потенційної цілі μ) розраховується набір статистик (*CandidateStats*), на основі яких застосовуються наступні фільтри («м'які гейти»).

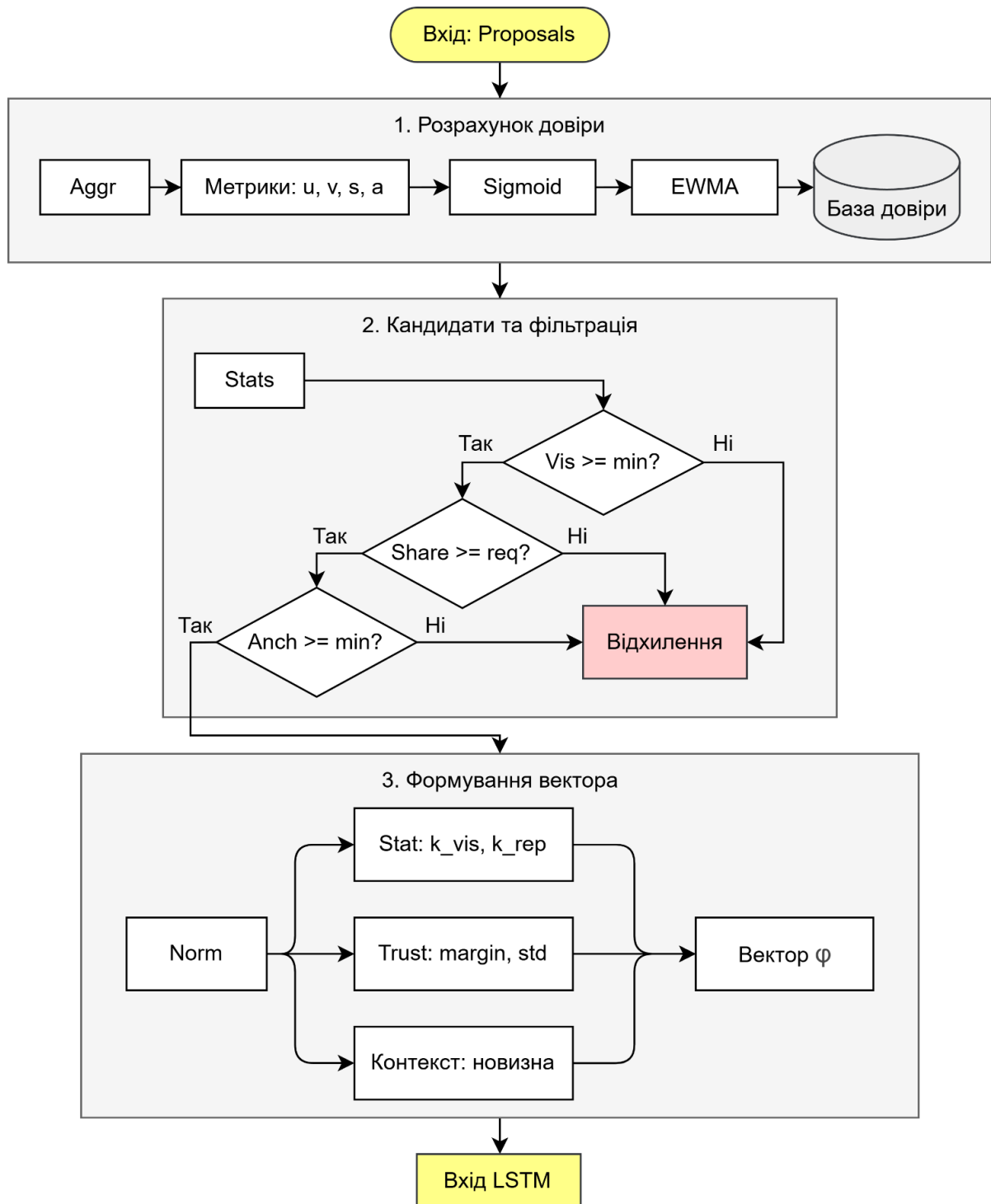


Рис. 3.10 Алгоритм підготовки вхідних параметрів для детермінованої та LSTM конфігурацій

Першим бар'єром є фільтр видимості ($Vis \geq N_{min}$), який відсіює пропозиції, про які повідомила недостатня кількість дронів. Параметр *min_visible_support* дозволяє ігнорувати випадкові шуми сенсорів або поодинокі галюцинації, вимагаючи підтвердження події мінімум кількома спостерігачами. Наступним кроком є застосування фільтру частки підтримки ($Share \geq Req$) – критичного механізму захисту від атак, коли зловмисники перебувають у меншості, але є активними. Система обчислює зважену частку підтримки:

$$Share = \frac{\sum_{i \in Reporters} \tau_i}{\sum_{j \in Visible} \tau_j}. \quad (3.2)$$

Кандидат розглядається лише тоді, коли *Share* перевищує динамічний поріг *visible_share_low* (або *visible_share_high* для випадків з малою кількістю спостерігачів). Це гарантує, що сумарна довіра до тих, хто «бачить» мітку, перевищує довіру до тих, хто її «ігнорує». Додатково застосовується фільтр якірної прив'язки ($Anch \geq K$), що вимагає наявності серед прихильників рішення щонайменше K «якірних» вузлів (*anchors*) – агентів, чия поточна довіра перевищує високий поріг *tau_anchor*. Цей механізм, реалізований параметром *min_anchor_count*, запобігає ситуаціям, коли група скомпрометованих ботів з низькою (але ненульовою) репутацією намагається «протиснути» хибне рішення чисельною перевагою. Кандидат, який успішно пройшов усі етапи фільтрації та набрав найбільшу суму зваженої довіри ($Score = \sum \tau_{pos}$), що перевищує встановлений поріг, обирається як локальне рішення (*Decision*).

Специфікою роїв БПЛА є можливість ситуацій, коли дрон знаходиться на периферії групи або в умовах сильної оклюзії, маючи лише 1-2 сусідів. Застосування стандартних статистичних методів тут неможливе через малу вибірку, тому в алгоритмі реалізовано окрему гілку логіки для умов розрідженої видимості (*Sparse Visibility Fallback*). Якщо кількість спостерігачів мала ($\leq small_vis_limit$), система перемикається на спеціальну логіку. Для випадку одного спостерігача рішення приймається лише якщо це сам дрон і його впевненість перевищує високий поріг *tau_single_min_one*. У випадку двох спостерігачів вимагається перехресне підтвердження (*self + peer*), де довіра до обох має бути

високою ($\tau_{single_min_other}$), а середня довіра – перевищувати $\tau_{pair_min_avg}$.

Для забезпечення гнучкості системи до динамічних умов місії та складних типів атак розроблено двомодульну архітектуру оцінювання довіри (рис. 3.11), яка поєднує нейромережвий та детермінований підходи. Обидва модулі оперують ідентичним простором вхідних ознак та інтегровані в єдиний конвеєр прийняття рішень, що дозволяє системі безшовно перемикатися між режимами або використовувати їх комбінацію для порівняльного аналізу ефективності.

Ключовим елементом адаптивності є клас *ThetaNNAdapter* (файл *nn_adapter.py*), в основі якого лежить легковагова рекурентна нейронна мережа на базі архітектури *LSTM*. Вибір *LSTM* зумовлений здатністю цієї архітектури виявляти часові залежності у поведінці вузлів, що є критичним для ідентифікації складних стратегій атак, таких як «*on-off attacks*», де злоумисник чергує нормальну та деструктивну поведінку.

Вхідний вектор ознак (*Feature Vector*), що формується методом *_build_nn_features*, є уніфікованим для обох підходів і складається з 19 нормалізованих компонентів, які детально характеризують контекст поточної пропозиції:

1. Статистика групи (4 ознаки): Кількість спостерігачів (k_{vis}), кількість тих, хто підтвердив (k_{rep}), нормований лічильник можливостей спостереження (opp_norm) та прапорець активності (opp_active_flag).
2. Метрики впевненості (4 ознаки): Запас позитивної довіри відносно базового порогу (pos_margin), частка негативних відгуків (neg_ratio), абсолютна частка підтримки ($share_abs$) та її надлишок над мінімумом ($share_surplus$).
3. Топологічні ознаки (4 ознаки): Частка та абсолютна кількість «якірних» вузлів ($anchors_ratio, anchors_len$), дисперсія довіри в групі (τ_{std}) та частка вузлів з низькою довірою (low_trust_ratio).
4. Контекстні прапорці (3 ознаки): Індикатори новизни цілі ($novelty_flag$), факту попереднього відвідування ($visited_flag$) та попередніх помилкових

відвідувань (*failed_flag*).

5. Часові та просторові показники (4 ознаки): Нормована відстань до цілі (*distance_local_norm*) та час, що пройшов з моменту останніх подій (коміту або візиту) для цієї цілі або глобально в системі.

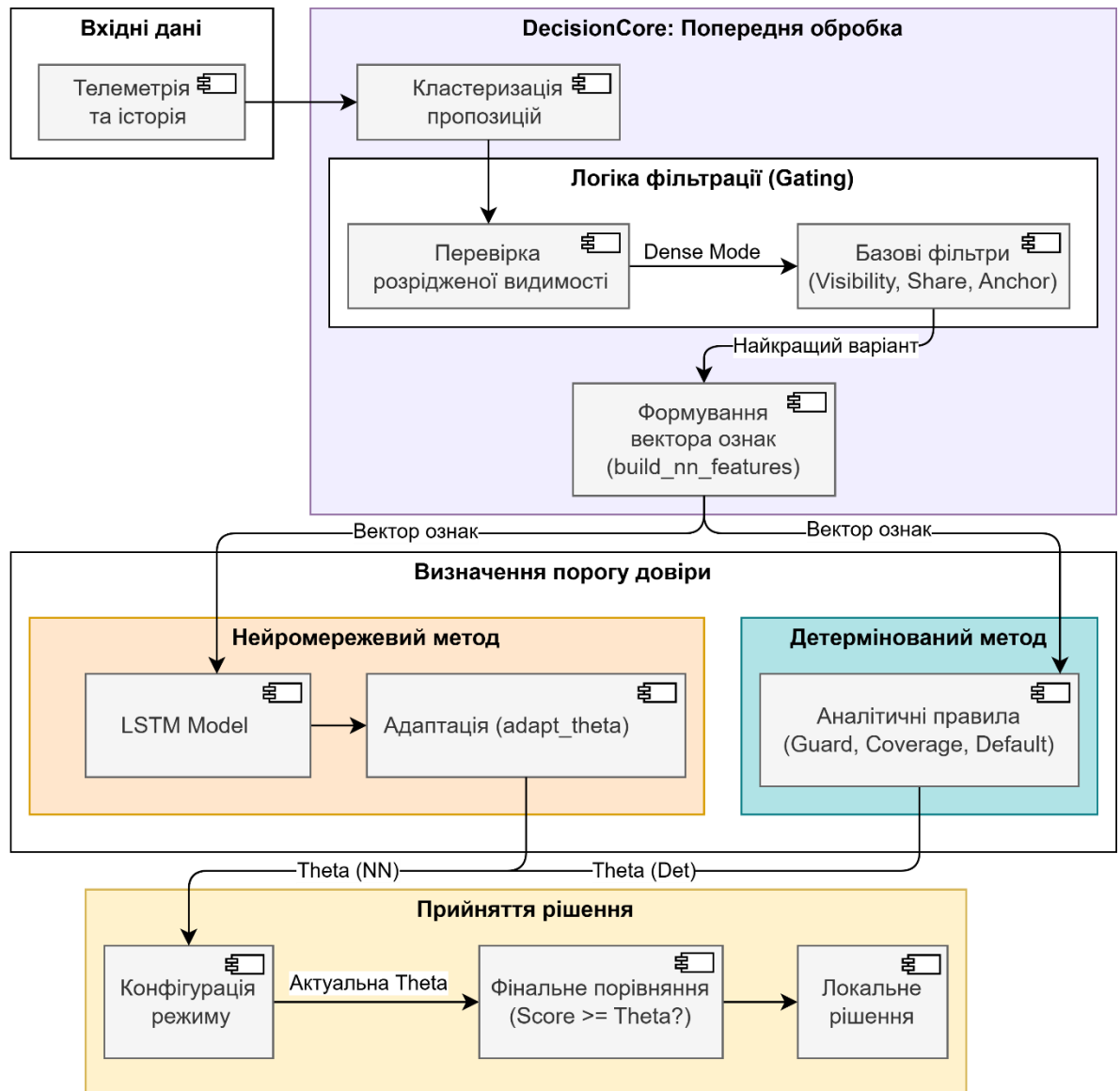


Рис. 3.11 Двомодульна архітектура оцінювання довіри

Архітектура нейромережевої моделі оптимізована для виконання на бортових обчислювачах з обмеженими ресурсами і складається з чотирьох послідовних шарів (рис. 3.12) та двох логічних блоків обробки:

1. *Input Layer*: Приймає послідовність векторів розмірністю 8×19 .
2. *Masking Layer*: Забезпечує коректну обробку часових рядів зі змінною

кількістю доступних історичних даних ($mask_value=0.0$), ігноруючи пропущені такти.

3. *LSTM Layer*: Шар довгої короточасної пам'яті з 16 прихованими нейронами ($units=16$), який виступає основним екстрактором часових патернів поведінки.
4. *Hidden Dense Layer*: Повнозв'язний шар з 8 нейронами та функцією активації *ReLU*, що виконує проміжну компресію ознак перед фінальною класифікацією.
5. *Output Layer*: Вихідний шар з 1 нейроном (активація *Sigmoid*), що формує ймовірнісну оцінку $p \in [0,1]$.

Вихід мережі p використовується для динамічного коригування порогу θ за формулою: $\theta_{new} = \theta_0 - \Delta_{scale} \cdot (p - 0.5) \cdot 2$. Це дозволяє системі ставати «м'якшою» або «суворішою» залежно від впевненості моделі.

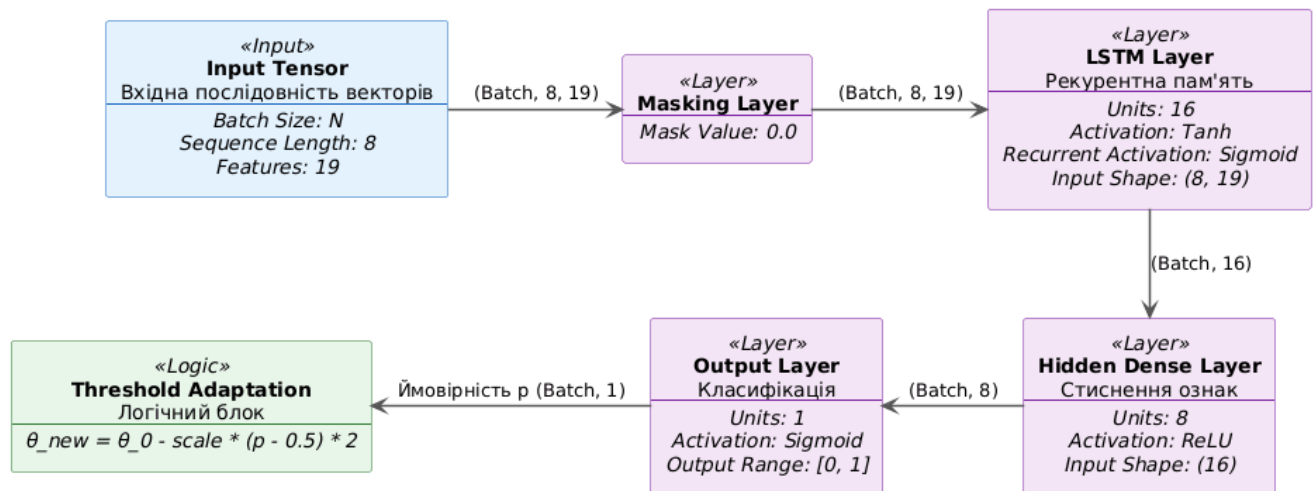


Рис. 3.12 Архітектура LSTM-нейромережі

У випадках, коли використання нейромережі неможливе або недоцільне (наприклад, для формування базової лінії порівняння), система працює в детермінованому режимі. Цей режим базується на аналітичних правилах (*Fallback Logic*), які також забезпечують динамічну адаптацію поведінки, але через явні евристичні механізми:

1. Механізм Guard: Дозволяє знизити поріг прийняття рішення, якщо частка

підтримки (*share*) або запас позитивної довіри (*pos_margin*) перевищують жорстко задані захисні інтервали (*lower_pos_window*), що свідчить про високу функціональну стабільність пропозиції навіть без санкції нейромережі.

2. Механізм *Coverage*: Стимулює дослідження нових цілей (*novelty*), знижуючи вимоги до довіри для кандидатів, які неодноразово ігнорувалися системою (*min_opp_rounds*), за умови відсутності попередніх помилок (*failed_flag*).
3. Обробка розрідженої видимості: Для ситуацій з 1-2 спостерігачами застосовується окрема гілка логіки з підвищеними вимогами до індивідуальної довіри ($\tau \geq 0.78$), що компенсує відсутність статистики групи.

Така організація забезпечує надійність системи: детермінований алгоритм гарантує базову безпеку та передбачуваність поведінки за допомогою прозорих правил, тоді як нейромережевий компонент надає додатковий рівень адаптивності, виявляючи нелінійні кореляції між 19-ма компонентами вектора стану, які важко формалізувати вручну. Уніфікація вхідних даних та вихідних інтерфейсів (корекція θ) для обох методів спрощує архітектуру та забезпечує коректність їх порівняльного аналізу в експериментальній частині.

Для забезпечення об'єктивного порівняльного аналізу ефективності розроблених методів (нейромережевого та детермінованого) у програмний комплекс інтегровано підсистему моніторингу, реалізовану в модулі *metrics.py*. Архітектура підсистеми побудована за принципом пасивного спостерігача, що дозволяє збирати статистичні дані без впливу на логіку прийняття рішень у реальному часі. Збір даних відбувається на двох рівнях абстракції: оперативному (оцінка якості вхідних даних у кожному раунді) та стратегічному (оцінка результативності місії в цілому).

Критичним елементом для розрахунку метрик точності є наявність достовірної інформації про оточення. У класі *DroneController* реалізовано механізм динамічного відстеження очікуваних міток (*_expected_marks*).

Симулятор, маючи повну карту розташування цілей, на кожному такті оновлює список об'єктів, які фізично знаходяться в радіусі дії сенсорів дрона, незалежно від того, чи спрацювали сенсори коректно. Ця множина $M_{expected}$ передається у ядро *DecisionCore* разом із фактичними показаннями сенсорів $M_{reported}$, що дозволяє класифікувати кожне рішення.

Клас *PropAgg* відповідає за миттєву оцінку якості консенсусу. На етапі завершення раунду (метод *decide* у *decision_core.py*) виконується попарне порівняння множин очікуваних та отриманих даних. Для розрахунку використовуються наступні базові показники матриці помилок:

- *TP (True Positive)*: Кількість реально існуючих міток, які були коректно ідентифіковані та підтверджені групою ($|M_{expected} \cap M_{reported}|$).
- *FP (False Positive)*: Кількість «галюцинацій» або хибних міток, нав'язаних злоумисниками ($|M_{reported} \setminus M_{expected}|$).
- *FN (False Negative)*: Кількість реальних міток, які були пропущені системою ($|M_{expected} \setminus M_{reported}|$).

На основі цих значень для кожного раунду t обчислюються:

- Точність (*Precision*): $P_t = \frac{TP}{TP+FP}$. Відображає стійкість системи до ін'єкції хибних даних (*Sybil/Spoofing attacks*).
- Повнота (*Recall*): $R_t = \frac{TP}{TP+FN}$. Характеризує здатність системи виявляти цілі в умовах зашумленого середовища або атаки ігнорування.
- *F1-Score*: Гармонійне середнє між точністю та повнотою, що є інтегральним показником якості сприйняття.

Додатково *PropAgg* накопичує статистику адаптивного порогу θ . Фіксуються мінімальне (θ_{min}), максимальне (θ_{max}) та середнє (θ_{avg}) значення за місію. Це дозволяє аналізувати динаміку роботи нейромережі: висока варіативність θ свідчить про активну адаптацію до змін у поведінці сусідів.

Клас *CommitAgg* відстежує наслідки прийнятих рішень, фокусуючись на фізичних діях дрона. Основною подією для цього агрегатора є «вердикт

відвідування» (*on_visit_verdict*), який генерується, коли дрон прибуває в точку цільового призначення. Метрики цього рівня включають:

- *Commit Rate*: Відношення кількості активних дій (польотів до цілей) до загальної кількості раундів прийняття рішень.
- *True/False Commits*: Абсолютна кількість успішних відвідувань реальних цілей та марних вильотів на хибні координати.
- Унікальність покриття (*Uniqueness*): Співвідношення унікальних знайдених цілей до загальної кількості візитів. Ця метрика дозволяє виявити неефективну поведінку, коли рій «зациклюється» на одній і тій самій цілі.
- *Visit Coverage*: Частка знайдених унікальних міток від загальної кількості міток на карті.

Усі розраховані метрики накопичуються в пам'яті протягом місії. При завершенні роботи (подія *_on_follow_path_end* у контролері) викликається метод *log_metrics*, який серіалізує дані у формат *JSON* або *CSV*. Структура вихідного файлу включає як агреговані показники (середні значення за місію), так і часові ряди зміни θ та точності, що дозволяє будувати графіки динаміки довіри в пост-аналізі.

Таким чином, розроблений програмний засіб реалізує гібридну дворівневу архітектуру для забезпечення стійкого консенсусу в рої БПЛА. Ця архітектура інтегрує класичні аналітичні методи з інтелектуальною адаптацією на основі рекурентних нейронних мереж. На базовому рівні функціонує система жорстких алгоритмічних фільтрів та механізмів гістерезису, таких як Менеджер ізоляції (*Quarantine Manager*) та багаторівнева фільтрація кандидатів (за видимістю, часткою підтримки та якірною прив'язкою). Цей рівень, реалізований як детермінований метод, гарантує безпеку та передбачуваність поведінки системи в граничних умовах (наприклад, при розрідженій видимості або на старті місії), забезпечуючи *Baseline* для порівняльного аналізу. На адаптивному рівні застосовується Нейромережевий адаптер (*ThetaNNAdapter*), який використовує легковагову мережу *LSTM*. Мета цього модуля – виявлення нелінійних та часових

патернів атак, недоступних для статичних правил. Мережа аналізує 19-компонентний вектор ознак, сформований з історії поведінки сусідів, та динамічно коригує поріг прийняття рішень (θ). Такий підхід дозволяє системі швидко адаптуватися до складних стратегій зловмисників (як-от «*on-off attacks*»), підвищуючи ефективність без шкоди для надійності. Критичною особливістю реалізації є уніфікація інтерфейсів, де обидва методи (детермінований та нейромережевий) оперують ідентичним простором ознак, а їхній вплив зводиться до маніпуляції єдиним параметром – порогом θ . Це дозволяє використовувати обидва підходи як рівноправні об'єкти для порівняння, що є основою експериментальних досліджень.

Вбудована підсистема збору метрик, розділена на *PropAgg* (для якості консенсусу в раунді) та *CommitAgg* (для результативності місії), забезпечує об'єктивну кількісну оцінку. Завдяки цьому інструментарію можливо точно визначити перевагу адаптивного методу над класичними підходами в умовах цілеспрямованої дезінформації.

3.3 Структура засобу для перевірки методу вибору лідера

Для комплексної оцінки ефективності запропонованого методу вибору лідера, що враховує метрики часу доставки повідомлень, було застосовано підхід симуляції, керованої трасами (*Trace-Driven Simulation*). Цей методологічний прийом дозволяє ізолювати логіку роботи алгоритму вибору від стохастичних флуктуацій мережевого середовища симулятора в реальному часі та забезпечити абсолютну відтворюваність експериментів при варіюванні вагових коефіцієнтів функції корисності.

Необхідність застосування такого підходу зумовлена специфікою моделювання у динамічному середовищі рою БПЛА засобами *Gazebo* та *ROS 2*, де на результати вимірювань суттєво впливають асинхронність виконання процесів, недетермінованість планувальника операційної системи та варіативність затримок при передачі повідомлень через міст *ros_gz_bridge*. Щоб усунути вплив цих побічних факторів при порівняльному аналізі різних конфігурацій алгоритму,

процес експерименту було розділено на дві фази. Перша фаза – це збір даних (*Data Acquisition*), що передбачає виконання польотних місій зі стандартними параметрами руху та запис повного логу станів усіх агентів. Друга фаза – апостеріорний аналіз (*Offline Replay Analysis*), під час якого відбувається реконструкція стану системи в кожен дискретний момент часу на основі записаних логів та розрахунок лідера за різними критеріями на одних і тих самих траєкторіях. Такий поділ дозволяє гарантувати проведення порівняння в суворо ідентичних умовах, виключаючи вплив випадкових факторів симуляції на результат вибору.

Практична реалізація цього підходу втілена у процедурі розрахунку лідера, яка виконується окремим програмним модулем-аналізатором *sla.py* (архітектура якого опублікована автором у [24] і наведена на рис. 3.13), що обробляє збережені траси польоту. Алгоритм працює ітеративно для кожного раунду взаємодії учасників рою, що відповідає частоті оновлення телеметрії в симуляторі, і складається з послідовності кроків.

Насамперед виконується парсинг та синхронізація логів: зчитуються координати $x_i(t)$, рівень заряду $E_i(t)$ та статусів сенсорів для кожного дрона $i \in V$, причому потоки даних від різних агентів синхронізуються за єдиною часовою шкалою симуляції. Далі відбувається реконструкція миттєвої топології: на основі координат будується граф зв'язності $G_t(V, E_t)$, де ребро (i, j) додається до множини E_t , якщо виконується умова комунікаційної досяжності:

$$|x_i(t) - x_j(t)| \leq R_{comm}, \quad (3.3)$$

де R_{comm} – радіус дії комунікаційного модуля, визначений сценарієм експерименту.

Генерація руху реалізована як рух групи за траєкторією центроїди з індивідуальними зсувами для кожного апарата (що відповідає моделі руху, дослідженій у [24] та відображеній на рис. 3.14). Базова траєкторія задається як еліпс на постійній висоті; допоміжний модуль формує орієнтації вздовж траєкторії та часи проходження контрольних точок відповідно до заданої швидкості.

Такий підхід застосовується для забезпечення стійкості формації рою дронів. Для кожного дрона обирається власний зсув від траєкторії центроїди, що утворює

бажану формацію. Вузол керування траєкторіями подає у плагін цільові точки послідовно; досягнення чергової точки визначається за близькістю вимірної позиції до цільової, після чого видається наступна точка. Швидкість для досягнення цільової точки подається разом із координатами цілі – це дозволяє синхронно проходити елементи траєкторії різної кривизни та різної довжини дуг для віддалених учасників формації.

Наступним кроком є розрахунок QoS -метрик. Для відновленого графа G_t обчислюється матриця найкоротших шляхів (у кількості хопів) між усіма парами вузлів. На основі цієї матриці для кожного кандидата розраховується теоретичний середній час доставки повідомлень $T_{avg}^i(t)$ згідно з математичною моделлю, описаною у другому розділі (формула 2.54).

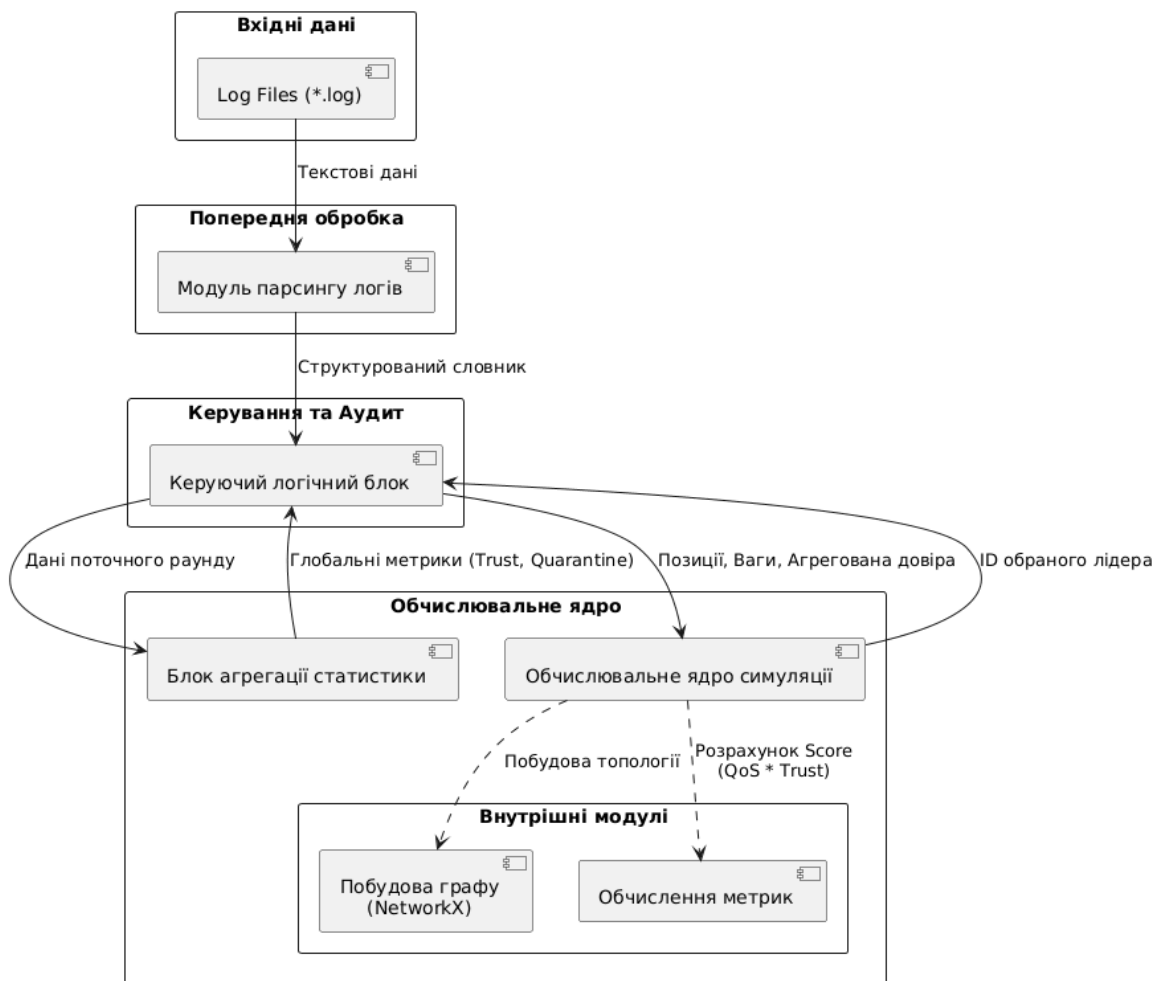


Рис. 3.13 Архітектура аналітичного інструментарію

Паралельно здійснюється інтеграція параметрів довіри: з логів модуля безпеки витягується значення динамічного коефіцієнта довіри $Trust_i(t)$, що був сформований системою під час польоту. Це дозволяє врахувати вплив історії поведінки вузла на його придатність до ролі лідера ще на етапі розрахунків. Завершується ітерація обчисленням вагової функції та визначенням лідера: для кожного вузла розраховується значення функції корисності $W_i(t)$ для заданого набору вагових коефіцієнтів ω . Порівняння проводиться для двох варіантів функції: базової U_{base} (енергія + топологія + час доставки + фільтр детермінованої довіри) та удосконаленої U_{ref} (енергія + топологія + час доставки + фільтр динамічної довіри, розрахованої за запропонованим методом), після чого лідером у момент часу t визнається вузол з максимальним значенням.

Для технічного забезпечення описаної методики розроблено спеціалізований аналітичний інструментарій на мові *Python*. Структурно скрипт реалізований як набір взаємопов'язаних функціональних блоків:

- Модуль парсингу логів (функція *parse_logs_with_quarantine*), що відповідає за десеріалізацію логів та перетворення їх у часові ряди станів.
- Блок агрегації статистики (функція *get_aggregated_stats*), який формує глобальну карту довіри та статусу карантину на основі розподілених даних.
- Обчислювальне ядро симуляції (функція *simulate_election*), яке виконує роль побудовника топології (на базі бібліотеки *NetworkX*) та калькулятора метрик, імплементуючи математичні моделі функцій корисності.
- Керуючий логічний блок (*Main execution loop*), що моделює хронологічний процес виборів, фіксує моменти зміни лідера та розраховує інтегральні характеристики функціональної стабільності обраного вузла.

Система виконує автоматизований аудит ефективності алгоритмів консенсусу, фокусуючись на метриках довіри (*Trust*) та стійкості до компрометації вузлів (*Security/Quarantine*). Застосування цього інструментарію дозволило провести масове моделювання вибору лідера на еталонних траєкторіях з широким діапазоном варіювання вагових коефіцієнтів, що забезпечило високу точність порівняльного аналізу без необхідності багаторазового перезапуску

ресурсомісткої симуляції в *Gazebo*.

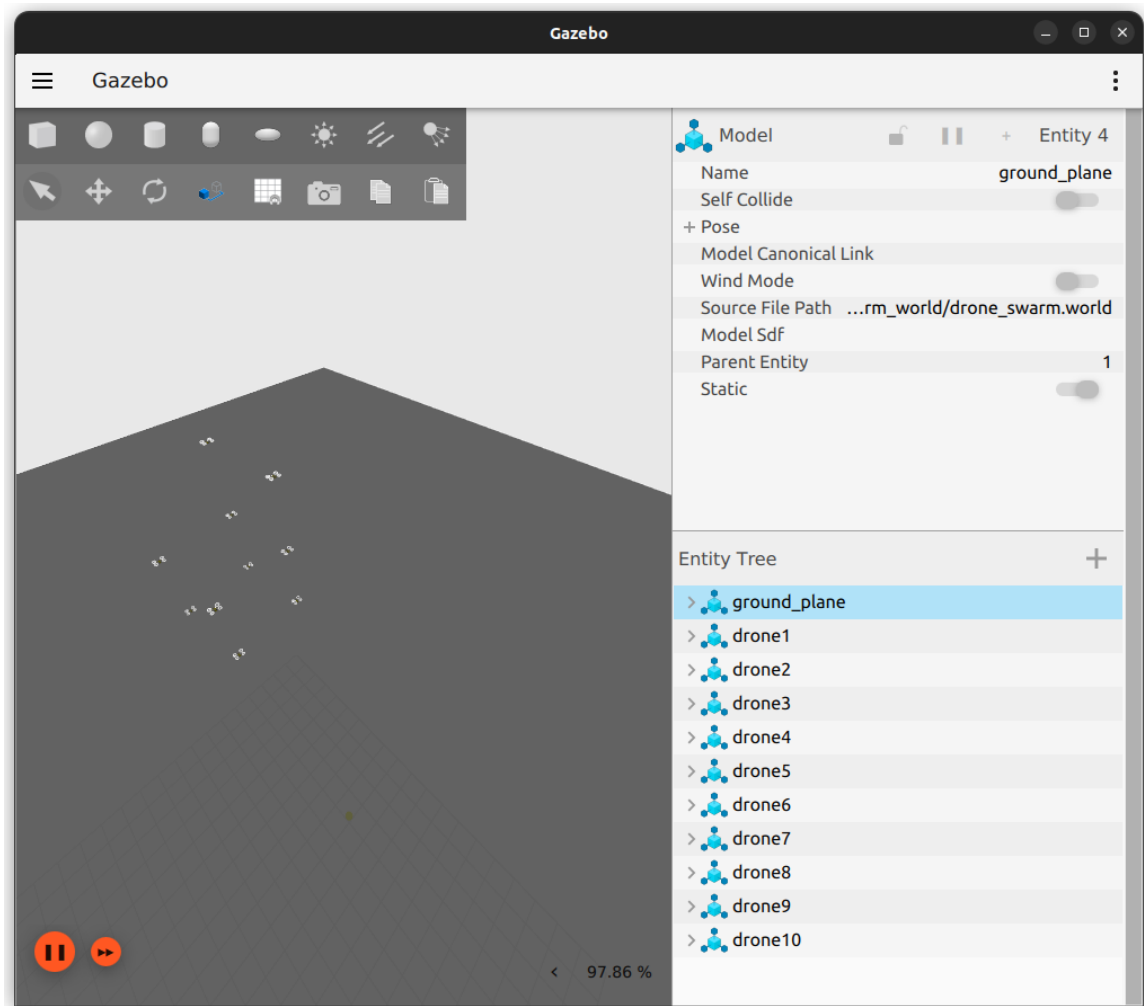


Рис. 3.14 Симуляція польоту рою з 10 дронів у кубічній формації

Усі компоненти симуляції працюють у багатопотоковому виконанні за допомогою багатопотокового виконавця ROS 2, що дає змогу паралельно обслуговувати окремі вузли дронів і збирача статистики, не блокуючи цикл симуляції. Частоти публікацій і періоди таймерів фіксуються у конфігурації експерименту; журнали станів і агрегованих метрик зберігаються з прив'язкою до параметрів сценарію, радіуса зв'язку та значення затримки на хоп. Такий підхід забезпечує відтворюваність експериментів і прозорість порівняння критеріїв вибору лідера в єдиному симуляційному середовищі ROS 2 + Gazebo з мостом `ros_gz` [4], [89].

Висновки до розділу 3

У третьому розділі вирішено задачу програмної реалізації та інструментального забезпечення для експериментальної перевірки теоретичних методів. Створено комплексний програмний полігон, що дозволяє моделювати поведінку рою безпілотних апаратів в умовах невизначеності та візантійських атак. Основні наукові та практичні результати розділу полягають у наступному:

1. Обґрунтовано та розгорнуто архітектуру симуляційного середовища. Для забезпечення реалістичності фізичної взаємодії та відтворюваності експериментів обрано технологічний стек *ROS 2 + Gazebo* [4], [89]. Реалізовано гібридну схему розробки: низькорівневе керування рухом винесено у C++ плагіни для забезпечення стабільності фізичного циклу, а високорівнева логіка консенсусу та довіри реалізована мовою *Python* для гнучкості налаштувань. Такий підхід дозволив досягти балансу між продуктивністю симуляції та зручністю збору аналітичних даних.

2. Реалізовано програмний засіб узгодження рішень із динамічною оцінкою довіри. Розроблено модульну архітектуру засобу, яка імплементує логіку стійкого консенсусу.

- Інтегровано модуль, що використовує легковагову *LSTM*-нейромережу для адаптивного прогнозування порогу довіри θ на основі вектора ознак.
- У рамках засобу програмно реалізовано механізми жорсткої фільтрації (*Hard Gate*) через менеджер карантину з гістерезисом та м'якої фільтрації (*Soft Gate*) через зважування голосів, що дозволяє ізолювати скомпрометовані вузли від процесу прийняття рішень.
- Забезпечено можливість роботи засобу у двох режимах – детермінованому (на основі евристик) та нейромережевому, що є необхідною умовою для проведення порівняльного аналізу ефективності.

3. Створено інструментальний засіб для перевірки методу вибору лідера. Розроблено спеціалізований аналітичний модуль для проведення симуляції, керованої трасами (*Trace-Driven Simulation*).

- Цей підхід дозволив ізолювати логіку вибору лідера від стохастичних

мережевих затримок симулятора, забезпечуючи абсолютну відтворюваність експериментів.

- Реалізовано алгоритми реконструкції миттєвої топології мережі та розрахунку *QoS*-метрик (затримки, енергія) на основі логів польоту, що дозволяє перевірити ефективність запропонованої мультиплікативної функції корисності порівняно з базовими адитивними моделями.

4. Впроваджено підсистему збору та агрегації метрик. У розроблений засіб інтегровано класи-агрегатори, які в режимі реального часу розраховують показники точності (*Precision*), повноти (*Recall*), *F1*-міри та успішності виконання місії (*Commit Rate*). Реалізовано механізм класифікації подій на основі порівняння з еталонним станом середовища («*Ground Truth*»), що дозволяє об'єктивно оцінити здатність засобу протидіяти атакам типу «ін'єкція хибних даних» та «саботаж».

Створений засіб є повністю готовим до проведення серії імітаційних експериментів для підтвердження гіпотез щодо переваг нейромережевого методу динамічної оцінки довіри та комплексної методики вибору лідера.

РОЗДІЛ 4

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ДОВІРИ ТА ВИБОРУ ЛІДЕРА

4.1. Методика досліджень та опис тестового стенду

Для проведення імітаційного моделювання, навчання нейронної мережі та верифікації запропонованих методів динамічної оцінки довіри й вибору лідера було розгорнуто експериментальний стенд на базі високопродуктивної обчислювальної станції. Вибір апаратної платформи зумовлений необхідністю забезпечити стабільну роботу фізичного симулятора в реальному часі паралельно з виконанням обчислювально ємних алгоритмів штучного інтелекту.

Апаратна конфігурація стенду включає наступні компоненти:

- Центральний процесор (CPU): *AMD Ryzen 5 PRO 4650G* (6 ядер, 12 потоків, базова частота 3.7 ГГц). Висока багатопотокова продуктивність процесора дозволила емулювати роботу розподілених обчислювальних вузлів кожного окремого БПЛА у рої без виникнення критичних затримок (*latency*), що могли б спотворити часові характеристики протоколів узгодження.
- Оперативна пам'ять (RAM): 32 ГБ *DDR4*. Такий обсяг пам'яті забезпечив можливість одночасного утримання в пам'яті станів симуляції для групи дронів, журналів подій та вагових коефіцієнтів нейронної мережі без використання файлу підкачки (*swap*), що гарантувало високу швидкість доступу до даних.
- Графічний прискорювач (GPU): *NVIDIA GeForce RTX 3060*. Відеокарта використовувалася для апаратного прискорення навчання рекурентної нейронної мережі (*LSTM*) та виконання інференсу моделі в процесі симуляції, а також для рендерингу візуального середовища.

Програмна реалізація комплексу виконана в середовищі операційної системи *Ubuntu 24.04 LTS*, яка є стандартом де-факто для розгортання робототехнічних систем завдяки стабільності ядра *Linux* та широкій підтримці спільноти.

Технологічний стек програмного забезпечення включає:

- Мову програмування *Python 3*: Використана для реалізації високорівневої логіки агентів, алгоритмів консенсусу та обробки даних.
- Бібліотеки машинного навчання: Для реалізації та навчання моделі *LSTM* використовувалися спеціалізовані бібліотеки (*PyTorch/TensorFlow*), оптимізовані для роботи з *CUDA*-ядрами графічного прискорювача.
- Середовище симуляції: Взаємодія агентів та фізика польоту моделювалися з використанням зв'язки *ROS 2* та симулятора *Gazebo*, що дозволило максимально наблизити умови експерименту до реальних польотних випробувань.

Важливим результатом технічного аналізу розробленого програмного засобу стала оцінка його ресурсоемності. Під час проведення серії експериментів із симуляцією повного циклу функціонування рою споживання оперативної пам'яті процесом моделювання коливалося в межах 2–4 ГБ.

Такий показник свідчить про високу оптимізацію розроблених алгоритмів. Враховуючи, що сучасні бортові комп'ютери для БПЛА (класу *NVIDIA Jetson* або *Raspberry Pi 4/5*) зазвичай оснащуються 4–8 ГБ оперативної пам'яті, отримані дані підтверджують можливість практичної імплементації запропонованого методу динамічної оцінки довіри безпосередньо на борту літальних апаратів без необхідності використання важкого серверного обладнання чи постійного зв'язку з наземною станцією керування. Низьке споживання ресурсів є критичною перевагою для автономних систем, що функціонують в умовах обмеженого енергопостачання та обчислювальної потужності.

Ключовим етапом експериментального дослідження стало формування репрезентативної вибірки тестових сценаріїв, що дозволяють оцінити стійкість та адаптивність запропонованого методу в умовах невизначеності. Оскільки отримання реальних польотних даних у необхідному обсязі обмежене специфікою проведення натурних випробувань та не покриває всього спектру можливих загроз, було застосовано метод генерації синтетичних наборів даних із контрольованими параметрами складності.

Для забезпечення статистичної достовірності отриманих результатів було згенеровано 20 унікальних польотних завдань (маршрутів). Кожне завдання являє собою незалежний сценарій польоту групи безпілотних літальних апаратів, що характеризується унікальною топологією просторового розміщення об'єктів інтересу. Використання такої кількості різнорідних сценаріїв дозволило нівелювати вплив специфіки конкретної конфігурації контрольних точок на підсумкові показники ефективності та уникнути ефекту перенавчання параметрів алгоритму під окремий маршрут.

Генерація маршрутів здійснювалася шляхом випадкового варіювання ключових параметрів польотного завдання у заданих діапазонах, що відповідають типовим умовам експлуатації розвідувальних груп БПЛА:

1. Протяжність маршруту (кількість етапів прийняття рішень): Довжина згенерованих маршрутів варіювалася від 10 до 20 навігаційних сегментів. Короткі маршрути (10 сегментів) дозволяють оцінити коректність роботи алгоритму та виявлення аномалій на початкових етапах функціонування системи, тоді як довші маршрути (до 20 сегментів) необхідні для перевірки стабільності роботи системи в динаміці та здатності до відновлення рівня довіри після завершення періодів деструктивного впливу.
2. Щільність істинних об'єктів: Кількість дійсних цільових точок на кожному маршруті варіювалася в межах від 10 до 30. Цей параметр визначає насиченість інформаційного потоку корисними даними. Варіативність кількості точок дозволяє перевірити здатність групи ефективно забезпечувати повноту покриття зони інтересу як при розрідженому, так і при щільному розташуванні цілей.
3. Інтенсивність завад: Кількість хибних міток, що імітують наслідки кібернетичних атак (наприклад, ін'єкція хибних даних) або апаратні збої сенсорів, становила від 3 до 30, але при цьому ніколи не перевищувала кількість дійсних точок у конкретному маршруті. Широкий діапазон кількості хибних точок дозволив змодельовати середовища з різним рівнем загроз:

- Середовище з низьким рівнем загрози: Кількість хибних точок є незначною порівняно з кількістю справжніх (співвідношення наближається до 1:5).
- Середовище з середнім рівнем загрози: Помірна кількість завад, що створює локальні конфлікти даних.
- Середовище з високим рівнем загрози: Кількість завад є співмірною з кількістю реальних об'єктів (співвідношення наближається до 1:1). Такі сценарії є найбільш складними для обробки, оскільки вимагають від алгоритму високої вибіркової здатності для відокремлення корисного сигналу від шуму.

Таким чином, сформований набір даних охоплює широкий спектр ситуацій – від штатних режимів роботи до умов масованої інформаційної атаки, що створює надійну базу для верифікації розроблених алгоритмів.

Специфіка задачі керування децентралізованим роєм в умовах інформаційної протидії вимагає комплексного підходу до оцінювання ефективності. Оскільки система функціонує на двох рівнях – індивідуальному (сприйняття та аналіз даних окремим агентом) та колективному (досягнення консенсусу та фізичне переміщення), використання лише стандартних інтегральних показників не дозволяє повною мірою розкрити динаміку процесів адаптації.

У зв'язку з цим, розроблена система моніторингу розділяє метрики на дві функціональні групи: метрики локального рівня прийняття рішень (*PA – Proposal Aggregation*) та метрики глобального рівня виконання місії (*CA – Commit Aggregation*).

Перша група показників характеризує якість роботи алгоритму оцінки довіри (як нейромережевого, так і детермінованого) на етапі формування пропозицій, тобто до моменту досягнення консенсусу. Вона відображає здатність окремого агента коректно класифікувати вхідні навігаційні дані як «істинні» або «хибні». Для формалізації оцінки використовується матриця помилок (*Confusion Matrix*), адаптована до контексту навігаційної задачі:

- Істинно-позитивне рішення (*True Positive, TP*): Агент приймає рішення

«рухатися до цілі», і дана ціль є верифікованим об'єктом. Це свідчить про коректну ідентифікацію валідної навігаційної точки.

- Істинно-негативне рішення (*True Negative, TN*): Агент вирішує ігнорувати сигнал і «слідувати маршрутом», і в даній локації дійсно відсутні валідні цілі. Це характеризує здатність системи ігнорувати інформаційний шум.
- Хибно-позитивне рішення (*False Positive, FP*): Агент ініціює рух до хибної цілі («рухатися до цілі»), що є наслідком успішної атаки типу «ін'єкція даних» або помилки сенсорів.
- Хибно-негативне рішення (*False Negative, FN*): Агент ігнорує валідну ціль («слідувати маршрутом»), що призводить до пропуску об'єкта інтересу.

Окремим важливим показником для запропонованого нейромережевого методу є статистика адаптивного порогу (θ -statistics). Оскільки модель *LSTM* динамічно змінює поріг впевненості (θ) залежно від контексту, аналізуються такі його характеристики:

- θ_{min} (*Theta Min*) – мінімальне значення порогу за час місії, що вказує на моменти максимальної схильності системи до ризику;
- θ_{max} (*Theta Max*) – максимальне значення, що свідчить про роботу в режимі суворої фільтрації;
- μ_{θ} (*Theta Mean*) – середній рівень вимогливості до достовірності даних.

У той час як локальні метрики описують процес прийняття рішень окремими агентами, наступна група показників (*CA*) оцінює безпосередню результативність фізичних дій рою, які є наслідком колективних рішень. Ключовим аспектом оцінювання на цьому рівні є аналіз унікальних відвідувань (*Unique Visits*). Це дозволяє уникнути дублювання статистики у випадках, коли одну й ту саму точку відвідують декілька агентів, і зосередитися на покритті території. Для порівняльного аналізу ефективності методів використовуються наступні інтегральні показники:

1. Точність відвідування (*Visit Precision*)

Визначає рівень безпеки місії та стійкість рою до дезінформації. Показник розраховується як відношення кількості унікальних відвіданих істинних цілей до

загальної кількості відвіданих точок (як істинних, так і хибних):

$$Precision = \frac{Visit_{true}^{unique}}{Visit_{true}^{unique} + Visit_{false}^{unique}}, \quad (4.1)$$

де $Visit_{true}^{unique}$ – кількість унікальних валідних цілей, успішно досягнутих роєм, а $Visit_{false}^{unique}$ – кількість помилкових відвідувань хибних міток. Значення $Precision \rightarrow 1.0$ свідчить про повну нейтралізацію атак на підміну навігаційних даних.

2. Повнота покриття (*Visit Recall* або *Coverage*)

Характеризує ступінь виконання поставленого польотного завдання. Розраховується як частка виявлених істинних цілей від їх загальної кількості на маршруті:

$$Recall = \frac{Visit_{true}^{unique}}{N_{total_valid_marks}}, \quad (4.2)$$

де $N_{total_valid_marks}$ – загальна кількість валідних об'єктів, що існували в середовищі під час місії. Низьке значення цього показника при високій точності може свідчити про надмірну обережність алгоритму.

3. Гармонійне середнє (*Visit F1-Score*)

Є узагальненою метрикою, що дозволяє знайти баланс між безпекою (*Precision*) та ефективністю пошуку (*Recall*):

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \quad (4.3)$$

У рамках даного дослідження пріоритетним критерієм ефективності визначено максимізацію показника *Visit Precision*, оскільки в умовах протидії критично важливим є недопущення скомпрометованих дій рою, навіть ціною незначного зниження повноти покриття.

4.2. Аналіз динаміки довіри та ефективності рішень агентів

Ключовим фактором, що визначає загальну стійкість рою безпілотних літальних апаратів до деструктивного інформаційного впливу, є здатність окремого агента самостійно верифікувати вхідні дані ще до моменту ініціювання

процедур колективного узгодження. Оскільки алгоритми консенсусу, такі як зважене голосування або *PBFT*, чутливі до частки зловмисних вузлів, якість первинної фільтрації навігаційних пропозицій на локальному рівні стає фундаментом безпеки всієї місії. У цьому підрозділі представлено детальний порівняльний аналіз ефективності функціонування агентів на етапі формування пропозицій (*Proposal Phase*), що базується на результатах серії експериментів загальною тривалістю 68 раундів прийняття рішень. Основна увага приділена дослідженню того, як застосування запропонованого нейромережевого підходу впливає на зниження ймовірності помилкових класифікацій об'єктів у порівнянні з традиційними детермінованими методами.

Для верифікації наукової гіпотези про перевагу адаптивних механізмів оцінювання довіри було проведено зіставлення результатів роботи двох алгоритмів: розробленого методу на базі рекурентної нейронної мережі (методу динамічної оцінки довіри) та класичного алгоритму *Trust-WCA* (*Trust-based Weighted Clustering Algorithm*). *Trust-WCA* є широко визнаним стандартом у галузі мобільних *ad-hoc* мереж (*MANET*) та роїв дронів, реалізуючи детермінований підхід до вибору лідера та формування кластерів на основі зваженої суми параметрів вузла. Саме статус *Trust-WCA* як класичного еталонного методу (*baseline*), що використовує фіксовані порогові значення довіри, обумовлює його вибір для порівняльного аналізу: це дозволяє наочно продемонструвати обмеження статичних підходів та переваги динамічної адаптації.

Ефективність алгоритмів оцінювалася через призму помилок першого та другого роду, де критично важливим завданням є пошук оптимального балансу між чутливістю системи до корисних сигналів та її стійкістю до хибних даних. Узагальнені статистичні показники, отримані в ході моделювання для групи з 10 агентів протягом 68 раундів, наведені в таблицях 4.1 та 4.2 та рисунку 4.1. Детальний аналіз наведених даних дозволяє виявити суттєві відмінності в «поведінці» досліджуваних алгоритмів, особливо в контексті обробки аномалій. Зокрема, найбільшу загрозу для безпеки місії становлять помилки першого роду (*False Positive*), які полягають у прийнятті агентом рішення про рух до хибної цілі.

Таблиця 4.1

Показники класифікації навігаційних об'єктів для методу *Trust-WCA*

<i>DroneID</i>	<i>TP</i>	<i>TN</i>	<i>FP</i>	<i>FN</i>	θ_{mean}	θ_{max}	θ_{min}
<i>drone1</i>	21	0	12	54	0.6	0.6	0.6
<i>drone2</i>	21	0	14	52	0.6	0.6	0.6
<i>drone3</i>	22	0	14	51	0.6	0.6	0.6
<i>drone4</i>	24	1	11	51	0.6	0.6	0.6
<i>drone5</i>	24	3	12	48	0.6	0.6	0.6
<i>drone6</i>	23	6	13	45	0.6	0.6	0.6
<i>drone7</i>	21	8	15	43	0.6	0.6	0.6
<i>drone8</i>	21	10	15	41	0.6	0.6	0.6
<i>drone9</i>	20	9	15	43	0.6	0.6	0.6
<i>drone10</i>	17	10	18	42	0.6	0.6	0.6

Таблиця 4.2

Показники класифікації навігаційних об'єктів для методу динамічної оцінки довіри

<i>DroneID</i>	<i>TP</i>	<i>TN</i>	<i>FP</i>	<i>FN</i>	θ_{mean}	θ_{max}	θ_{min}
<i>drone1</i>	19	0	14	35	0.5687	0.6235	0.4806
<i>drone2</i>	18	6	13	31	0.6117	0.915	0.4976
<i>drone3</i>	20	0	11	37	0.6046	0.7848	0.5017
<i>drone4</i>	24	0	7	37	0.6352	0.7506	0.6
<i>drone5</i>	23	2	11	32	0.6082	0.6183	0.6
<i>drone6</i>	23	6	9	30	0.6202	0.7438	0.5467
<i>drone7</i>	23	8	11	26	0.614	0.6846	0.5785
<i>drone8</i>	22	10	8	28	0.6067	0.826	0.4648
<i>drone9</i>	21	9	13	25	0.5933	0.6476	0.5522
<i>drone10</i>	15	7	13	33	0.6088	0.89	0.4791

Така помилка свідчить про успішність атаки типу «ін'єкція хибних даних» (*False Data Injection*), коли зловмисник змушує систему повірити в існування неіснуючого об'єкта (рис. 4.2). Як свідчать результати експерименту, інтеграція нейромережевого модуля дозволила зменшити кількість таких інцидентів на 20.9% (110 випадків проти 139 у методу *Trust-WCA*). Це покращення пояснюється здатністю мережі *LSTM* аналізувати часові патерни поведінки сусідів: навіть якщо миттєве значення репутації вузла формально перевищує поріг, нейромережа може виявити аномалію в динаміці (наприклад, різку зміну показників або

невідповідність історичному профілю) і знизити рівень довіри, ефективно фільтруючи недостовірну інформацію.

Такі можливості відіграють вирішальну роль у протидії *On-Off* атакам, коли порушник чергує періоди шкідливої та нормальної активності. Інерційність застосованого механізму гістерезису [25] усуває дестабілізуючий ефект «миготіння» (*flapping*) статусу вузла та формує стійкий динамічний поріг.

Не менш важливою є проблема надмірної консервативності систем захисту, яка проявляється у помилках другого роду (*False Negative*). У цьому випадку агент ігнорує дійсну ціль, помилково класифікуючи джерело інформації як функціонально нестабільне. Високий рівень *FN* призводить до «сліпоти» рою, зниження повноти розвідки та пропуску важливих цільових об'єктів. За цим показником метод *NN* продемонстрував ще більш виражену перевагу, знизивши кількість помилок на 33.2% (314 проти 470). Класичний алгоритм *Trust-WCA*, обмежений жорстко заданим порогом, часто відхиляв валідні дані, які мали незначні відхилення від ідеальних значень, тоді як нейромережевий підхід дозволяв врахувати контекст ситуації та позитивну історію взаємодії, зберігаючи високу чутливість сенсорної мережі.

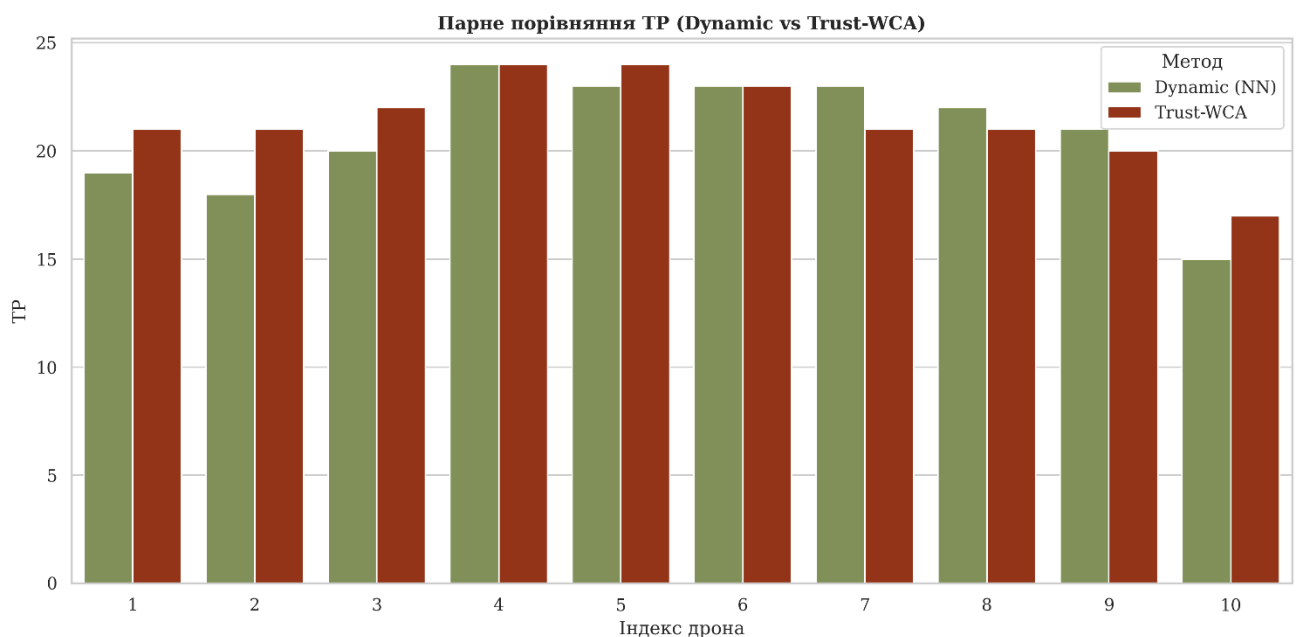


Рис. 4.1. Середня кількість прийняття локальних TP рішень

Фундаментальною причиною такої різниці в ефективності є механізм формування порогу прийняття рішень (θ). У класичних підходах, таких як *Trust-WCA*, цей параметр є статичною константою (в експерименті $\theta_{Trust-WCA} = 0.6$), що не дозволяє алгоритму адаптуватися до змінних умов середовища. Статичний поріг створює дилему: занадто високе значення робить систему безпечною, але неефективною (багато пропущених цілей), а занадто низьке – вразливою до атак.

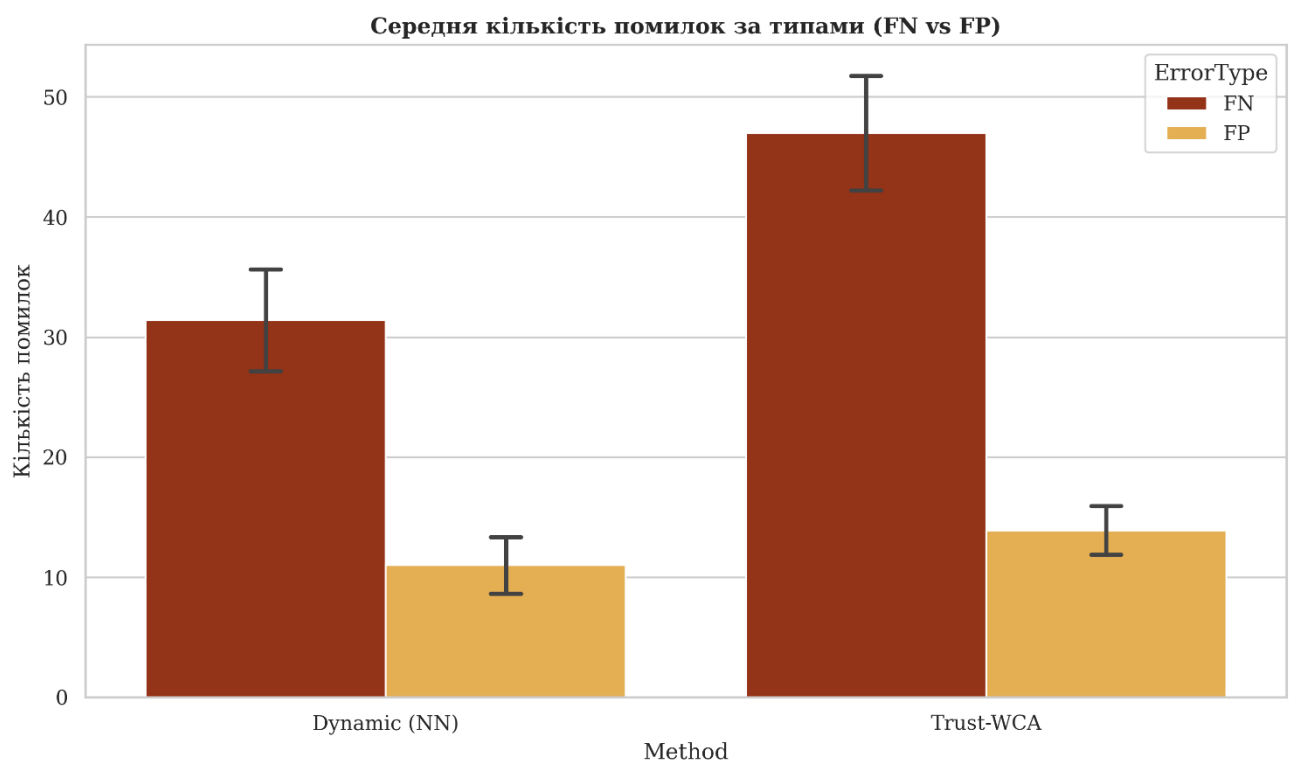


Рис. 4.2. Середня кількість допускання локальних помилок

Натомість, запропонований метод реалізує концепцію динамічного адаптивного порогу, який автоматично підлаштовується під поточний рівень невизначеності. Розподіл значень адаптивного порогу для окремих агентів наведено в таблицях 4.1 та 4.2, проілюстровано на рисунках 4.3 та 4.4.

Аналізуючи варіативність порогу θ , можна виділити два характерні режими роботи нейромережевого контролера. Перший режим – «реакція на загрозу», коли система фіксує ознаки атаки або неузгодженості даних.

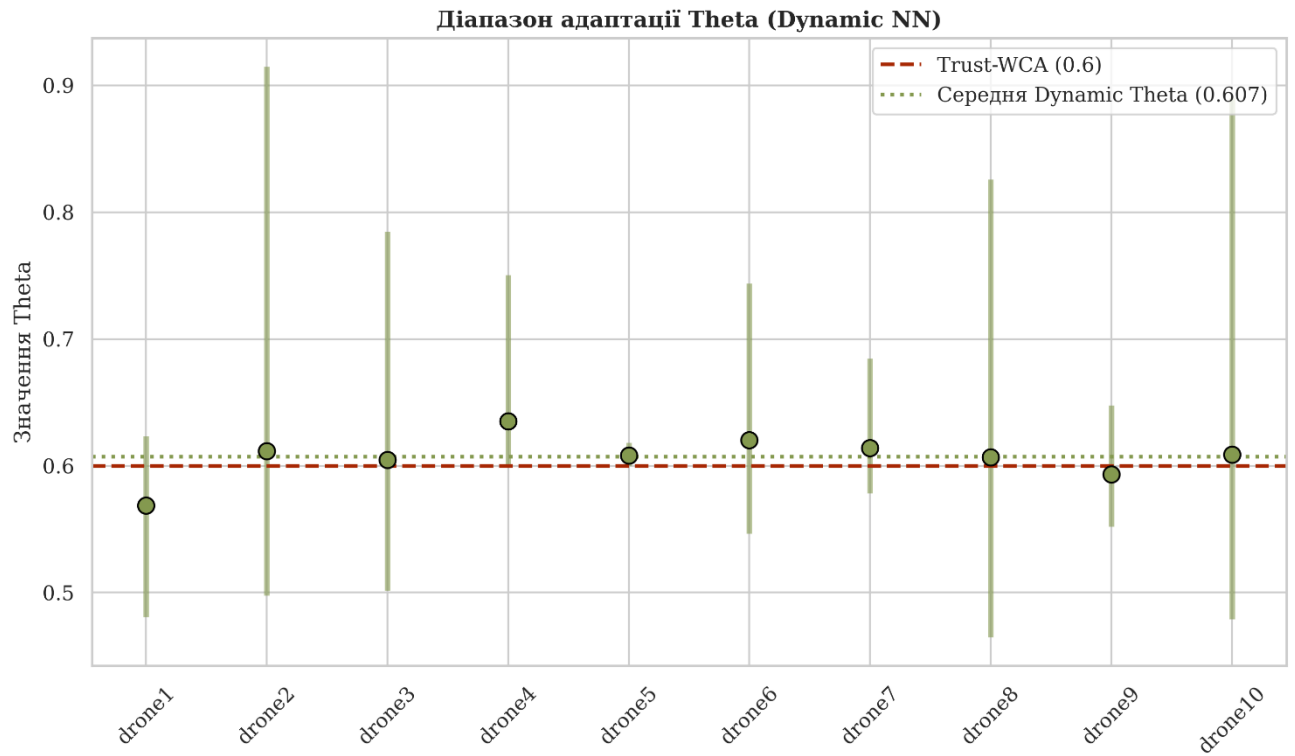


Рис. 4.3. Діапазон адаптації θ для методу динамічної оцінки довіри для кожного дрону

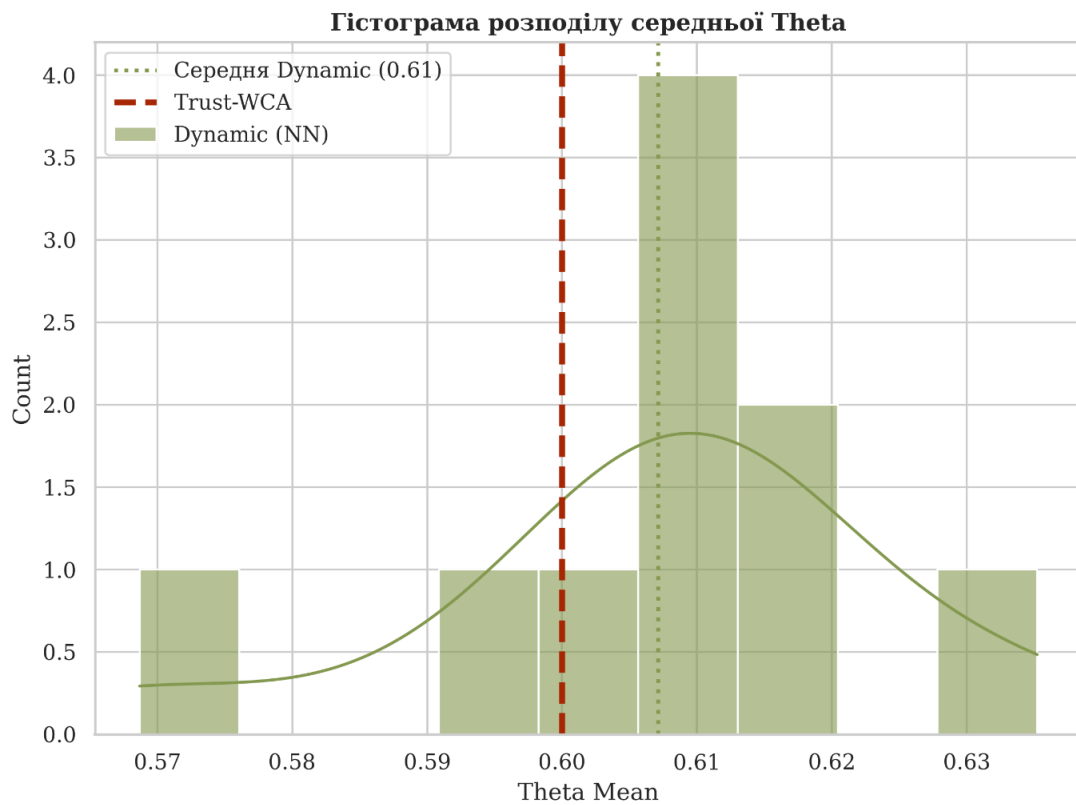


Рис. 4.4. Гістограма розподілу θ для методу динамічної оцінки довіри

У такі моменти поріг для окремих агентів (наприклад, drone2) піднімався до екстремальних значень 0.915, фактично переводячи агента в стан «нульової довіри». Це дозволило миттєво відсікти підозрілі джерела інформації, що було б неможливо при фіксованому порозі 0.6, який використовує Trust-WCA. Другий режим – «довірча взаємодія», що активується в умовах стабільного середовища та підтвердженої репутації сусідів.

У цьому стані поріг знижувався до 0.46–0.48, дозволяючи системі приймати рішення на основі менш формальних ознак і успішно детектувати цілі, які в іншому випадку були б відфільтровані як шум.

Проведені дослідження переконливо доводять, що інтеграція механізмів глибокого навчання у процедуру оцінки довіри дозволяє подолати обмеження статичних методів, забезпечуючи суттєве зниження помилок класифікації на локальному рівні агентів.

Окрім аналізу абсолютних значень помилок, важливим індикатором ефективності є динаміка відносних показників точності (*Precision*) та повноти (*Recall*). У середньому, впровадження нейромережевого підходу забезпечило зростання показника *Precision* на 4% по всьому рою. При цьому детальний аналіз по окремих дронах виявляє значну варіативність результатів: зміна точності коливається в діапазоні від -5% до +15%. Нижче наведено діаграму розподілу приросту точності (*Delta Precision*) по агентах (рис. 4.5).

Однак, варто зазначити, що висока точність індивідуальних рішень (*PA Metrics*) є лише необхідною, але не достатньою умовою для успіху всієї місії. Кінцевий результат функціонування рою залежить від того, наскільки ефективно ці локальні оцінки трансформуються у колективні дії через механізми досягнення консенсусу.

Чи здатна система зберегти свою стійкість, коли індивідуальні пропозиції об'єднуються у глобальне рішення, та як це впливає на фізичне виконання польотного завдання – ці питання потребують окремого розгляду. Тому наступним логічним кроком дослідження є аналіз метрик глобального рівня (*CA Metrics*).

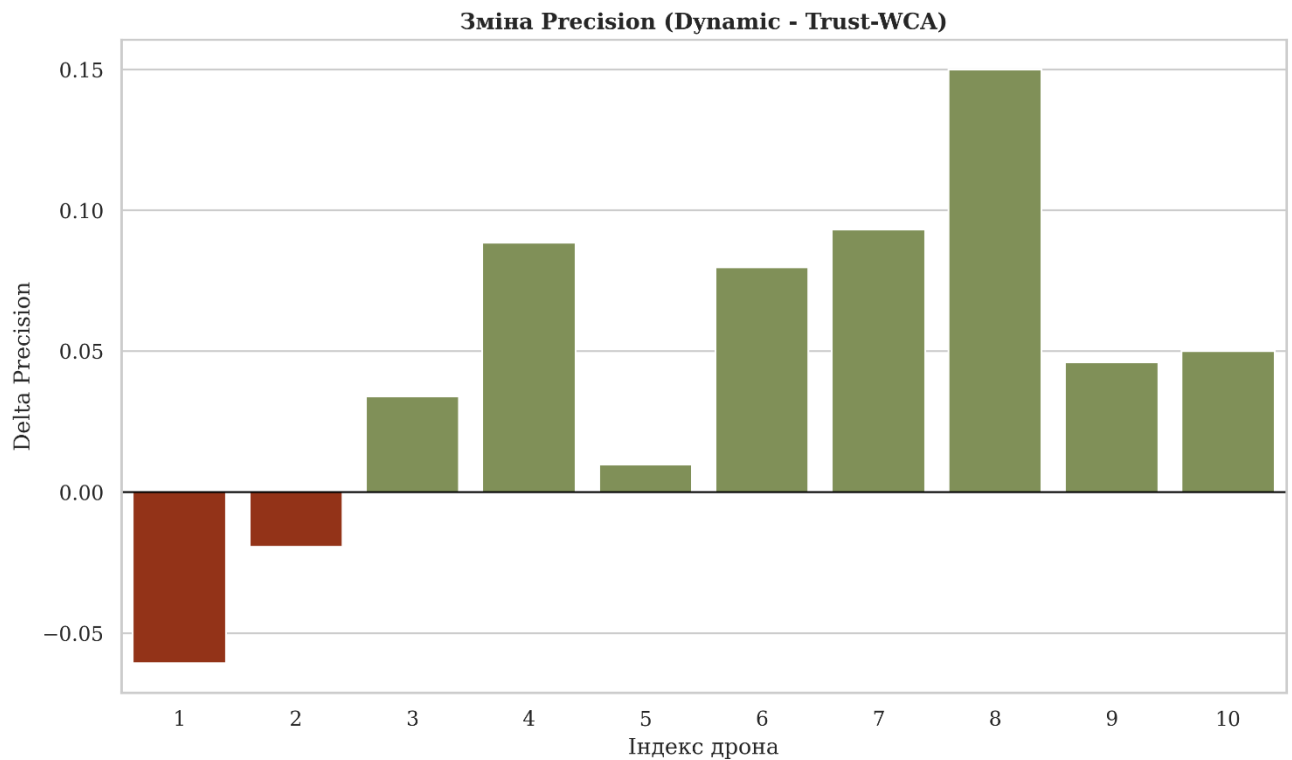


Рис. 4.5 Діаграма розподілу Precision по кожному з дронів

Варто наголосити, що в контексті розподілених систем керування кінцевою метою функціонування рою БПЛА є виконання поставленої задачі для всієї групи як єдиного організму, а не досягнення високих показників окремими дронами. У рамках досліджуваної навігаційної задачі цей критерій формалізується як максимізація кількості відвіданих коректних цільових точок при мінімізації випадків потрапляння у пастки чи зони хибних міток. Саме здатність рою колективно оминати загрози та ефективно покривати цільову область є головним індикатором успішності алгоритму.

Для отримання статистично значущих результатів було проведено моделювання 20 унікальних сценаріїв зі змінною топологією та різним рівнем агресивності середовища. Оцінювання проводилося за ключовими метриками ефективності: точністю (*Precision*), повнотою покриття (*Coverage*) та часткою хибних відвідувань (*False Rate*), результати експериментів відображені у таблицях 4.3 та 4.4.

Узагальнені результати порівняльного аналізу запропонованого методу

динамічної оцінки довіри та класичного алгоритму Trust-WCA наведені в таблиці 4.5. Фундаментальним критерієм надійності автономної системи в умовах інформаційної протидії є її здатність ігнорувати хибні цілі. Як свідчать узагальнені дані, запропонований метод динамічної оцінки довіри забезпечив середній рівень точності 0.809 (приблизно 81%), що на 8 відсоткових пунктів перевищує показник класичного алгоритму *Trust-WCA* (0.728). Це статистично значуща різниця, яка вказує на якісно вищий рівень фільтрації дезінформації на етапі досягнення консенсусу.

Таблиця 4.3

Результати експериментів з використанням методу динамічної оцінки довіри

Кількість цілей	Кількість відвіданих цілей	Кількість відвіданих пасток	<i>Precision</i>	<i>Coverage</i>	<i>F1</i>	<i>False rate</i>
26	21	5	0.8077	0.8077	0.8077	0.1923
21	18	5	0.7826	0.8571	0.8182	0.2174
14	10	3	0.7692	0.7143	0.7407	0.2308
14	12	3	0.8	0.8571	0.8276	0.2
18	15	3	0.8333	0.8333	0.8333	0.1667
22	15	4	0.7895	0.6818	0.7317	0.2105
27	15	5	0.75	0.5556	0.6383	0.25
30	30	9	0.7692	1	0.8695	0.2308
13	12	2	0.8571	0.9231	0.8889	0.1429
24	21	5	0.8077	0.875	0.84	0.1923
25	18	3	0.8571	0.72	0.7826	0.1429
27	20	5	0.8	0.7407	0.7692	0.2
30	29	4	0.8788	0.9667	0.9207	0.1212
20	10	3	0.7692	0.5	0.6061	0.2308
21	20	4	0.8333	0.9524	0.8889	0.1667
19	13	4	0.7647	0.6842	0.7222	0.2353
22	10	2	0.8333	0.4545	0.5882	0.1667
10	10	2	0.8333	1	0.9091	0.1667
28	14	4	0.7778	0.5	0.6087	0.2222
25	20	3	0.8696	0.8	0.8333	0.1304

Ще більш показовим є аналіз частки хибних відвідувань (*False Rate*) – метрики, що відображає відсоток ресурсів рою, витрачених на політ до неіснуючих або шкідливих об'єктів («фантомів»). Для алгоритму *Trust-WCA* цей показник у

окремих складних сценаріях сягав критичних значень у 30–35%. Це означає, що майже третина польотного часу та енергії групи витрачалася непродуктивно, що створює ризики для успішного завершення місії через вичерпання заряду батарей. Натомість, метод динамічної оцінки довіри продемонстрував значно вищу стабільність: показник *False Rate* утримувався в діапазоні 12–20% навіть у сценаріях з високою інтенсивністю атак.

Таблиця 4.4

Результати експериментів з використанням методу адаптивного зваженого
голосування

Кількість цілей	Кількість відвіданих цілей	Кількість відвіданих пасток	<i>Precision</i>	<i>Coverage</i>	<i>F1</i>	<i>False rate</i>
26	20	8	0.7143	0.7692	0.7407	0.2857
21	17	7	0.7083	0.8095	0.7555	0.2917
14	9	5	0.6429	0.6429	0.6429	0.3571
14	11	3	0.7857	0.7857	0.7857	0.2143
18	13	5	0.7222	0.7222	0.7222	0.2778
22	15	7	0.6818	0.6818	0.6818	0.3182
27	15	8	0.6522	0.5556	0.6	0.3478
30	27	10	0.7297	0.9	0.806	0.2703
13	12	3	0.8	0.9231	0.8572	0.2
24	21	8	0.7241	0.875	0.7924	0.2759
25	17	5	0.7727	0.68	0.7234	0.2273
27	18	8	0.6923	0.6667	0.6793	0.3077
30	27	6	0.8182	0.9	0.8572	0.1818
20	9	4	0.6923	0.45	0.5455	0.3077
21	20	5	0.8	0.9524	0.8696	0.2
19	12	7	0.6316	0.6316	0.6316	0.3684
22	9	3	0.75	0.4091	0.5294	0.25
10	10	3	0.7692	1	0.8695	0.2308
28	14	7	0.6667	0.5	0.5714	0.3333
25	18	4	0.8182	0.72	0.766	0.1818

Важливо було також перевірити, чи не призводить підвищена обережність системи та фільтрація загроз до «паралічу» прийняття рішень, коли рій уникає відвідування будь-яких точок, включно з валідними. Аналіз метрики покриття (*Coverage*) спростовує це побоювання.

Таблиця 4.5

Порівняння середніх показників ефективності виконання місії

Показник ефективності	<i>Trust-WCA</i> (Класичний)	Метод динамічної оцінки довіри (Запропонований)	Динаміка змін
Середня точність (<i>Mean Precision</i>)	0.72862	0.80917	+8%
Середня частка помилок (<i>Mean False Rate</i>)	0.27138	0.19083	-8%
Середнє покриття (<i>Mean Coverage</i>)	0.72874	0.771175	+4%
Середній <i>F1-Score</i>	0.721365	0.781245	+5.9%

У багатьох сценаріях метод динамічної оцінки довіри забезпечив навіть вище покриття (до 0.96), ніж *Trust-WCA*. Хоча в окремих випадках покриття запропонованого методу було співмірним або незначно нижчим за класичний аналог (рис. 4.6), це компенсувалося суттєво вищою точністю. Така поведінка свідчить про реалізацію раціональної стратегії: алгоритм «жертвує» відвідуванням сумнівних точок заради гарантування безпеки групи, але загалом зберігає високу ефективність пошуку, про що свідчить зростання інтегрального показника *F1-Score* на 5.9%.

Високі показники точності виконання місії є прямим наслідком вдосконаленого механізму вибору лідера розподіленої системи. У децентралізованих системах саме лідер часто формує фінальний вектор руху групи, тому його функціональна стабільність є критичною. Високий рівень помилок у методі *Trust-WCA* пояснюється недостатньою ізоляцією скомпрометованих вузлів: оскільки цей алгоритм використовує статичні вагові коефіцієнти, зловмисник із формально коректною поведінкою (наприклад, високим рівнем заряду) міг отримати статус лідера і спрямувати групу на хибну ціль. Застосування методу динамічної оцінки довіри кардинально змінює ситуацію. Завдяки використанню рекурентної нейронної мережі, будь-які аномалії в історії поведінки кандидата призводять до різкого зниження його рейтингу, створюючи ефект «м'якої ізоляції».

Кластеризація за типом експерименту (Dynamic vs Trust-WCA)

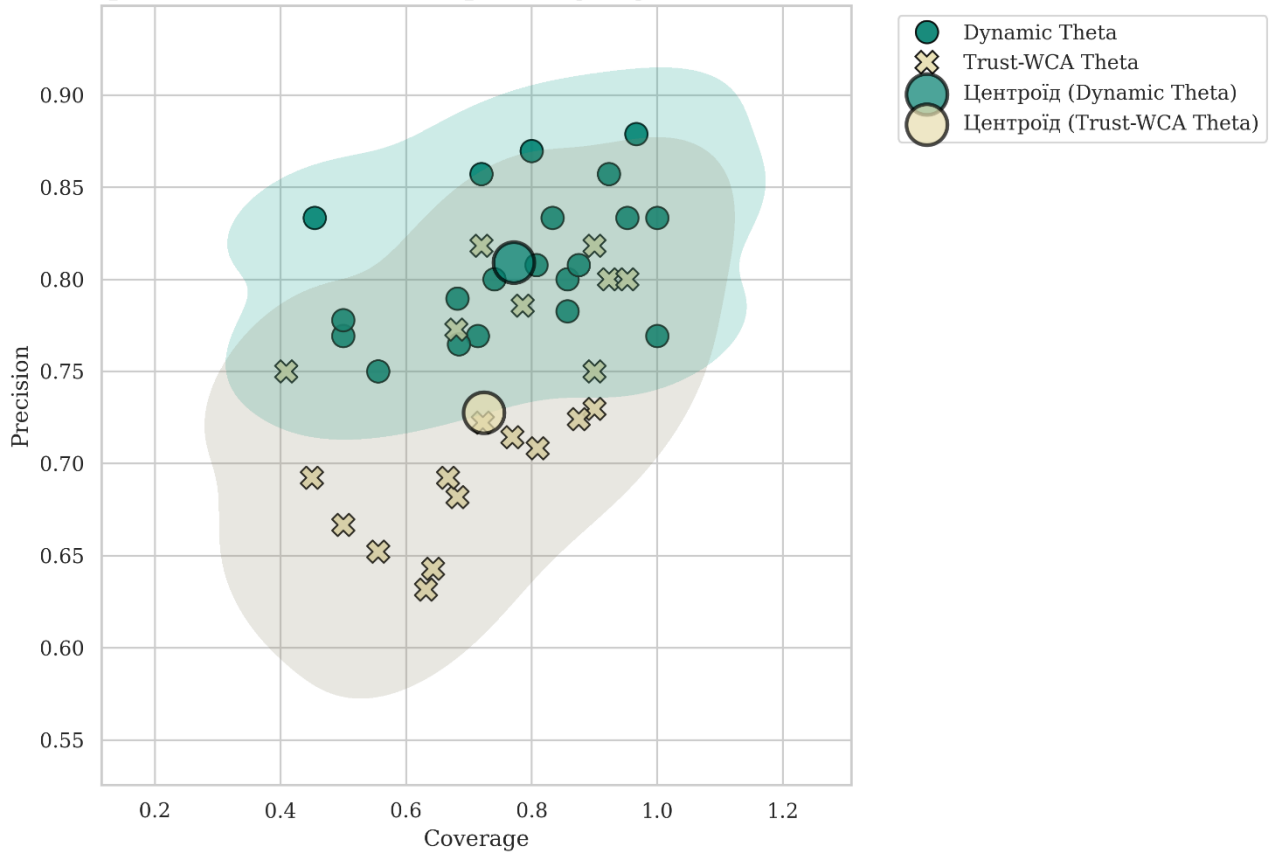


Рис. 4.6 Співвідношення Precision та Coverage (точності та покриття) протягом 20 експериментів

Підсумовуючи результати експериментального моделювання (рис. 4.7), можна стверджувати, що розроблений метод довів свою практичну ефективність.

Порівняльний аналіз дозволив виділити наступні ключові переваги запропонованого підходу:

- Підвищення безпеки місії: Досягнуто зростання середньої точності відвідування цілей на 8%, що мінімізує ризики фізичної втрати дронів або демаскування групи через хибні маневри.
- Оптимізація ресурсів: Зниження частки непродуктивних польотів до хибних цілей з 27% до 19% дозволяє значно економити енергетичний ресурс рою та збільшувати тривалість активної фази операції.
- Стійкість управління: Підтверджено здатність методу ефективно усувати скомпрометованих учасників від процесу управління, забезпечуючи

стабільність виконання місії в умовах динамічної зміни поведінки злоумисників.

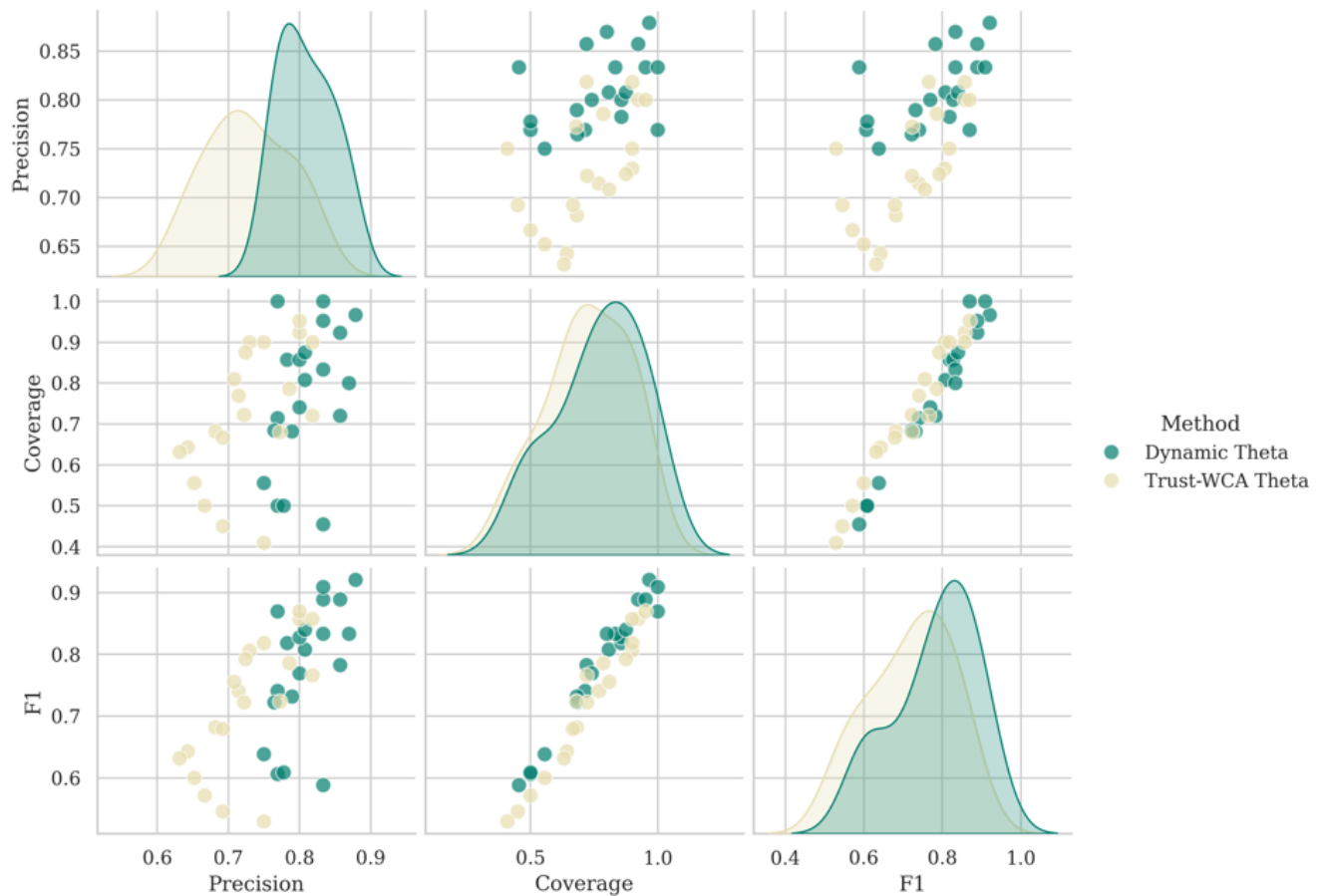


Рис. 4.7 Співвідношення метрик рою у проведених експериментах

Таким чином, отримані результати підтверджують, що метод динамічної оцінки довіри дозволяє створити надійну систему керування роєм, яка здатна адаптуватися до складних умов інформаційної протидії.

4.3. Порівняльний аналіз алгоритмів вибору лідера на польотних даних

Завершальним етапом експериментального дослідження стала верифікація запропонованого алгоритму вибору лідера розподіленої системи. Оскільки функціональна стабільність вузла-лідера є критичною умовою для безпечного функціонування децентралізованих систем, необхідно було перевірити наукову гіпотезу про те, що динамічне врахування довіри (через адаптивний параметр θ) дозволяє ефективніше ізолювати скомпрометовані вузли, ніж класичні підходи, що

базуються на статичних вагових коефіцієнтах.

Для забезпечення чистоти експерименту та відтворюваності результатів (принцип *ceteris paribus* – за інших рівних умов) було застосовано метод ретроспективного аналізу (*Trace-Driven Simulation*). Замість генерації нових синтетичних даних, порівняння алгоритмів проводилося на основі реальних логів координат (x, y, z) та значень довіри, отриманих у ході експериментів, описаних у підрозділі 4.2. Такий підхід дозволив поставити обидва досліджувані алгоритми в абсолютно ідентичні топологічні та контекстні умови, виключаючи вплив випадкових факторів генерації маршруту.

Для виявлення реальних відмінностей між алгоритмами під час налаштування параметрів моделювання було відкинуто тривіальний сценарій «абсолютної безпеки», де вага довіри встановлюється домінуючою. У такому сценарії обидва методи обирали б однаково функціонально стабільних, але топологічно неефективних лідерів, що робило б задачу їх порівняння складнішою. Натомість було змодельовано граничні умови, де вага компоненти довіри була знижена до рівня, що змушує систему балансувати між вибором топологічно вигідного лідера (наприклад, геометричний центр системи) та лідера з бездоганною репутацією. Це дозволило оцінити алгоритми у критичних ситуаціях. Порівняння проводилося між двома підходами:

1. Класичний метод (*Trust-WCA*): Базується на алгоритмі *Weighted Clustering Algorithm* з додаванням параметру довіри. Використовує фіксовані вагові коефіцієнти та лінійне врахування репутації.
2. Запропонований адаптивний метод: Використовує динамічний параметр θ , що обчислюється рекурентною нейронною мережею залежно від контексту.

Універсальна формула розрахунку ваги (W_i) кандидата в лідери, що використовувалася для обох методів, має вигляд:

$$W_i = \left(w_1 \cdot E_i + w_2 \cdot d_i + w_3 \cdot \left(1 - \frac{T_i^{avg}}{T_{max}} \right) \right) \cdot Trust_i^{2\theta}, \quad (4.4)$$

де:

E_i – нормалізований рівень залишкової енергії (в умовах експерименту приймався константою для ізоляції факторів);

d_i – степінь зв'язності вузла (*Node Degree*);

T_i^{avg} – середня кількість хопів до інших членів розподіленої системи (*Average Hop Distance*);

T_{max} – нормалізуючий коефіцієнт діаметра мережі;

$Trust_i$ – поточний рівень довіри до вузла;

θ – параметр важливості довіри.

Ключова відмінність полягає в природі параметру θ . У класичному методі *Trust-WCA* θ є константою ($\theta = 0.6$). У запропонованому методі θ змінюється динамічно в діапазоні $[0.46, 0.92]$ на основі виходу *LSTM*-мережі. Знаходження θ у показнику степеня забезпечує нелінійну реакцію системи: при високих значеннях θ (підозра на атаку) навіть незначне зниження довіри ($Trust < 1.0$) призводить до різкого зменшення підсумкової ваги W_i , що гарантовано виключає підозрілий вузол з перегонів за лідерство, навіть якщо його топологічні параметри є ідеальними.

Для об'єктивного порівняння алгоритмів введено систему метрик, що дозволяє кількісно оцінити баланс між безпекою (здатністю уникати загроз) та топологічною ефективністю (якістю обслуговування мережі).

1. Частота компрометації лідера (*Liar Frequency*, F_{liar}). Ця метрика є головним індикатором безпеки системи. Вона визначає частку часу, протягом якого лідером кластера було обрано вузол, що в даний момент часу є скомпрометованим (зловмисником або несправним).

$$F_{liar} = \frac{1}{T} \sum_{t=1}^T I(L_t \in M_t), \quad (4.5)$$

де:

T – загальна кількість раундів виборів;

L_t – ідентифікатор вузла, обраного лідером у раунді t ;

M_t – множина скомпрометованих вузлів («зловмисників») у раунді t ;

$I(\cdot)$ – індикаторна функція, що дорівнює 1, якщо умова виконується, і 0 в іншому випадку.

Інтерпретація: Чим нижче значення F_{liar} , тим стійкішою є система до захоплення керування.

2. Показник функціональної стабільності лідера (S_f). Ця метрика безпосередньо доповнює попередню та відображає здатність системи зберігати безпечне керування протягом заданого інтервалу часу без інцидентів компрометації. Розраховується за формулою:

$$S_f = \frac{T_{lead}}{T_{total}} \cdot 100\%, \quad (4.6)$$

де:

T_{lead} – сумарний час (кількість циклів), протягом яких лідером залишався функціонально стабільний (нескомпрометований) вузол;

T_{total} – загальний час функціонування системи (або тривалість експерименту).

3. Коефіцієнт топологічної ефективності (Topology Efficiency Score, E_{topo}). Ця метрика характеризує якість обслуговування (QoS) з точки зору зв'язності мережі. Вона показує, наскільки обраний лідер близький до «ідеального» геометричного центру графа в даний момент часу. Розрахунок базується на нормалізації топологічної ваги обраного лідера відносно максимально можливої ваги в поточній конфігурації мережі:

$$E_{topo} = \frac{1}{T} \sum_{t=1}^T \frac{W_{topo}(L_t)}{\max_{v \in V} W_{topo}(v)}, \quad (4.7)$$

де:

$W_{topo}(v)$ – топологічна складова ваги вузла v (без урахування довіри), що розраховується як $w_1 \cdot E_v + w_2 \cdot d_v + w_3 \cdot \left(1 - \frac{T_v^{avg}}{T_{max}}\right)$;

$\max_{v \in V} W_{topo}(v)$ – вага «ідеального» кандидата в поточному раунді (еталон).

Інтерпретація: Значення $E_{topo} \approx 1.0$ означає, що система завжди обирає вузол з найкращою позицією (центр). Зниження показника свідчить про те, що алгоритм змушений обирати субоптимальних лідерів (наприклад, на периферії) заради інших цілей (безпеки).

4. Середні топологічні втрати (Avg Topology Loss, L_{topo}). Метрика відображає «ціну безпеки» – наскільки погіршується топологія мережі через відмову від обрання підозрілих, але топологічно вигідних кандидатів.

$$L_{topo} = 1 - E_{topo}. \quad (4.8)$$

Для детального розуміння механізму прийняття рішень було проаналізовано фрагмент роботи системи (Раунди 0–200), протягом якого в рої були присутні скомпрометовані вузли, що періодично займали вигідні топологічні позиції. Детальний розбір механізму м'якої ізоляції для цих моментів часу вперше висвітлено автором у публікації [24]. Результати порівняльного аналізу для характерних моментів часу наведено в таблиці 4.6.

Аналіз даних таблиці 4.4 демонструє характерну поведінку алгоритмів у критичних ситуаціях:

- Ігнорування загрози класичним методом (Раунд 3): Алгоритм *Trust-WCA* обрав лідером Дрон №2. Цей вузол мав ідеальну топологічну оцінку (1.000) та високу формальну довіру (0.99). Проте фактично він був скомпрометований (*Status*: 0), що призвело до помилки вибору. Статична вага довіри виявилася недостатньою, щоб переважити вигоду від топологічного положення.
- Успішна адаптація (Раунд 3): Запропонований метод у тій самій ситуації відхилив кандидатуру Дрона №2 і обрав Дрона №8. Хоча Дрон №8 мав дещо гіршу топологічну позицію (*Topology Score* 0.950 проти 1.000), він був функціонально стабільним (*Status*: 1). Це приклад свідомого компромісу: метод пожертвував 5% топологічної ефективності заради гарантування безпеки, що стало можливим завдяки підвищенню параметру θ у відповідь на нестабільність середовища.
- Реакція на зниження репутації (Раунд 8): Ситуація повторилася, коли

довіра до Дрона №2 знизилася до 0.90. *Trust-WCA* продовжив обирати його через домінуючу топологічну оцінку. Адаптивний метод, використовуючи нелінійне штрафування ($0.90^{2\theta}$), різко знизив вагу цього кандидата і обрав безпечного Дрона №4.

Загалом у розглянутій вибірці класичний метод припустився 6 помилок (обрав лідера-зловмисника), тоді як адаптивний метод – лише 4. Здатність уникати лідерів з низькою довірою у моменти, коли вони є топологічно привабливими, підтверджує ефективність динамічного регулювання θ .

Таблиця 4.6

Фрагмент порівняльного аналізу вибору лідера (вибірка показових раундів)

Раунд	Theta	Drone ID	Trust	Dynamic			Trust-WCA		
				QoS	QoS Loss	Status	QoS	QoS Loss	Status
0	0.58	4	0.65	1	0	0	1	0	0
1	0.57	8	0.77	0.969	0.031	1	0.95	0.05	1
2	0.57	5	0.9	1	0	1	0.975	0	1
3	0.57	8	0.99	0.95	0.05	1	1	0	0
4	0.57	2	0.99	0.8	0	1	1	0	1
5	0.56	9	0.99	1	0	1	1	0	1
6	0.56	2	0.99	0.833	0	1	1	0	1
7	0.57	1	0.99	1	0	1	0.833	0.083	1
8	0.56	4	0.96	1	0	1	1	0	0
9	0.57	8	0.93	0.625	0.375	1	0.792	0.208	1
10	0.57	8	0.99	1	0	1	1	0	1
11	0.58	5	0.99	1	0	0	1	0	1
12	0.57	8	0.99	1	0	1	0.95	0.05	1
13	0.57	3	0.99	0.9	0	0	1	0	1
14	0.57	2	0.99	1	0	1	1	0	1
15	0.57	2	0.99	0.944	0	1	1	0	0
16	0.58	8	0.99	0.95	0.05	1	1	0	1
17	0.57	6	0.99	0.833	0.167	1	1	0	1
18	0.57	2	0.99	1	0	0	1	0	0
19	0.58	2	0.99	1	0	1	1	0	0
20	0.57	2	0.99	1	0	1	1	0	1

Для отримання узагальненої оцінки було проаналізовано повний масив даних (понад 200 раундів симуляції). Порівняння проводилося за двома конфліктуючими критеріями: функціональною стабільністю (Liar Frequency) та якістю обслуговування мережі (QoS, зважена сума параметрів виключаючи довіру, що характеризує ефективність комунікацій системи при обраному лідері, формула 4.4). Узагальнені результати ефективності інтеграції динамічної експоненти, які висвітлено у [24], демонструють зниження частоти обрання скомпрометованих лідерів без суттєвої втрати мережевої якості обслуговування (таблиця 4.7).

Необхідно пояснити природу високих значень коефіцієнта топологічної ефективності (понад 0.93) для обох методів. Методика розрахунку передбачає нормалізацію: у кожному раунді прийняття рішень агент із найкращими топологічними показниками (максимальна кількість сусідів, мінімальна середня відстань) отримує еталонну оцінку 1.0. Оцінки всіх інших агентів нормуються відносно цього лідера в діапазоні [0,1].

Таблиця 4.7

Аналіз інтегральних показників ефективності

Показник	Класичний метод (Trust-WCA)	Адаптивний метод (Запропонований)	Покращення
Liar Frequency (Частота помилок)	6.0%	5.3%	11.6% (відн.)
Коефіцієнт топологічної ефективності	0.9356	0.9434	0.8%
Avg Topology Loss	0.0344	0.0284	17.4% (відн.)

Оскільки класичний метод (*Trust-WCA*) має тенденцію ігнорувати незначні коливання довіри, він майже завжди обирає саме цього «ідеального» агента, забезпечуючи значення метрики, близьке до 1.0. Запропонований адаптивний метод у критичних ситуаціях свідомо відхиляє кандидатуру лідера з оцінкою 1.0, якщо він є скомпрометованим, і обирає «наступного найкращого» функціонально стабільного кандидата. У щільних розподілених системах різниця між позицією ідеального центру та його найближчих функціонально стабільних сусідів є мінімальною (наприклад, 1.0 проти 0.95), що пояснює незначне відхилення у

чисельних показниках топології при суттєвому виграші в безпеці.

Відповідно до результатів *Trace-Driven* симуляції, висвітлених у [24], отримані результати демонструють що запропонований метод показав покращення за обома напрямками одночасно:

- Безпека: Частота обрання скомпрометованих лідерів (*Liar Frequency*) знизилася з 6.0% до 5.3%. Це доводить, що динамічне θ спрацьовує як запобіжник саме в ті моменти, коли ризик компрометації є найвищим.
- Ефективність (*QoS*): Всупереч очікуванням, середній коефіцієнт топологічної ефективності лідерів не знизився, а зріс (0.9434 проти 0.9356). Це свідчить про те, що «чесні» дрони, яких пріоритезував адаптивний алгоритм, часто займали навіть кращі позиції в довгостроковій перспективі, ніж ті, на які помилково вказував класичний метод, орієнтуючись лише на миттєву вигоду.

Проведений ретроспективний аналіз підтвердив, що використання динамічного параметру θ як показника степеня у ваговій формулі вибору лідера дозволяє реалізувати нелінійний адаптивний фільтр. В умовах налаштування системи на граничну чутливість (баланс між довірою та *QoS*), запропонований метод продемонстрував здатність утримувати високий рівень якості обслуговування розподіленої системи (Коефіцієнт ефективності > 0.94) при одночасній мінімізації ймовірності захоплення контролю злоумисниками. Це дозволяє рекомендувати розроблений підхід для використання в розподілених системах БПЛА, що оперують в агресивному інформаційному середовищі.

Висновки до розділу 4

У четвертому розділі наведено результати експериментального дослідження розроблених методів динамічної оцінки довіри та вибору лідера в розподілених системах безпілотних літальних апаратів. Комплексне моделювання, проведене на базі 20 унікальних сценаріїв, що відрізнялися складністю та протяжністю маршрутів, а також інтенсивністю атак (ін'єкцій хибних даних), дозволило підтвердити адекватність запропонованих моделей та сформулювати наступні висновки:

- Підтверджено ефективність нейромережевого методу динамічного оцінювання довіри на локальному рівні. Порівняльний аналіз із класичним детермінованим підходом (*Trust-WCA*) продемонстрував, що використання рекурентної нейронної мережі (*LSTM*) для динамічної адаптації порогу прийняття рішень забезпечує стабільне покращення якості класифікації об'єктів. Зокрема, точність локальних рішень (*Local Precision*) зросла в середньому на 4,3%, а повнота виявлення валідних цілей (*Local Recall*) – на 7,8%. Встановлено, що ключовим фактором результативності є здатність системи варіювати поріг довіри (θ) у діапазоні $[0,46; 0,92]$ залежно від контексту, що забезпечує гнучкий баланс між чутливістю та безпекою.
- Доведено підвищення функціональної стабільності виконання колективної місії рою. Інтеграція адаптивних локальних оцінок у процес глобального узгодження (консенсусу) забезпечила зростання середньої точності фізичного відвідування цілей (*Visit Precision*) на 8% (до рівня 0,81). При цьому частка непродуктивних витрат ресурсів рою, пов'язаних із рухом до хибних цілей («пасток»), знизилася на 8% (в абсолютному вимірі). Це підтверджує, що запропонований метод дозволяє ефективніше фільтрувати сфальсифіковані дані на рівні прийняття колективних рішень, зберігаючи при цьому достатній рівень покриття цільової області (приріст *F1-Score* склав 5.9%).
- Удосконалено метод вибору лідера рою, ефективність якого частково висвітлена у праці [23]. За результатами ретроспективного аналізу

польотних даних (*Trace-Driven Simulation*) встановлено, що використання нелінійної вагової функції з динамічним експоненціальним параметром ($Trust^{2\theta}$) забезпечує комплексну перевагу над класичними підходами за сукупністю критеріїв. В умовах налаштування на граничну чутливість розроблений метод продемонстрував відносне зниження частоти обрання скомпрометованих лідерів на 11,6% (з 6,0% до 5,3%) при одночасному покращенні топологічних характеристик: коефіцієнт ефективності мережі зріс на 0,8% (до 0,943), а середні втрати топології (*Topology Loss*) зменшилися на 17,4% (з 0,0344 до 0,0284). Таке поєднання показників свідчить про стабільність роботи алгоритму та його здатність уникати критичних загроз без погіршення якості обслуговування (*QoS*).

- Підтверджено практичну застосовність розробленого програмного забезпечення. Аналіз споживання обчислювальних ресурсів показав, що модуль інференсу нейронної мережі та розрахунку довіри створює мінімальне додаткове навантаження на систему порівняно з базовими процесами навігації та симуляції фізики. Низька обчислювальна складність алгоритмів дозволяє рекомендувати їх до впровадження в бортові системи керування БПЛА (класу *Edge Computing*), що функціонують в умовах обмеженого енергопостачання та реального часу.

Таким чином, результати експериментів повністю підтверджують наукові положення дисертаційної роботи: запропонований комплекс методів забезпечує підвищення стійкості розподілених систем до кібератак, гарантуючи безпеку управління та ефективність виконання місій у динамічному середовищі.

ВИСНОВКИ

У дисертаційній роботі виконано теоретичне обґрунтування та одержано нове вирішення науково-прикладної задачі підвищення ефективності та безпеки функціонування розподілених систем (зокрема, роїв безпілотних літальних апаратів) в умовах інформаційної протидії та візантійських відмов. З цією метою розроблено новий комплексний підхід до адаптивного керування розподіленою системою, який базується на інтеграції методів глибокого навчання, алгоритмів досягнення консенсусу та зваженої кластеризації. Розроблений підхід складається з математичної моделі системи та загроз, методу узгодження рішень із динамічною довірою, вдосконаленого методу вибору лідера та комплексу програмних засобів їх реалізації в середовищі робототехнічних систем.

Основні наукові та практичні результати дисертаційного дослідження полягають у наступному:

1. Досліджено існуючі методи керування розподіленими системами та виявлено критичне протиріччя між вимогами до оперативності прийняття рішень у децентралізованих розподілених системах та необхідністю захисту від інтелектуальних атак. Встановлено, що класичні детерміновані алгоритми (зокрема алгоритми зваженої кластеризації на основі довіри) через використання статичних моделей оцінювання є неефективними проти динамічних атак, а блокчейн-орієнтовані методи вносять неприпустимі часові затримки. Обґрунтовано необхідність розробки адаптивної системи, яка здатна використовувати рекурентні нейронні мережі для формування динамічного коефіцієнта довіри в реальному часі, що дозволяє підлаштовувати чутливість системи під зміни тактичної обстановки.

2. Розроблено та теоретично обґрунтовано математичну модель оцінювання довіри в розподілених системах, яка, на відміну від існуючих адитивних моделей, формалізує простір станів агентів та модель загроз семантичної маніпуляції, враховуючи нелінійні часові залежності поведінки. Це дозволяє виявляти приховані аномалії та стратегічні атаки (*on-off attacks*), які ігноруються класичними статистичними методами, забезпечуючи прогностну здатність системи

безпеки без суттєвого збільшення обчислювальної складності.

3. Розроблено метод динамічної оцінки довіри, ключовою особливістю якого є застосування рекурентної нейронної мережі (*LSTM*) з механізмом онлайн-донавчання для розрахунку адаптивного порогу прийняття рішень (θ). На відміну від методів із фіксованим порогом, запропонований метод автоматично підлаштовує чутливість системи під рівень невизначеності середовища (у діапазоні від 0,46 до 0,92). Це дозволило на локальному рівні підвищити точність класифікації вхідних даних на 4,3% та повноту виявлення валідних цілей на 7,8%, забезпечуючи баланс між безпекою та ефективністю розвідки.

4. Удосконалено метод вибору лідера (*Cluster Head*) на основі модифікованого алгоритму зваженої кластеризації [24]. Новизна методу полягає в інтеграції динамічного коефіцієнта довіри як змінної експоненти ($Trust^{2\theta}$) у функцію корисності, що діє як адаптивний мультиплікативний фільтр. На відміну від традиційних моделей з фіксованими вагами, такий підхід дозволяє реалізувати механізм «м'якої ізоляції» підозрілих вузлів. За результатами ретроспективного моделювання досягнуто відносне зниження частоти перехоплення керування зловмисниками на 11,6% при одночасному покращенні топологічної ефективності мережі на 0,8% та зменшенні втрат топології на 17,4%.

5. Розроблено комплексний програмний полігон для моделювання та керування розподіленою системою агентів (на прикладі рою безпілотних літальних апаратів як репрезентативного варіанту таких систем), реалізований на базі технологічного стеку *ROS 2* та *Gazebo* з використанням гібридної архітектури (*C++* для низькорівневого керування, *Python* для логіки довіри). Створено спеціалізований інструментальний засіб для проведення ретроспективного моделювання на основі збережених польотних даних, а також підсистему збору метрик у реальному часі. Архітектура програмного забезпечення оптимізована для використання на бортових обчислювачах класу *Edge Computing*, що підтверджено низькими показниками споживання системних ресурсів.

6. Експериментальне дослідження довело ефективність запропонованого комплексу методів на 20 унікальних сценаріях польотних місій зі змінною

топологією та інтенсивністю атак. За рахунок використання розроблених рішень середня точність виконання місії (*Visit Precision*) зросла на 8% (до рівня 0,81), а частка непродуктивних витрат ресурсів на відвідування хибних цілей зменшилася на 8% (в абсолютному вимірі) порівняно з класичним алгоритмом *Trust-WCA*. Загальна функціональна стабільність системи, оцінена за критерієм *F1-Score*, підвищилася на 5,9%, що підтверджує практичну цінність отриманих результатів для створення захищених автономних систем.

Отримані наукові та практичні результати формують підґрунтя для подальшого розвитку методів безпечного керування розподіленими системами. Зокрема, перспективними напрямками для подальших наукових досліджень можуть слугувати поглиблений теоретичний аналіз та математичне моделювання поведінки розроблених алгоритмів у нелінійних стохастичних мультиагентних системах, залучення методів математичної статистики для оцінки довірчих інтервалів при масштабуванні розподілених систем, а також проведення розширеного аналізу чутливості гіперпараметрів нейромережевих моделей.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- [1] H. Kopetz i W. Steiner, *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-031-11992-7.
- [2] X. Xia, S. M. M. Fattah, i M. A. Babar, «A Survey on UAV-Enabled Edge Computing: Resource Management Perspective», *ACM Comput. Surv.*, т. 56, вип. 3, С. 1—36, берез. 2024, doi: 10.1145/3626566.
- [3] S. Yang *та ін.*, «Hybrid ant colony-based inter-cluster routing protocol for FANET», *Sci. Rep.*, т. 14, вип. 1, С. 15632, лип. 2024, doi: 10.1038/s41598-024-64454-1.
- [4] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, i W. Woodall, «Robot Operating System 2: Design, architecture, and uses in the wild», *Sci. Robot.*, т. 7, вип. 66, С. eabm6074, трав. 2022, doi: 10.1126/scirobotics.abm6074.
- [5] K.-Y. Tsao, T. Girdler, i V. G. Vassilakis, «A survey of cyber security threats and solutions for UAV communications and flying ad-hoc networks», *Ad Hoc Netw.*, т. 133, С. 102894, серп. 2022, doi: 10.1016/j.adhoc.2022.102894.
- [6] J. Sharma i P. S. Mehra, «Secure communication in IOT-based UAV networks: A systematic survey», *Internet Things*, т. 23, С. 100883, жовт. 2023, doi: 10.1016/j.iot.2023.100883.
- [7] T. S. Delwar *та ін.*, «The Intersection of Machine Learning and Wireless Sensor Network Security for Cyber-Attack Detection: A Detailed Analysis», *Sensors*, т. 24, вип. 19, С. 6377, жовт. 2024, doi: 10.3390/s24196377.
- [8] Y. Zhang, Z. Hu, Z. Wang, X. Wen, i Z. Lu, «Survivability Analysis of Unmanned Aerial Vehicle Network based on Dynamic Weighted Clustering Algorithm with Dual Cluster Heads», *Electronics*, т. 12, вип. 7, С. 1743, квіт. 2023, doi: 10.3390/electronics12071743.
- [9] J. Wang, J. Wang, Z. Tong, Z. Jiao, M. Zhang, i C. Jiang, «ACBFT: Adaptive Chained Byzantine Fault-Tolerant Consensus Protocol for UAV Ad Hoc Networks», *IEEE Trans. Veh. Technol.*, т. 74, вип. 7, С. 11324—11336, лип. 2025, doi: 10.1109/TVT.2025.3548281.

- [10] L. Kong, B. Chen, i F. Hu, «LAP-BFT: Lightweight Asynchronous Provable Byzantine Fault-Tolerant Consensus Mechanism for UAV Network», *Drones*, т. 6, вип. 8, С. 187, лип. 2022, doi: 10.3390/drones6080187.
- [11] A. Volokyta i M. Melenchukov, «Analysis of defense and attack methods in distributed systems», представлений у The International Conference on Security, Kyiv, Ukraine, чер. 2024. [URL]. Доступний у: <https://icsfti-proc.kpi.ua/issue/view/16681>
- [12] Y. Liu, J. Wang, Z. Yan, Z. Wan, i R. Jäntti, «A Survey on Blockchain-Based Trust Management for Internet of Things», *IEEE Internet Things J.*, т. 10, вип. 7, С. 5898—5922, квіт. 2023, doi: 10.1109/IJOT.2023.3237893.
- [13] J. Yuan, J. Peng, Q. Yan, G. He, H. Xiang, i Z. Liu, «Deep Reinforcement Learning-Based Energy Consumption Optimization for Peer-to-Peer (P2P) Communication in Wireless Sensor Networks», *Sensors*, т. 24, вип. 5, С. 1632, безз. 2024, doi: 10.3390/s24051632.
- [14] Y. L. Sun, Z. Han, W. Yu, i K. J. R. Liu, «A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks», у *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, Barcelona: IEEE, квіт. 2006, С. 1—13. doi: 10.1109/INFOCOM.2006.154.
- [15] F. G. Mármol i G. M. Pérez, «Security threats scenarios in trust and reputation models for distributed systems», *Comput. Secur.*, т. 28, вип. 7, С. 545—556, жовт. 2009, doi: 10.1016/j.cose.2009.05.005.
- [16] N. H. Vaidya, L. Tseng, i G. Liang, «Iterative approximate byzantine consensus in arbitrary directed graphs», у *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, Madeira Portugal: ACM, лип. 2012, С. 365—374. doi: 10.1145/2332432.2332505.
- [17] A. R. Khouas, M. R. Bouadjenek, H. Hacid, i S. Aryal, «Training Machine Learning models at the Edge: A Survey», 2024. , *arXiv*. doi: 10.48550/ARXIV.2403.02619.
- [18] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, i J. Stainer, «Byzantine-Tolerant Machine Learning», 2017. , *arXiv*. doi: 10.48550/ARXIV.1703.02757.

- [19] F. S. Gharehchopogh, «A Survey and Taxonomy of Leader Election Algorithms in Distributed Systems», *Indian J. Sci. Technol.*, т. 7, вип. 6, С. 815—830, чер. 2014, doi: 10.17485/ijst/2014/v7i6.14.
- [20] P. Basu, N. Khan, i T. D. C. Little, «A mobility based metric for clustering in mobile ad hoc networks», у *Proceedings 21st International Conference on Distributed Computing Systems Workshops*, Mesa, AZ, USA: IEEE Comput. Soc, 2001, С. 413—418. doi: 10.1109/CDCS.2001.918738.
- [21] National Technical University of Ukraine “Ihor Sikorskyi Kyiv Polytechnic Institute”, A. Volokyta, M. Melenchukov, i National Technical University of Ukraine “Kyiv Polytechnic Institute named after Igor Sikorsky”, «NEURAL NETWORKS IN DETECTING ATTACKS ON DISTRIBUTED SYSTEMS», *Tech. Sci. Technol.*, вип. 1(35), С. 135—145, 2024, doi: 10.25140/2411-5363-2024-1(35)-135-145.
- [22] А. Волокита і М. Меленчуков, «ДОСЛІДЖЕННЯ МОДЕЛЕЙ ВИЯВЛЕННЯ АТАК НА РОЗПОДІЛЕНІ СИСТЕМИ ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ», *Meas. Comput. DEVICES Technol. Process.*, вип. 3, С. 224—229, серп. 2024, doi: 10.31891/2219-9365-2024-79-29.
- [23] А. Волокита і М. Меленчуков, «МЕТОД ВИБОРУ ЛІДЕРА ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ КОМУНІКАЦІЇ В РОЇ БПЛА», *Meas. Comput. DEVICES Technol. Process.*, вип. 3, С. 409—414, серп. 2025, doi: 10.31891/2219-9365-2025-83-50.
- [24] А. Волокита і М. Меленчуков, «Адаптивний метод вибору лідера в мобільних розподілених системах на основі інтеграції динамічної довіри», *Проблеми Інформатизації Та Управління*, т. 1, вип. 85, С. 35—41, квіт. 2026, doi: 10.18372/2073-4751.85.21091.
- [25] А. Волокита і М. Меленчуков, «Dynamic Trust Assessment Method Using Modified Hysteresis in UAV Swarms», *Telecommun. Inf. Technol.*, т. 90, вип. 1, С. 218—224, берез. 2026, doi: 10.31673/2412-4338.2026.019021.
- [26] М. Melenchukov, «Improving safety and efficiency of distributed systems: a predictive navigation model for uav swarms», Kyiv, Ukraine: The International

- Conference on Security, чер. 2025. [URL]. Доступний у: <https://icsfti-proc.kpi.ua/article/view/354612>
- [27] M. van Steen і A. S. Tanenbaum, *Distributed systems*, Third edition, Version 3.01. Erscheinungsort nicht ermittelbar: Maarten van Steen, 2017.
- [28] Cisco, «Cisco Annual Internet Report (2018–2023) White Paper», *Cisco Syst.*, 2020, [URL]. Доступний у: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>.
- [29] L. Gupta, R. Jain, і G. Vaszkun, «Survey of Important Issues in UAV Communication Networks», *IEEE Commun. Surv. Tutor.*, т. 18, вип. 2, С. 1123—1152, 2016, doi: 10.1109/COMST.2015.2495297.
- [30] A. Chriki, H. Touati, H. Snoussi, і F. Kamoun, «FANET: Communication, mobility models and security issues», *Comput. Netw.*, т. 163, С. 106877, листоп. 2019, doi: 10.1016/j.comnet.2019.106877.
- [31] Z. Yuan, J. Jin, L. Sun, K.-W. Chin, і G.-M. Muntean, «Ultra-Reliable IoT Communications with UAVs: A Swarm Use Case», *IEEE Commun. Mag.*, т. 56, вип. 12, С. 90—96, груд. 2018, doi: 10.1109/MCOM.2018.1800161.
- [32] E. A. Brewer, «Towards robust distributed systems (abstract)», у *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, Portland Oregon USA: ACM, лип. 2000, С. 7. doi: 10.1145/343477.343502.
- [33] İ. Bekmezci, O. K. Sahingoz, і Ş. Temel, «Flying Ad-Hoc Networks (FANETs): A survey», *Ad Hoc Netw.*, т. 11, вип. 3, С. 1254—1270, трав. 2013, doi: 10.1016/j.adhoc.2012.12.004.
- [34] L. Lamport, R. Shostak, і M. Pease, «The Byzantine Generals Problem», *ACM Trans. Program. Lang. Syst.*, т. 4, вип. 3, С. 382—401, лип. 1982, doi: 10.1145/357172.357176.
- [35] Garcia-Molina, «Elections in a Distributed Computing System», *IEEE Trans. Comput.*, т. C—31, вип. 1, С. 48—59, січ. 1982, doi: 10.1109/TC.1982.1675885.
- [36] P.-Y. Kong, «A Survey of Cyberattack Countermeasures for Unmanned Aerial Vehicles», *IEEE Access*, т. 9, С. 148244—148263, 2021, doi:

10.1109/ACCESS.2021.3124996.

- [37] S. Marti, T. J. Giuli, K. Lai, i M. Baker, «Mitigating routing misbehavior in mobile ad hoc networks», y *Proceedings of the 6th annual international conference on Mobile computing and networking*, Boston Massachusetts USA: ACM, серп. 2000, С. 255—265. doi: 10.1145/345910.345955.
- [38] K. Hoffman, D. Zage, i C. Nita-Rotaru, «A survey of attack and defense techniques for reputation systems», *ACM Comput. Surv.*, т. 42, вип. 1, С. 1—31, груд. 2009, doi: 10.1145/1592451.1592452.
- [39] S. Mohsen Hassan, M. M. Mohamad, i F. Binti Muchtar, «Enhancing MANET Security Through Long Short-Term Memory-Based Trust Prediction in Location-Aided Routing Protocols», *IEEE Access*, т. 13, С. 120142—120168, 2025, doi: 10.1109/ACCESS.2025.3572619.
- [40] L. Eschenauer, V. D. Gligor, i J. Baras, «On Trust Establishment in Mobile Ad-Hoc Networks», y *Security Protocols*, т. 2845, B. Christianson, B. Crispo, J. A. Malcolm, i M. Roe, Ред., y *Lecture Notes in Computer Science*, vol. 2845. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, С. 47—66. doi: 10.1007/978-3-540-39871-4_6.
- [41] J.-H. Cho, A. Swami, i I.-R. Chen, «A Survey on Trust Management for Mobile Ad Hoc Networks», *IEEE Commun. Surv. Tutor.*, т. 13, вип. 4, С. 562—583, 2011, doi: 10.1109/SURV.2011.092110.00088.
- [42] K. Govindan i P. Mohapatra, «Trust Computations and Trust Dynamics in Mobile Adhoc Networks: A Survey», *IEEE Commun. Surv. Tutor.*, т. 14, вип. 2, С. 279—298, 2012, doi: 10.1109/SURV.2011.042711.00083.
- [43] L. Zhang, X. Feng, B. Yang, i Z. Lei, «Threat-Adaptive Cooperative Control for Heterogeneous UAV Swarms via Dynamic Graph Neural Networks», y *Proceedings of the 2025 9th International Conference on Computer Science and Artificial Intelligence*, Beijing China: ACM, груд. 2025, С. 690—699. doi: 10.1145/3788149.3788175.
- [44] M. U. Afzal i A. A. Abdellatif, «Energy-Efficient Secure Federated Learning for UAV Swarms», y *2024 7th International Conference on Energy Conservation and*

- Efficiency (ICECE)*, Lahore, Pakistan: IEEE, без. 2024, С. 1—5. doi: 10.1109/ICECE61222.2024.10505278.
- [45] M. Deutel, M. Mallah, J. Wissing, i S. Scheele, «Recent Trends in Edge AI: Efficient Design, Training and Deployment of Machine Learning Models», y *Charting the Intelligence Frontiers – Edge AI Systems Nexus*, 1e вид., New York: River Publishers, 2025, С. 181—220. doi: 10.1201/9788743808862-9.
- [46] H. Fatemidokht, M. Kuchaki Rafsanjani, V. Arya, i B. B. Gupta, «Routing protocols in FANETs: A comprehensive survey and taxonomy», *J. Syst. Archit.*, т. 175, С. 103774, чеп. 2026, doi: 10.1016/j.sysarc.2026.103774.
- [47] M. Krichen i A. Mihoub, «Long Short-Term Memory Networks: A Comprehensive Survey», *AI*, т. 6, вып. 9, С. 215, верес. 2025, doi: 10.3390/ai6090215.
- [48] M. J. Fischer, N. A. Lynch, i M. S. Paterson, «Impossibility of distributed consensus with one faulty process», *J. ACM*, т. 32, вып. 2, С. 374—382, квіт. 1985, doi: 10.1145/3149.214121.
- [49] C. Fan, S. Ghaemi, H. Khazaei, i P. Musilek, «Performance Evaluation of Blockchain Systems: A Systematic Survey», *IEEE Access*, т. 8, С. 126927—126950, 2020, doi: 10.1109/ACCESS.2020.3006078.
- [50] M. Salimitari, M. Chatterjee, i Y. P. Fallah, «A survey on consensus methods in blockchain for resource-constrained IoT networks», *Internet Things*, т. 11, С. 100212, верес. 2020, doi: 10.1016/j.iot.2020.100212.
- [51] H. J. LeBlanc, H. Zhang, X. Koutsoukos, i S. Sundaram, «Resilient Asymptotic Consensus in Robust Networks», *IEEE J. Sel. Areas Commun.*, т. 31, вып. 4, С. 766—781, квіт. 2013, doi: 10.1109/JSAC.2013.130413.
- [52] S. Vasudevan, J. Kurose, i D. Towsley, «Design and analysis of a leader election algorithm for mobile ad hoc networks», y *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, Berlin, Germany: IEEE, 2004, С. 350—360. doi: 10.1109/ICNP.2004.1348124.
- [53] M. Pease, R. Shostak, i L. Lamport, «Reaching Agreement in the Presence of Faults», *J. ACM*, т. 27, вып. 2, С. 228—234, квіт. 1980, doi: 10.1145/322186.322188.

- [54] A. Clement, E. Wong, L. Alvisi, M. Dahlin, i M. Marchetti, «Making Byzantine fault tolerant systems tolerate Byzantine faults», у *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, The USENIX Association, 2009, С. 153—168. [URL]. Доступний у: <https://www.usenix.org/conference/nsdi-09/making-byzantine-fault-tolerant-systems-tolerate-byzantine-faults>
- [55] F. B. Schneider, «Implementing fault-tolerant services using the state machine approach: a tutorial», *ACM Comput. Surv.*, т. 22, вип. 4, С. 299—319, груд. 1990, doi: 10.1145/98163.98167.
- [56] X. Défago, A. Schiper, i P. Urbán, «Total order broadcast and multicast algorithms: Taxonomy and survey», *ACM Comput. Surv.*, т. 36, вип. 4, С. 372—421, груд. 2004, doi: 10.1145/1041680.1041682.
- [57] M. P. Herlihy i J. M. Wing, «Linearizability: a correctness condition for concurrent objects», *ACM Trans. Program. Lang. Syst.*, т. 12, вип. 3, С. 463—492, лип. 1990, doi: 10.1145/78969.78972.
- [58] W. B. Heinzelman, A. P. Chandrakasan, i H. Balakrishnan, «An application-specific protocol architecture for wireless microsensor networks», *IEEE Trans. Wirel. Commun.*, т. 1, вип. 4, С. 660—670, жовт. 2002, doi: 10.1109/TWC.2002.804190.
- [59] X. Luo, M. Dong, i Y. Huang, «On distributed fault-tolerant detection in wireless sensor networks», *IEEE Trans. Comput.*, т. 55, вип. 1, С. 58—70, січ. 2006, doi: 10.1109/TC.2006.13.
- [60] G. Ricart i A. K. Agrawala, «An optimal algorithm for mutual exclusion in computer networks», *Commun. ACM*, т. 24, вип. 1, С. 9—17, січ. 1981, doi: 10.1145/358527.358537.
- [61] T. D. Chandra i S. Toueg, «Unreliable failure detectors for reliable distributed systems», *J. ACM*, т. 43, вип. 2, С. 225—267, берез. 1996, doi: 10.1145/226643.226647.
- [62] M. Chatterjee, S. K. Das, i D. Turgut, «WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks», *Clust. Comput.*, т. 5, вип. 2, С. 193—204, квіт. 2002, doi: 10.1023/A:1013941929408.

- [63] B. Wu, J. Chen, J. Wu, i M. Cardei, «A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks», y *Wireless Network Security*, Y. Xiao, X. S. Shen, i D.-Z. Du, Ред., y Signals and Communication Technology. , Boston, MA: Springer US, 2007, С. 103—135. doi: 10.1007/978-0-387-33112-6_5.
- [64] R. H. Jhaveri, N. M. Patel, i D. C. Jinwala, «A Composite Trust Model for Secure Routing in Mobile Ad-Hoc Networks», y *Ad Hoc Networks*, J. H. Ortiz i A. P. D. L. Cruz, Ред., InTech, 2017. doi: 10.5772/66519.
- [65] A. Goldsmith, *Wireless communications*. Cambridge ; New York: Cambridge University Press, 2005.
- [66] C. Karlof i D. Wagner, «Secure routing in wireless sensor networks: attacks and countermeasures», *Ad Hoc Netw.*, т. 1, вип. 2—3, С. 293—315, верес. 2003, doi: 10.1016/S1570-8705(03)00008-8.
- [67] National Institute of Standards and Technology (US), «Digital signature standard (DSS)», National Institute of Standards and Technology (U.S.), Washington, D.C., error: 186-4, 2013. doi: 10.6028/NIST.FIPS.186-4.
- [68] A. D. Wood i J. A. Stankovic, «Denial of service in sensor networks», *Computer*, т. 35, вип. 10, С. 54—62, жовт. 2002, doi: 10.1109/MC.2002.1039518.
- [69] W. Dargie i C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. y Wiley series on wireless communications and mobile computing. Chichester, West Sussex, U.K. Hoboken, NJ: Wiley, 2010.
- [70] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, i G. Das, «Everything You Wanted to Know About the Blockchain: Its Promise, Components, Processes, and Problems», *IEEE Consum. Electron. Mag.*, т. 7, вип. 4, С. 6—14, лип. 2018, doi: 10.1109/MCE.2018.2816299.
- [71] W. Wang *та ін.*, «A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks», *IEEE Access*, т. 7, С. 22328—22370, 2019, doi: 10.1109/ACCESS.2019.2896108.
- [72] C. Gorenflo, S. Lee, L. Golab, i S. Keshav, «FastFabric: Scaling hyperledger fabric to 20 000 transactions per second», *Int. J. Netw. Manag.*, т. 30, вип. 5, С. e2099, верес. 2020, doi: 10.1002/nem.2099.

- [73] A. Dorri, S. S. Kanhere, R. Jurdak, i P. Gauravaram, «Blockchain for IoT security and privacy: The case study of a smart home», у *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, HI: IEEE, без. 2017, С. 618—623. doi: 10.1109/PERCOMW.2017.7917634.
- [74] S. M. Alrubei, E. A. Ball, J. M. Rigelsford, i Callum. A. Willis, «Latency and Performance Analyses of Real-World Wireless IoT-Blockchain Application», *IEEE Sens. J.*, т. 20, вип. 13, С. 7372—7383, лип. 2020, doi: 10.1109/JSEN.2020.2979031.
- [75] M. Dorigo, M. Birattari, i T. Stutzle, «Ant colony optimization», *IEEE Comput. Intell. Mag.*, т. 1, вип. 4, С. 28—39, листоп. 2006, doi: 10.1109/MCI.2006.329691.
- [76] R. Chauhan i R. Kumar, «BIO Inspired Clustering and Routing for Wireless Sensor Networks», *Int. J. Comput. Appl.*, т. 154, вип. 8, С. 22—27, листоп. 2016, doi: 10.5120/ijca2016912184.
- [77] D. Karaboga i B. Basturk, «A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm», *J. Glob. Optim.*, т. 39, вип. 3, С. 459—471, жовт. 2007, doi: 10.1007/s10898-007-9149-х.
- [78] R. V. Kulkarni i G. K. Venayagamoorthy, «Particle Swarm Optimization in Wireless-Sensor Networks: A Brief Survey», *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, т. 41, вип. 2, С. 262—267, без. 2011, doi: 10.1109/TSMCC.2010.2054080.
- [79] B. Singh i D. K. Lobiyal, «A novel energy-aware cluster head selection based on particle swarm optimization for wireless sensor networks», *Hum.-Centric Comput. Inf. Sci.*, т. 2, вип. 1, С. 13, груд. 2012, doi: 10.1186/2192-1962-2-13.
- [80] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, i M. Guizani, «A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security», *IEEE Commun. Surv. Tutor.*, т. 22, вип. 3, С. 1646—1685, 2020, doi: 10.1109/COMST.2020.2988293.
- [81] S. Hayat, E. Yanmaz, i R. Muzaffar, «Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint», *IEEE Commun.*

- Surv. Tutor.*, т. 18, вип. 4, С. 2624—2661, 2016, doi: 10.1109/COMST.2016.2560343.
- [82] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneş, і M. B. Yagoubi, «A survey on position-based routing protocols for Flying Ad hoc Networks (FANETs)», *Veh. Commun.*, т. 10, С. 29—56, жовт. 2017, doi: 10.1016/j.vehcom.2017.10.003.
- [83] M. A. Khan, A. Safi, I. M. Qureshi, і I. U. Khan, «Flying ad-hoc networks (FANETs): A review of communication architectures, and routing protocols», у *2017 First International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT)*, Karachi: IEEE, листоп. 2017, С. 1—9. doi: 10.1109/INTELLECT.2017.8277614.
- [84] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, і M. H. Rehmani, «Applications of Blockchains in the Internet of Things: A Comprehensive Survey», *IEEE Commun. Surv. Tutor.*, т. 21, вип. 2, С. 1676—1717, 2019, doi: 10.1109/COMST.2018.2886932.
- [85] M. Gerla і J. Tzu-Chieh Tsai, «Multicluster, mobile, multimedia radio network», *Wirel. Netw.*, т. 1, вип. 3, С. 255—265, верес. 1995, doi: 10.1007/BF01200845.
- [86] V. Paxson, M. Allman, J. Chu, і M. Sargent, «Computing TCP's Retransmission Timer», RFC Editor, RFC6298, чер. 2011. doi: 10.17487/rfc6298.
- [87] R. J. Hyndman і G. Athanasopoulos, *Forecasting: principles and practice*, Third edition. Melbourne, Australia: OTexts, 2021.
- [88] P. Michiardi і R. Molva, «Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks», у *Advanced Communications and Multimedia Security*, т. 100, В. Jerman-Blažič і T. Klobučar, Ред., у IFIP Advances in Information and Communication Technology, vol. 100. , Boston, MA: Springer US, 2002, С. 107—121. doi: 10.1007/978-0-387-35612-9_9.
- [89] N. Koenig і A. Howard, «Design and use paradigms for gazebo, an open-source multi-robot simulator», у *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Sendai, Japan: IEEE, 2004, С. 2149—2154. doi: 10.1109/IROS.2004.1389727.
- [90] A. D. Kshemkalyani і M. Singhal, *Distributed computing: principles, algorithms,*

and systems, 1. paperback edition(with corrections). Cambridge New York Melbourne Madrid Cape Town Singapore: Cambridge University Press, 2011.

- [91] G. F. Coulouris, Ред., *Distributed systems: concepts and design*, 5. ed. y International computer science series. Boston Munich: Addison-Wesley, 2012.
- [92] D. Ongaro i J. Ousterhout, «In search of an understandable consensus algorithm», y *Proceedings of the 2014 USENIX Annual Technical Conference*, USENIX Association, 2014, C. 305—319.

ДОДАТОК А

Частина програмного коду

```

class ThetaNNAdapter:

    DEFAULT_INPUT_DIM = 5

    def __init__(
        self,
        drone_name: str,
        model_dir: str,
        delta_scale: float,
        theta_min: float,
        theta_max: float,
        online: bool,
        online_lr: float,
        buffer_size: int,
        update_every: int,
        batch_size: int,
        logger,
        seq_len: int = 8,
        input_dim: int | None = None,
        unres_lambda: float = 0.6,
        uncert_min_weight: float = 0.0,
        up_scale: float = 1.0,
        p_center_beta: float = 0.997
    ):
        self.enabled = False
        self.model = None
        self.drone_name = drone_name
        self.model_path = f»{os.path.dirname(model_dir)}/{self.drone_name}_model.h5»
        self.delta_scale = float(delta_scale)
        self.theta_min = float(theta_min)
        self.theta_max = float(theta_max)
        self.online = bool(online)
        self.buffer_size = int(buffer_size)
        self.update_every = int(update_every)
        self.batch_size = int(batch_size)
        self.logger = logger
        self.input_dim = int(max(1, input_dim if input_dim is not None else
self.DEFAULT_INPUT_DIM))
        self._Xseq: list[np.ndarray] = []
        self._Y: list[float] = []
        self._W: list[float] = []

```

```

self._steps = 0
self.seq_len = int(max(1, seq_len))
self.unres_lambda = float(unres_lambda)
self.uncert_min_weight = float(uncert_min_weight)
self._tracks: dict[tuple[float, float, float] | str, list[list[float]]] = {}
self._p_center = 0.5
self._p_center_beta = float(p_center_beta)
self._up_scale = float(up_scale)
self._train_queue: queue.Queue | None = None
self._trainer_thread: threading.Thread | None = None
self._stop_event = threading.Event()

# Try load; if missing → build a fresh tiny model
try:
    self.model = tf.keras.models.load_model(self.model_path)
    # Force rebuild if a non-LSTM (2D input) model was loaded
    if len(self.model.inputs[0].shape) != 3:
        raise ValueError(«Loaded non-LSTM model; rebuilding.»)
except Exception:
    self.logger.warn(«Creating LSTM model»)
    inputs = tf.keras.Input(shape=(self.seq_len, self.input_dim))
    x = tf.keras.layers.Masking(mask_value=0.0)(inputs)
    x = tf.keras.layers.LSTM(16, return_sequences=False)(x)
    x = tf.keras.layers.Dense(8, activation=«relu»)(x)
    outputs = tf.keras.layers.Dense(1, activation=«sigmoid»)(x)
    self.model = tf.keras.Model(inputs, outputs)

# Compile / set LR
if not hasattr(self.model, «optimizer») or self.model.optimizer is None:
    self.model.compile(
        optimizer=tf.keras.optimizers.Adam(online_lr),
        loss=«binary_crossentropy»,
        metrics=[«AUC»],
    )
else:
    try:
        tf.keras.backend.set_value(self.model.optimizer.learning_rate, online_lr)
    except Exception:
        self.logger.warn(«TF Failed to set value»)
        pass

if self.online:
    self._train_queue = queue.Queue(maxsize=self.buffer_size)
    self._trainer_thread = threading.Thread(target=self._train_loop, daemon=True)
    self._trainer_thread.start()

```

```

self.logger.warn(«NN training worker started»)

self.enabled = True

def _push_step(self, key: tuple[float, float, float] | str | None, feats: list[float]) ->
None:
    if key is None:
        key = «__global__»
    if key not in self._tracks:
        self._tracks[key] = []
    vec = [float(f) for f in feats]
    if len(vec) != self.input_dim:
        if len(vec) < self.input_dim:
            vec.extend([0.0] * (self.input_dim - len(vec)))
        else:
            vec = vec[: self.input_dim]
    self._tracks[key].append(vec)
    if len(self._tracks[key]) > self.seq_len:
        overflow = len(self._tracks[key]) - self.seq_len
        if overflow > 0:
            del self._tracks[key][0:overflow]

def _pad_seq(self, seq: list[list[float]]) -> np.ndarray:
    T = self.seq_len
    out = np.zeros((T, self.input_dim), dtype=np.float32)
    sl = min(T, len(seq))
    if sl > 0:
        out[T - sl:T, :] = np.asarray(seq[-sl:], dtype=np.float32)
    return out

def _train_loop(self) -> None:
    while not self._stop_event.is_set():
        time.sleep(0.1)
        try:
            item = self._train_queue.get(timeout=0.1)
        except queue.Empty:
            continue
        if item is None:
            break
        Xb, Yb, Wb = item
        try:
            self.model.train_on_batch(Xb, Yb, sample_weight=Wb)
        except Exception as exc:
            self.logger.warn(f»NN train step failed: {exc}»)

```

```

def save(self, path: str | None = None) -> None:
    if not (self.enabled and self.model is not None):
        return
    self.close()
    p = path or self.model_path
    p = os.path.expanduser(p)
    os.makedirs(os.path.dirname(p) or «.», exist_ok=True)
    self.model.save(p)
    self.model_path = p

def close(self) -> None:
    if self._train_queue is None or self._trainer_thread is None:
        return
    self._stop_event.set()
    if self._trainer_thread.is_alive():
        try:
            self._train_queue.put_nowait(None)
        except queue.Full:
            pass
    self._trainer_thread.join(timeout=2.0)

def __del__(self):
    try:
        self.close()
    except Exception:
        pass

@staticmethod
def from_cfg(
    drone_name: str,
    cfg,
    logger,
    input_dim: int | None = None,
) -> ThetaNNAdapter | None:
    if not cfg.enabled:
        return None
    return ThetaNNAdapter(
        drone_name=drone_name,
        model_dir=cfg.model_path,
        delta_scale=float(cfg.delta_scale),
        theta_min=float(cfg.theta_min),
        theta_max=float(cfg.theta_max),
        online=bool(cfg.online),
        online_lr=float(cfg.online_lr),
        buffer_size=int(cfg.buffer_size),

```

```

    update_every=int(cfg.update_every),
    batch_size=int(cfg.batch_size),
    logger=logger,
    seq_len=int(cfg.seq_len),
    input_dim=input_dim,
    unres_lambda=float(cfg.unresolved_boost),
    uncert_min_weight=float(cfg.uncertainty.min_weight),
    up_scale=float(cfg.up_scale),
    p_center_beta=float(cfg.p_center_beta),
)

```

```

def adapt_theta(self, theta0: float, feats: list[float], key=None) -> float:
    theta = max(self.theta_min, min(self.theta_max, theta0))
    if not self.enabled or self.model is None:
        return theta
    try:
        self._push_step(key, feats)
        k = key if key is not None else «__global__»
        seq = self._pad_seq(self._tracks[k] if k in self._tracks else [])
        x = tf.convert_to_tensor([seq], dtype=tf.float32)
        p_commit = float(self.model(x, training=False).numpy().squeeze())
    except Exception:
        return theta
    p_commit = min(1.0, max(0.0, p_commit))
    beta = self._p_center_beta
    self._p_center = beta * self._p_center + (1.0 - beta) * p_commit
    p_adj = p_commit - (self._p_center - 0.5)
    p_adj = min(1.0, max(0.0, p_adj))
    self.logger.warn(f»P_commit: {p_commit:.3f} (center {self._p_center:.3f})»)
    delta_unit = (p_adj - 0.5) * 2.0
    if delta_unit < 0:
        delta_unit *= self._up_scale
    theta_new = theta - self._delta_scale * delta_unit
    return max(self.theta_min, min(self.theta_max, theta_new))

def predict_p(self, feats: list[float], key=None) -> float:
    if not (self.enabled and self.model is not None):
        return 0.5
    try:
        k = key if key is not None else «__global__»
        base = self._tracks.get(k, [])
        vec = [float(f) for f in feats]
        if len(vec) != self.input_dim:
            if len(vec) < self.input_dim:
                vec.extend([0.0] * (self.input_dim - len(vec)))

```

```

    else:
        vec = vec[: self.input_dim]
        tmp = (base + [vec])[-self.seq_len:]
        seq = self._pad_seq(tmp)
        x = tf.convert_to_tensor([seq], dtype=tf.float32)
        return float(self.model(x, training=False).numpy().squeeze())
except Exception:
    return 0.5

def observe(self, feats: list[float], y: int, w: float = 1.0, key=None) -> None:
    if not (self.enabled and self.model is not None and self.online):
        return
    self._push_step(key, feats)
    k = key if key is not None else «__global__»
    seq = self._pad_seq(self._tracks[k] if k in self._tracks else [])
    self._Xseq.append(seq)
    self._Y.append(float(y))
    self._W.append(float(w))
    if len(self._Xseq) > self.buffer_size:
        self._Xseq = self._Xseq[-self.buffer_size:]
        self._Y = self._Y[-self.buffer_size:]
        self._W = self._W[-self.buffer_size:]
    self._steps += 1
    if self._steps % self.update_every != 0:
        return
    import numpy as _np
    n = len(self._Xseq)
    if n == 0:
        return
    bsz = min(self.batch_size, n)
    Yarr = _np.array(self._Y, dtype=_np.float32)
    pos_idx = _np.where(Yarr == 1.0)[0]
    neg_idx = _np.where(Yarr == 0.0)[0]
    if len(pos_idx) and len(neg_idx):
        half = bsz // 2
        pi = _np.random.choice(pos_idx, size=min(half, len(pos_idx)),
replace=(len(pos_idx) < half))
        ni = _np.random.choice(neg_idx, size=min(bsz - len(pi), len(neg_idx)),
replace=(len(neg_idx) < (bsz - len(pi))))
        idx = _np.concatenate([pi, ni])
    else:
        idx = _np.random.choice(n, size=bsz, replace=False)
    Xb = _np.array([self._Xseq[i] for i in idx], dtype=_np.float32)
    Yb = _np.array([self._Y[i] for i in idx], dtype=_np.float32)
    w_pos = (len(neg_idx) / max(1, len(pos_idx))) if (len(pos_idx) and len(neg_idx))

```

```

else 1.0
    Wb_class = _np.where(Yb == 1.0, w_pos, 1.0).astype(_np.float32)
    Wb_obs = _np.array([self._W[i] for i in idx], dtype=_np.float32)
    Wb = (Wb_class * Wb_obs).astype(_np.float32)
    if self._train_queue is not None and not self._stop_event.is_set():
        try:
            self._train_queue.put_nowait((Xb, Yb, Wb))
        except queue.Full:
            self.logger.warn(«NN train queue full; dropping batch»)
    else:
        self.logger.warn(«NN Executing train on batch (sync)»)
        self.model.train_on_batch(Xb, Yb, sample_weight=Wb)

def observe_uncertain(self, key, feats: list[float], u: float, p: float) -> None:

    if not (self.enabled and self.model is not None and self.online):
        return
    w = max(0.0, min(1.0, self.unres_lambda * float(u) * float(1.0 - p)))
    w = max(w, self.uncert_min_weight * max(0.0, float(u)))
    if w <= 0.0:
        return
    self.observe(feats, y=1, w=w, key=key)

from rclpy.node import Node
from rclpy.qos import QoSProfile, ReliabilityPolicy, DurabilityPolicy
from rclpy.clock import Clock, ClockType

from drone_interfaces.msg import DroneShift
from drone_interfaces.msg import Proposal
from drone_interfaces.msg import Decision
from drone_interfaces.msg import SwarmMission
from drone_interfaces.msg import SensorReport

from .drone_mover import DroneMover
from .drone_database import DroneStorage
from .decision_core import DecisionCore
from .decision_core import DECISION_FLY_TO_MARK
from .decision_core import DECISION_FOLLOW_PATH

from geometry_msgs.msg import Pose
from geometry_msgs.msg import Point
from geometry_msgs.msg import Quaternion

```

```
from collections import defaultdict
```

```
import math
```

```
import time
```

```
class DroneController(Node):
```

```
    def __init__(self, drone_name: str, marks: list[list[float]]):
```

```
        super().__init__(f»drone_pilot_{drone_name}»)
```

```
        self.drone_name = drone_name
```

```
        self.PROPOSAL_TIMER_PERIOD_S = 2.0
```

```
        self.drones_in_swarm_count = 10
```

```
        self._current_pose: Pose = Pose()
```

```
        self._proposal_round_id: int = 0
```

```
        self.proposal_sync_delay_timer = None
```

```
        self.steady_clock = Clock(clock_type=ClockType.STEADY_TIME)
```

```
        self._action : Decision = Decision()
```

```
        self._committed_prev = None
```

```
        self._expected_marks = defaultdict(set)
```

```
        self._last_action_round = -1
```

```
        sub_qos = QoSProfile(
```

```
            depth=1,
```

```
            reliability=ReliabilityPolicy.RELIABLE,
```

```
            durability=DurabilityPolicy.TRANSIENT_LOCAL,
```

```
        )
```

```
        proposal_qos = QoSProfile(
```

```
            depth=200,
```

```
            reliability=ReliabilityPolicy.RELIABLE,
```

```
            durability=DurabilityPolicy.VOLATILE)
```

```
        mission_topic = «swarm_mission»
```

```
        pos_topic = f»/model/{drone_name}/pos»
```

```
        cmd_move_topic = f»/model/{drone_name}/cmd_move»
```

```
        proposal_topic = «/swarm_mission/proposals»
```

```
        decision_topic = «/swarm_mission/decision»
```

```
        sensor_report_topic = f»/model/{drone_name}/mark_sensor»
```

```
        self._swarm_mission_subscription = self.create_subscription(
```

```
            SwarmMission, mission_topic, self._swarm_mission_callback, sub_qos
```

```
        )
```

```
        self._drone_pos_subscription = self.create_subscription(
```

```
            Pose, pos_topic, self._pos_callback, 10
```

```
        )
```

```
        self._mark_sensor_subscription = self.create_subscription(
```

```

        SensorReport, sensor_report_topic, self._mark_sensor_callback, 10
    )
    self._proposal_subscription = self.create_subscription(
        Proposal, proposal_topic, self._proposal_callback, proposal_qos
    )
    self._decision_subscription = self.create_subscription(
        Decision, decision_topic, self._decision_callback, proposal_qos
    )

    self._cmd_move_publisher_ = self.create_publisher(Quaternion,
cmd_move_topic, 10)
    self._proposal_publisher_ = self.create_publisher(Proposal, proposal_topic,
proposal_qos)
    self._decision_publisher_ = self.create_publisher(Decision, decision_topic,
proposal_qos)

    self._drone_mover = DroneMover(
        self.drone_name,
        self.get_logger(),
        lambda msg: self._cmd_move_publisher_.publish(msg),
        lambda point: self.dc.add_visited_mark(point)
    )
    self.database = DroneStorage(self.drone_name)
    self._true_marks = {tuple(map(float, point)) for point in marks}
    self.dc = DecisionCore(self.drone_name, logger=self.get_logger(), true_marks =
self._true_marks)

    self.get_logger().info(f»Drone {self.drone_name} is ready for mission»)

def _swarm_mission_callback(self, swarm_mission: SwarmMission) -> None:
    current_drone_shift: DroneShift
    drone_shift: DroneShift
    for drone_shift in swarm_mission.drone_shifts:
        if drone_shift.drone_name != self.drone_name:
            continue

        current_drone_shift = drone_shift
        break

    self.get_logger().info(
        f»Got swarm mission, shift: {current_drone_shift}, start_eta_s:
{swarm_mission.start_eta_s}, desired_velocity: {swarm_mission.desired_velocity}»)
    )
    self._set_proposal_timer(swarm_mission.start_eta_s)
    self._follow_path(

```

```

        swarm_mission.centroid_path,
        current_drone_shift,
        swarm_mission.start_eta_s,
        swarm_mission.desired_velocity,
    )

def _pos_callback(self, pose: Pose):
    self._current_pose = pose
    self._drone_mover.pos_callback(pose)
    self._update_expected_marks(pose.position)

def _mark_sensor_callback(self, report: SensorReport) -> None:
    self.database.add_observation(self._proposal_round_id, report.marks_coords)

def _proposal_callback(self, proposal: Proposal) -> None:
    self.database.add_swarm_proposal(proposal.round_id, proposal)
    if self.database.get_swarm_round_proposals_count(proposal.round_id) >=
self.drones_in_swarm_count:
        swarm_proposals =
self.database.get_swarm_round_proposals(proposal.round_id)
        decision = self.dc.calculate_decision(proposal.round_id, swarm_proposals,
self._expected_marks[proposal.round_id], self._committed_prev)
        decision.insertion_segment = self._drone_mover._next_point_id
        decision.fraction_to_branch = self._drone_mover.get_branch_fraction()
        self._decision_publisher.publish(decision)
        self.get_logger().debug(f'Publishing decision {decision}»')

def _decision_callback(self, decision: Decision) -> None:
    self.database.add_swarm_decision(decision.round_id, decision)
    if self.database.get_swarm_round_decisions_count(decision.round_id) >=
self.drones_in_swarm_count:
        swarm_decisions =
self.database.get_swarm_round_decisions(decision.round_id)
        swarm_proposals =
self.database.get_swarm_round_proposals(decision.round_id)
        commit = self.dc.consider_quorum(decision.round_id, swarm_proposals,
swarm_decisions)
        if commit:
            self._committed_prev = commit
            if decision.round_id > self._last_action_round:
                action = self.dc.control_action(decision.round_id)
                self._perform_action(action)
                self._last_action_round = decision.round_id

def _perform_action(self, action: Decision) -> None:

```

```

        if action.decision == DECISION_FLY_TO_MARK and action.mark_coords !=
self._action.mark_coords:
            self._logger.info(f»Performing action {action}»)
            # Add mark to the path to visit it
            self._drone_mover.add_point_to_visit(action.mark_coords,
action.insertion_segment, action.fraction_to_branch)

        self._action = action

def _set_proposal_timer(self, start_time: float) -> None:
    if self.proposal_sync_delay_timer is None:
        now = time.monotonic()
        delay = max(0.0, start_time - now)
        self.get_logger().info(f»Setting proposal timer with delay {delay}s»)
        self.proposal_sync_delay_timer = self.create_timer(
            delay, self._start_proposal_timer, clock = self.steady_clock
        )

def _start_proposal_timer(self):
    self.proposal_sync_delay_timer.cancel()
    self._proposal_round_id = 0
    self.proposal_timer = self.create_timer(
        self.PROPOSAL_TIMER_PERIOD_S, self._handle_proposal_round, clock =
self.steady_clock
    )

def _handle_proposal_round(self):
    proposal = self.database.create_round_proposal(
        self._proposal_round_id, self._current_pose.position
    )
    self._proposal_publisher.publish(proposal)
    self.get_logger().debug(f»Publishing proposal {proposal}»)
    self._proposal_round_id += 1

def _on_follow_path_end(self):
    self.get_logger().info(
        f»Finished following path, current position {self._current_pose}»)
    )
    self.proposal_timer.cancel()
    self.dc.log_metrics()
    self.dc.theta_nn.save()

def _follow_path(
    self,

```

```

centroid_path: list[Point],
drone_shift: DroneShift,
start_eta_s: float,
desired_velocity: float,
):
    self.get_logger().info(«Following path»)
    self._action.decision = DECISION_FOLLOW_PATH
    self._action.drone_name = self.drone_name
    self._action.round_id = self._proposal_round_id
    self._drone_mover.follow_path(
        centroid_path,
        drone_shift,
        start_eta_s,
        desired_velocity,
        lambda: self._on_follow_path_end(),
    )

def _update_expected_marks(self, point: Point) -> None:
    p = tuple((point.x, point.y, point.z))
    R_sense = 40.0
    for mark in self._true_marks:
        if math.dist(mark, p) <= R_sense:
            self._expected_marks[self._proposal_round_id].add(mark)

```

```

from typing import Optional
from dataclasses import dataclass, field
import math

```

```

from geometry_msgs.msg import Point
from drone_interfaces.msg import Proposal, Decision

```

```

from .nn_adapter import ThetaNNAdapter
from .metrics import PropAgg, CommitAgg

```

```

from collections import defaultdict

```

```

@dataclass
class CandidateStats:
    sum_pos: float
    sum_neg: float
    share: float
    share_req: float
    share_surplus: float

```

```
anchors_len: int
anchors_ratio: float
```

```
@dataclass
class TrustSettings:
    s_min: int = 2
    k_max: int = 5
    weights: tuple[float, ...] = (1.0, 2.2, 0.8, 0.5, 0.2, 1.8)
    bias: float = -1.0
    gamma: float = 0.18
    tau_idle: float = 0.60
    tau_bootstrap: float = 0.60
    sensor_range: float = 40.0
    miss_penalty: float = 0.0
```

```
@dataclass
class GatingSettings:
    min_visible_support: int = 1
    min_trust_sum: float = 0.60
    visible_share_low: float = 0.50
    visible_share_high: float = 0.60
    anchor_visible_min: int = 4
    tau_anchor: float = 0.6
    min_anchor_count: int = 1
```

```
@dataclass
class FallbackSettings:
    small_vis_limit: int = 2
    tau_single_min_one: float = 0.78
    tau_single_min_two: float = 0.66
    tau_single_min_other: float = 0.72
    tau_pair_min_avg: float = 0.66
```

```
@dataclass
class PenaltySettings:
    commit_miss_penalty: float = 0.35
    commit_hit_reward: float = 0.12
    commit_false_penalty: float = 0.05
    visit_false_penalty: float = 0.18
```

```
@dataclass
class QuarantineSettings:
    tau_min: float = 0.55
    tau_release: float = 0.65
    min_quarantine_rounds: int = 1
```

```
release_streak: int = 2
```

```
@dataclass
```

```
class QuorumSettings:
    threshold: float = 0.67
    rep_lambda: float = 0.10
```

```
@dataclass
```

```
class NNGuardSettings:
    enabled: bool = True
    lower_share_margin: float = 0.05
    lower_pos_window: float = 0.05
    abs_share: float = 0.75
    anchor_bonus: int = 2
```

```
@dataclass
```

```
class NNCoverageSettings:
    enabled: bool = True
    min_opp_rounds: int = 2
    max_lower: float = 0.04
    share_floor: float = 0.62
    novelty_bonus: float = 0.0
```

```
@dataclass
```

```
class NNUncertaintySettings:
    pos_window: float = 0.15
    neg_window: float = 0.12
    neg_weight: float = 0.25
    opp_gain: float = 0.25
    min_weight: float = 0.05
    enabled: bool = True
```

```
@dataclass
```

```
class NNQuorumSettings:
    enabled: bool = True
    weight: float = 0.30
    share_floor: float = 0.82
    strength_pow: float = 1.0
```

```
@dataclass
```

```
class NNVisitWeights:
    true_weight: float = 2.6
    false_weight: float = 1.2
```

```
@dataclass
```

```

class NNSettings:
    enabled: bool = True
    model_path: str = «/home/mmelench/Documents/phd/dissertation/bfp_swarm/nn/»
    delta_scale: float = 0.08
    theta_min: float = 0.35
    theta_max: float = 0.70
    online: bool = True
    online_lr: float = 1e-4
    buffer_size: int = 256
    update_every: int = 2
    batch_size: int = 8
    seq_len: int = 8
    up_scale: float = 1.7
    p_center_beta: float = 0.997
    round_norm: float = 6.0
    low_trust_tau: float = 0.55
    unresolved_boost: float = 0.6
    guard: NNGuardSettings = field(default_factory=NNGuardSettings)
    coverage: NNCoverageSettings = field(default_factory=NNCoverageSettings)
    uncertainty: NNUncertaintySettings = field(default_factory=NNUncertaintySettings)
    quorum: NNQuorumSettings = field(default_factory=NNQuorumSettings)
    visit: NNVisitWeights = field(default_factory=NNVisitWeights)

```

@dataclass

```

class NoveltySettings:
    commit_new_bonus: float = 0.02
    commit_repeat_bonus: float = 0.01

```

@dataclass

```

class DecisionConfig:
    trust: TrustSettings = field(default_factory=TrustSettings)
    gating: GatingSettings = field(default_factory=GatingSettings)
    fallback: FallbackSettings = field(default_factory=FallbackSettings)
    penalties: PenaltySettings = field(default_factory=PenaltySettings)
    quarantine: QuarantineSettings = field(default_factory=QuarantineSettings)
    quorum: QuorumSettings = field(default_factory=QuorumSettings)
    nn: NNSettings = field(default_factory=NNSettings)
    novelty: NoveltySettings = field(default_factory=NoveltySettings)

```

DECISION_FOLLOW_PATH = «follow_path»

DECISION_FLY_TO_MARK = «fly_to_mark»

```

class DecisionCore:

```

```

    def __init__(self, drone_name: str, cfg: DecisionConfig | None = None,
logger=None, true_marks: set[tuple] | None = None):

```

```

self.drone_name = drone_name
self.cfg = cfg if cfg is not None else DecisionConfig()
self.state: dict = {«peers»: {}}
self.logger = logger
self.commit_by_round: dict[int, Decision] = {}
self.visited: set[tuple[float, float, float]] = set()
self._last_candidate_feats: dict | None = None

self._true_marks: set[tuple[float, float, float]] = set(true_marks) if true_marks else
set()
self._pa = PropAgg()
self._ca = CommitAgg()
self._ca.set_known_true_marks(self._true_marks)

self._committed_keys: set[tuple[float, float, float]] = set()
self._opp_counts: dict[tuple[float, float, float], int] = defaultdict(int)
self._failed_keys: set[tuple[float, float, float]] = set()
self._nn_pending: dict[tuple[float, float, float], list[dict]] = {}
self._pending_visit_keys: set[tuple[float, float, float]] = set()
self._latest_proposals: list[Proposal] = []
self._latest_quarantined: dict[str, bool] = {}
self._current_round_id: int = -1
self._last_commit_round_global: int | None = None
self._key_last_commit_round: dict[tuple[float, float, float], int] = {}
self._key_last_visit_round: dict[tuple[float, float, float], int] = {}

dummy_feats = self._build_nn_features(
    k_vis=0.0,
    k_rep=0.0,
    pos_margin=0.0,
    neg_ratio=0.0,
    share_surplus=0.0,
    share_abs=0.0,
    anchors_len=0,
    anchors_ratio=0.0,
    novelty_flag=0.0,
    visited_flag=0.0,
    opp_count=0,
    min_opp_req=max(1, self.cfg.nn.coverage.min_opp_rounds),
    extra={
        «tau_std»: 0.0,
        «low_trust_ratio»: 0.0,
        «distance_local_norm»: 0.0,
        «since_key_commit_norm»: 0.0,
        «since_key_visit_norm»: 0.0,
    }
)

```

```

        «since_any_commit_norm»: 0.0,
        «opp_active_flag»: 0.0,
        «failed_flag»: 0.0,
    },
)
input_dim = len(dummy_feats)
self.theta_nn = ThetaNNAdapter.from_cfg(
    self.drone_name,
    self.cfg.nn,
    self.logger,
    input_dim=input_dim,
)

def _pkey(
    self, p: Point | tuple[float, float, float]
) -> tuple[float, float, float]:
    if isinstance(p, Point):
        return (p.x, p.y, p.z)
    return (float(p[0]), float(p[1]), float(p[2]))

def _sigmoid(self, x: float) -> float:
    if x >= 0:
        z = math.exp(-x)
        return 1.0 / (1.0 + z)
    z = math.exp(x)
    return z / (1.0 + z)

def _clip01(self, x: float) -> float:
    return 0.0 if x < 0.0 else (1.0 if x > 1.0 else x)

def _normalize_round_delta(self, current: int, previous: int | None, norm: float) ->
float:
    if previous is None:
        return 1.0
    norm = max(1.0, float(norm))
    delta = max(0, int(current) - int(previous))
    return min(1.0, delta / norm)

def _build_nn_features(
    self,
    *,
    k_vis: float,
    k_rep: float,
    pos_margin: float,
    neg_ratio: float,

```

```

share_surplus: float,
share_abs: float,
anchors_len: int,
anchors_ratio: float,
novelty_flag: float,
visited_flag: float,
opp_count: int,
min_opp_req: int,
extra: dict | None = None,
) -> list[float]:
    opp_norm = 0.0
    if min_opp_req > 0:
        opp_norm = min(1.0, opp_count / float(min_opp_req))
    feats = [
        float(k_vis),
        float(k_rep),
        float(pos_margin),
        float(neg_ratio),
        float(share_surplus),
        float(share_abs),
        float(anchors_ratio),
        float(anchors_len),
        float(novelty_flag),
        float(visited_flag),
        float(opp_norm),
    ]
    extras = extra or {}
    feats.extend(
        [
            float(extras.get(«tau_std», 0.0)),
            float(extras.get(«low_trust_ratio», 0.0)),
            float(extras.get(«distance_local_norm», 0.0)),
            float(extras.get(«since_key_commit_norm», 0.0)),
            float(extras.get(«since_key_visit_norm», 0.0)),
            float(extras.get(«since_any_commit_norm», 0.0)),
            float(extras.get(«opp_active_flag», 0.0)),
            float(extras.get(«failed_flag», 0.0)),
        ]
    )
    return feats

def _nn_extra_features(
    self,
    *,
    round_id: int,

```

```

key: tuple[float, float, float],
reporters: set[str],
d2p: dict[str, Proposal],
opp_count: int,
min_opp_req: int,
) -> dict[str, float]:
    cfg = self.cfg
    peers = self.state.setdefault(«peers», {})
    tau_boot = float(cfg.trust.tau_bootstrap)

    reporter_taus: list[float] = []
    for dn in reporters:
        peer = peers.get(dn, {})
        reporter_taus.append(float(peer.get(«tau», tau_boot)))

    tau_std = 0.0
    low_trust_ratio = 0.0
    if reporter_taus:
        mean_tau = sum(reporter_taus) / len(reporter_taus)
        variance = sum((t - mean_tau) ** 2 for t in reporter_taus) / len(reporter_taus)
        tau_std = math.sqrt(max(0.0, variance))
        tau_low_thr = float(cfg.nn.low_trust_tau)
        low_trust_ratio = sum(1 for t in reporter_taus if t < tau_low_thr) /
len(reporter_taus)

    distance_local_norm = 1.0
    local_pr = d2p.get(self.drone_name)
    if local_pr is not None and getattr(local_pr, «drone_pos», None) is not None:
        pos = local_pr.drone_pos
        dx = float(key[0]) - float(pos.x)
        dy = float(key[1]) - float(pos.y)
        dz = float(key[2]) - float(pos.z)
        dist = math.sqrt(dx * dx + dy * dy + dz * dz)
        sense = float(cfg.trust.sensor_range)
        distance_local_norm = min(1.0, dist / max(1.0, sense))

    round_norm = float(cfg.nn.round_norm)
    since_key_commit_norm = self._normalize_round_delta(
        round_id, self._key_last_commit_round.get(key), round_norm
    )
    since_key_visit_norm = self._normalize_round_delta(
        round_id, self._key_last_visit_round.get(key), round_norm
    )
    since_any_commit_norm = self._normalize_round_delta(
        round_id, self._last_commit_round_global, round_norm

```

)

```
opp_active_flag = 1.0 if opp_count >= max(1, int(min_opp_req)) else 0.0
failed_flag = 1.0 if key in self._failed_keys else 0.0
```

```
return {
    «tau_std»: tau_std,
    «low_trust_ratio»: low_trust_ratio,
    «distance_local_norm»: distance_local_norm,
    «since_key_commit_norm»: since_key_commit_norm,
    «since_key_visit_norm»: since_key_visit_norm,
    «since_any_commit_norm»: since_any_commit_norm,
    «opp_active_flag»: opp_active_flag,
    «failed_flag»: failed_flag,
}
```

```
def compute_local_tau(
    self,
    proposals: list[Proposal],
    committed_prev: Optional[Decision] | None = None,
) -> dict[str, float]:
```

```
    ««««
```

Обчислити/оновити локальну довіру τ_j для всіх джерел з переданих proposals.

Повертає dict {drone_name: tau}. Оновлює self.state[«peers»][...][«tau»] і [«last_marks»].

Важливо: якщо джерело цього раунду нічого не надіслало і не бачило спільних міток,

ми не штрафуємо – ставимо $\text{hat_tau} = \tau_{\text{idle}}$.

```
    ««««
```

```
trust = self.cfg.trust
s_min = trust.s_min
k_max = trust.k_max
gamma = trust.gamma
miss_pen = trust.miss_penalty
weights = trust.weights
bias = trust.bias
tau_idle = trust.tau_idle
tau_bootstrap = trust.tau_bootstrap
```

```
peers_state = self.state.setdefault(«peers», {})
```

```
counts: dict[tuple[float, float, float], int] = {}
```

```
for pr in proposals:
```

```
    for m in pr.marks_coords:
```

```

    key = self._pkey(m)
    counts[key] = counts.get(key, 0) + 1

drone2proposal = {pr.drone_name: pr for pr in proposals}
min_vis = self.cfg.gating.min_visible_support
feats: dict[str, tuple[float, float, float, float, float, float]] = {}
vis_now_map: dict[str, int] = {}

for drone_name, pr in drone2proposal.items():
    M = list(pr.marks_coords)
    n = max(1, len(M))

    u_hits = sum(1 for m in M if counts.get(self._pkey(m), 0) >= s_min)
    u = u_hits / n

    v_solo = sum(1 for m in M if counts.get(self._pkey(m), 0) == 1)
    v = v_solo / n

    r = 1.0 if M else 0.0

    peer_prev = peers_state.get(
        drone_name, {«last_marks»: [], «tau»: tau_bootstrap}
    )
    set_now = {self._pkey(p) for p in M}
    set_prev = {self._pkey(p) for p in peer_prev.get(«last_marks», [])}
    if set_now or set_prev:
        inter = len(set_now & set_prev)
        union = len(set_now | set_prev)
        s = inter / union if union > 0 else 0.5
    else:
        s = 0.5

    q = min(1.0, k_max / max(1, len(M)))

    if (
        committed_prev
        and committed_prev.decision == DECISION_FLY_TO_MARK
        and committed_prev.mark_coords is not None
    ):
        a = 1.0 if self._pkey(committed_prev.mark_coords) in set_now else 0.0
    elif committed_prev and committed_prev.decision ==
DECISION_FOLLOW_PATH:
        a = 1.0 if len(M) == 0 else 0.0
    else:
        a = 0.5

```

```

feats[drone_name] = (u, v, r, s, q, a)

vis_now_map[drone_name] = len(M)

taus: dict[str, float] = {}
for drone_name, (u, v, r, s, q, a) in feats.items():
    peer = peers_state.setdefault(drone_name, {'last_marks': [], 'tau':
tau_bootstrap})

    if (
        len(drone2proposal[drone_name].marks_coords) == 0
        and vis_now_map.get(drone_name, 0) == 0
    ):
        hat_tau = float(tau_idle)
    else:
        x = [u, -v, r, s, q, a]
        z = sum(wi * xi for wi, xi in zip(weights, x)) + bias
        hat_tau = self._sigmoid(z)

    peer['tau'] = (1.0 - gamma) * float(peer['tau']) + gamma * float(hat_tau)
    peer['tau'] = self._clip01(peer['tau'])
    taus[drone_name] = peer['tau']

if miss_pen > 0.0:
    observed = set(drone2proposal.keys())
    for dn, peer in peers_state.items():
        if dn not in observed:
            peer['tau'] = max(0.0, float(peer['tau']) - miss_pen)
            taus[dn] = peer['tau']

for pr in proposals:
    peer = peers_state.setdefault(pr.drone_name, {'last_marks': [], 'tau':
tau_bootstrap})
    peer['last_marks'] = list(pr.marks_coords)

return taus

def compute_quarantine(self) -> dict[str, bool]:
    quarantine = self.cfg.quarantine
    tau_min = quarantine.tau_min
    tau_rel = quarantine.tau_release
    qN = quarantine.min_quarantine_rounds
    rN = quarantine.release_streak

```

```

out: dict[str, bool] = {}
peers = self.state.setdefault(«peers», {})

for dn, peer in peers.items():
    tau = peer.get(«tau», 0.5)
    q_streak = peer.get(«q_streak», 0)
    r_streak = peer.get(«r_streak», 0)
    quarantined = peer.get(«quarantined», False)

    if not quarantined:
        if tau < tau_min:
            q_streak += 1
            if q_streak >= qN:
                quarantined = True
                r_streak = 0
        else:
            q_streak = 0
    else:
        if tau > tau_rel:
            r_streak += 1
            if r_streak >= rN:
                quarantined = False
                q_streak = 0
        else:
            r_streak = 0

    peer[«q_streak»] = q_streak
    peer[«r_streak»] = r_streak
    peer[«quarantined»] = quarantined
    out[dn] = quarantined

return out
def compute_quarantine_global(self) -> dict[str, bool]:
    quarantine = self.cfg.quarantine
    tau_min = quarantine.tau_min
    tau_rel = quarantine.tau_release
    qN = quarantine.min_quarantine_rounds
    rN = quarantine.release_streak
    bootstrap = self.cfg.trust.tau_bootstrap

    peers = self.state.setdefault(«peers», {})
    out: dict[str, bool] = {}

    for dn, peer in peers.items():
        rt = float(peer.get(«rep_tau», bootstrap))

```

```

gq_streak = int(peer.get(«gq_streak», 0))
gr_streak = int(peer.get(«gr_streak», 0))
gq = bool(peer.get(«gquarantined», False))

if not gq:
    if rt < tau_min:
        gq_streak += 1
        if gq_streak >= qN:
            gq = True
            gr_streak = 0
    else:
        gq_streak = 0
else:
    if rt > tau_rel:
        gr_streak += 1
        if gr_streak >= rN:
            gq = False
            gq_streak = 0
    else:
        gr_streak = 0

peer[«gq_streak»] = gq_streak
peer[«gr_streak»] = gr_streak
peer[«gquarantined»] = gq
out[dn] = gq

return out

def _cluster_counts(
    self, proposals: list[Proposal]
) -> dict[tuple[float, float, float], set[str]]:
    cc: dict[tuple[float, float, float], set[str]] = {}
    for pr in proposals:
        for m in pr.marks_coords:
            k = self._pkey(m)
            s = cc.setdefault(k, set())
            s.add(pr.drone_name)
    return cc

def _fuse_local_decision(
    self, round_id: int, proposals: list[Proposal], quarantined: dict[str, bool]
) -> tuple[Decision, float]:
    cfg = self.cfg
    gating = cfg.gating
    fallback = cfg.fallback

```

```

trust = cfg.trust
min_vis = gating.min_visible_support
tau_a = gating.tau_anchor
min_a = gating.min_anchor_count
a_vismin = gating.anchor_visible_min
min_pos_base = gating.min_trust_sum
share_lo = gating.visible_share_low
share_hi = gating.visible_share_high
min_opp_req = int(cfg.nn.coverage.min_opp_rounds)

small_vis_limit = fallback.small_vis_limit
tau_single_min_one = fallback.tau_single_min_one
tau_single_min_two = fallback.tau_single_min_two
tau_single_min_other = fallback.tau_single_min_other
tau_pair_min_avg = fallback.tau_pair_min_avg

allowed = {dn for dn, q in quarantined.items() if not q}
cc = self._cluster_counts(proposals)
d2p = {pr.drone_name: pr for pr in proposals}

def tau_of(dn: str) -> float:
    return self.state[«peers»].get(dn, {}).get(«tau», trust.tau_bootstrap)

def compute_stats(
    key_local: tuple[float, float, float],
    visible_local: set[str],
    reporters_local: set[str],
) -> CandidateStats:
    pos_set = {dn for dn in reporters_local if dn != self.drone_name}
    neg_set = {dn for dn in (visible_local - reporters_local) if dn !=
self.drone_name}
    sum_pos_local = sum(tau_of(dn) for dn in pos_set)
    sum_neg_local = sum(tau_of(dn) for dn in neg_set)
    share_req_local = share_lo if len(visible_local) <= a_vismin else share_hi
    denom = sum_pos_local + sum_neg_local
    share_local = (sum_pos_local / denom) if denom > 0.0 else 0.0
    share_surplus_local = max(0.0, share_local - share_req_local)
    anchors_len_local = 0
    if len(visible_local) >= a_vismin:
        anchors_len_local = sum(
            1
            for dn in reporters_local
            if tau_of(dn) >= tau_a or dn == self.drone_name
        )
    anchors_ratio_local = anchors_len_local / max(1.0, len(visible_local))

```

```

return CandidateStats(
    sum_pos=sum_pos_local,
    sum_neg=sum_neg_local,
    share=share_local,
    share_req=share_req_local,
    share_surplus=share_surplus_local,
    anchors_len=anchors_len_local,
    anchors_ratio=anchors_ratio_local,
)

def compose_nn_features(
    *,
    key_local: tuple[float, float, float],
    visible_local: set[str],
    reporters_local: set[str],
    stats_local: CandidateStats,
    opp_count_local: int,
) -> list[float]:
    extra_local = self._nn_extra_features(
        round_id=round_id,
        key=key_local,
        reporters=reporters_local,
        d2p=d2p,
        opp_count=opp_count_local,
        min_opp_req=min_opp_req,
    )
    novelty_flag_local = 1.0 if (key_local not in self._committed_keys and not
self.has_visited(key_local)) else 0.0
    visited_flag_local = 1.0 if self.has_visited(key_local) else 0.0
    denom = stats_local.sum_pos + stats_local.sum_neg + 1e-9
    return self._build_nn_features(
        k_vis=float(len(visible_local)),
        k_rep=float(len(reporters_local)),
        pos_margin=float(stats_local.sum_pos - min_pos_base),
        neg_ratio=float(stats_local.sum_neg / denom),
        share_surplus=stats_local.share_surplus,
        share_abs=stats_local.share,
        anchors_len=stats_local.anchors_len,
        anchors_ratio=stats_local.anchors_ratio,
        novelty_flag=novelty_flag_local,
        visited_flag=visited_flag_local,
        opp_count=opp_count_local,
        min_opp_req=min_opp_req,
        extra=extra_local,
    )

```

```

best_key: Optional[tuple[float, float, float]] = None
best_score = -1.0
best_feats: Optional[list[float]] = None
theta_used: float | None = None

shadow_sum_pos = -1.0
shadow_theta: float | None = None
shadow_feats: Optional[list[float]] = None
shadow_key: Optional[tuple[float, float, float]] = None

self.logger.info(f»Proposals clusters: {cc}»)
for key, srcs in cc.items():
    mu = Point(x=key[0], y=key[1], z=key[2])

    if self.has_visited(key):
        continue

    reporters = srcs & allowed
    if not reporters:
        continue
    visible = set(allowed)
    self.logger.info(f»For point {key} reporters: {reporters}»)

    if len(reporters) <= small_vis_limit:
        sum_pos = sum(tau_of(dn) for dn in (reporters - {self.drone_name}))
        sum_neg = sum(tau_of(dn) for dn in ((visible - reporters) -
{self.drone_name}))
        theta = min_pos_base

    opp_gate = self._opp_counts.get(key, 0)
    min_opp_req_fallback = int(self.cfg.nn.coverage.min_opp_rounds)
    self_req = max(1, min_opp_req_fallback - 1)
    peer_req = max(2, min_opp_req_fallback)

    if len(reporters) == 1 and opp_gate >= self_req:
        only = next(iter(reporters))
        t_only = tau_of(only)
        self.logger.info(f»Falling into single observer case: {t_only}»)
        if (
            only == self.drone_name
            and t_only >= tau_single_min_one
            and key not in self._failed_keys
        ):
            self.logger.info(f»Storing best key and tau: {key} {t_only}»)

```

```

        best_key, best_score = key, t_only
        theta_used = theta
        shadow_sum_pos = max(shadow_sum_pos, sum_pos)
        shadow_theta = theta
        break
    else:
        self.logger.info(«Single observer fallback rejected; falling back to
general gating.»)

    if len(reporters) == 2 and opp_gate >= peer_req:
        if self.drone_name in reporters:
            other = next(iter(reporters - {self.drone_name}))
            tau_self = tau_of(self.drone_name)
            tau_other = tau_of(other)
            avg = 0.5 * (tau_self + tau_other)
            self.logger.info(f»Falling into double observer case: self={tau_self}
other={tau_other} avg={avg}»)
            if (
                tau_self >= tau_single_min_other
                and tau_other >= tau_single_min_other
                and avg >= tau_pair_min_avg
                and (key not in self._failed_keys)
            ):
                self.logger.info(f»Storing best key and tau: {key} {avg}»)
                best_key, best_score = key, avg
                theta_used = theta
                shadow_sum_pos = max(shadow_sum_pos, sum_pos)
                shadow_theta = theta
                break
            else:
                self.logger.info(«Double observer fallback rejected; falling back to
general gating.»)
        else:
            self.logger.info(«Double observer fallback rejected (local reporter
missing).»)

    if len(reporters) < min_vis:
        continue

    stats = compute_stats(key, visible, reporters)
    if stats.share < stats.share_req:
        continue
    if len(visible) >= a_vismin and stats.anchors_len < min_a:
        continue

```

```

theta0 = min_pos_base
pos_margin = float(stats.sum_pos - theta0)
anchors_len = stats.anchors_len
opp_count_current = self._opp_counts.get(key, 0)
feats = compose_nn_features(
    key_local=key,
    visible_local=visible,
    reporters_local=reporters,
    stats_local=stats,
    opp_count_local=opp_count_current,
)
theta = theta0
if stats.share >= stats.share_req and stats.sum_pos < theta0:
    self._opp_counts[key] = self._opp_counts.get(key, 0) + 1
else:
    self._opp_counts[key] = 0

if self.theta_nn is not None:
    novelty_allowed = key not in self._failed_keys
    theta_raw = self._adapt_theta(theta0, feats, key=key)
    theta_used = theta_raw
    guard_cfg = self.cfg.nn.guard
    if guard_cfg.enabled:
        allow_lower = True
        allow_lower &= (
            (stats.share >= (stats.share_req + guard_cfg.lower_share_margin)) or
            (stats.share >= guard_cfg.abs_share)
        )
        win = guard_cfg.lower_pos_window
        allow_lower &= (stats.sum_pos < theta0) and (stats.sum_pos >= (theta0 -
win))

        if len(visible) >= a_vismin:
            allow_lower &= (anchors_len >= (min_a + guard_cfg.anchor_bonus))
        else:
            allow_lower = True
    cov_allow = False
    coverage_cfg = self.cfg.nn.coverage
    share_floor_cov = float(coverage_cfg.share_floor)
    if coverage_cfg.enabled and novelty_allowed:
        novelty = (key not in self._committed_keys) and (not self.has_visited(key))
        if (
            novelty
            and stats.share >= share_floor_cov
            and self._opp_counts.get(key, 0) >= coverage_cfg.min_opp_rounds
        ):

```

```

        cov_allow = True
        cap = float(coverage_cfg.max_lower)
        if theta_raw < (theta0 - cap):
            theta_used = theta0 - cap
        if (theta_used < theta0) and (not (allow_lower or cov_allow)):
            theta = theta0
        else:
            theta = theta_used

    if stats.sum_pos > shadow_sum_pos:
        shadow_sum_pos = stats.sum_pos
        shadow_theta = theta
        shadow_feats = feats
        shadow_key = key

    if stats.sum_pos < theta:
        continue

    score = stats.sum_pos
    if (theta is not None) and (stats.sum_pos >= theta):
        if (key not in self._committed_keys) and (not self.has_visited(key)):
            score += float(self.cfg.nn.coverage.novelty_bonus)
        if score > best_score:
            best_score = score
            best_key = key
            best_feats = feats
            theta_used = theta

    if theta_used is None:
        theta_used = shadow_theta if shadow_theta is not None else min_pos_base
        best_feats = shadow_feats
    try:
        used_key = best_key if best_key is not None else shadow_key
        self._last_candidate_feats = (
            {«feats»: best_feats, «key»: used_key, «round_id»: round_id}
            if (best_feats is not None and used_key is not None) else None
        )
    except Exception:
        pass

    d = Decision()
    d.round_id = round_id
    d.drone_name = self.drone_name
    d.fraction_to_branch = -1.0
    if best_key is None:

```

```

d.decision = DECISION_FOLLOW_PATH
d.mark_coords = Point(x=math.nan, y=math.nan, z=math.nan)
else:
    d.decision = DECISION_FLY_TO_MARK
    d.mark_coords = Point(x=best_key[0], y=best_key[1], z=best_key[2])
    if self.logger:
        self.logger.info(f»Local decision: {d}»)

if self.theta_nn is not None:
    pos_win = float(self.cfg.nn.uncertainty.pos_window)
    def _entropy(p: float) -> float:
        if p <= 0.0 or p >= 1.0:
            return 0.0
        return float(-(p*math.log(p) + (1.0-p)*math.log(1.0-p)) / math.log(2.0))
    for key_u, srcs_u in cc.items():
        if self.has_visited(key_u):
            continue
        reporters_u = srcs_u & allowed
        if len(reporters_u) < min_vis or len(reporters_u) <= small_vis_limit:
            continue
        visible_u = set(allowed)
        stats_u = compute_stats(key_u, visible_u, reporters_u)
        if stats_u.share < stats_u.share_req:
            continue
        theta0_u = min_pos_base
        pos_margin_u = float(stats_u.sum_pos - theta0_u)
        if not (-pos_win <= pos_margin_u < 0.0):
            continue
        opp_u = self._opp_counts.get(key_u, 0)
        feats_u = compose_nn_features(
            key_local=key_u,
            visible_local=visible_u,
            reporters_local=reporters_u,
            stats_local=stats_u,
            opp_count_local=opp_u,
        )
        u_soft = _entropy(stats_u.share) * (1.0 - abs(pos_margin_u) / max(1e-6,
pos_win))
        extra_count = max(0, opp_u - min_opp_req)
        u_soft *= (1.0 + self.cfg.nn.uncertainty.opp_gain * extra_count)
        if u_soft > 0.0:
            try:
                p_pred = self.theta_nn.predict_p(feats_u, key=key_u)
                self.theta_nn.observe_uncertain(key=key_u, feats=feats_u, u=u_soft,
p=p_pred)

```

```

        except Exception:
            pass
    if self.cfg.nn.uncertainty.enabled:
        neg_win = float(self.cfg.nn.uncertainty.neg_window)
        if pos_margin_u <= -neg_win:
            share_excess = max(0.0, stats_u.share - stats_u.share_req) / max(1e-6,
1.0 - stats_u.share_req)
            neg_ratio_u = float(stats_u.sum_neg / (stats_u.sum_pos +
stats_u.sum_neg + 1e-9))
            w0 = float(self.cfg.nn.uncertainty.neg_weight)
            w_neg = w0 * share_excess * (1.0 + neg_ratio_u)
            w_neg *= (1.0 + self.cfg.nn.uncertainty.opp_gain * extra_count)
            w_neg = min(1.0, max(self.cfg.nn.uncertainty.min_weight, w_neg))
            try:
                self.theta_nn.observe(feats_u, y=0, w=w_neg, key=key_u)
            except Exception:
                pass
    return d, theta_used

```

```

def calculate_decision(
    self,
    round_id: int,
    proposals: list[Proposal],
    expected_marks: set,
    committed_prev: Optional[Decision] = None,
) -> Decision:
    self._current_round_id = round_id
    taus = self.compute_local_tau(proposals, committed_prev)
    quarantined = self.compute_quarantine()
    self.logger.debug(f'Quarantined: {quarantined}»)
    decision, theta = self._fuse_local_decision(round_id, proposals, quarantined)
    self._pa.update_from_decision(expected_marks, decision, theta)
    self._latest_proposals = proposals
    self._latest_quarantined = quarantined
    self._process_pending_visit_labels()
    return decision

```

```

def _dkey(self, d: Decision) -> tuple[str, Optional[tuple[float, float, float]]]:
    if d.decision == DECISION_FLY_TO_MARK:
        return (
            DECISION_FLY_TO_MARK,
            (d.mark_coords.x, d.mark_coords.y, d.mark_coords.z),
        )
    return (DECISION_FOLLOW_PATH, None)

```

```

def consider_quorum(
    self,
    round_id: int,
    proposals: list[Proposal],
    decisions: list[Decision],
    update_reputation: bool = True,
) -> Decision:
    self._current_round_id = round_id

    cfg = self.cfg
    gating = cfg.gating
    quorum_cfg = cfg.quorum
    tau_a = gating.tau_anchor
    min_a = gating.min_anchor_count
    a_vismin = gating.anchor_visible_min
    share_lo = gating.visible_share_low
    share_hi = gating.visible_share_high
    min_pos = gating.min_trust_sum

    gq = self.compute_quarantine_global()
    trust = cfg.trust

    def tau_of(dn: str) -> float:
        return self.state[«peers»].get(dn, {}).get(«rep_tau», trust.tau_bootstrap)

    d2p: dict[str, Proposal] = {pr.drone_name: pr for pr in proposals}

    buckets: dict[tuple[str, Optional[tuple[float, float, float]]], set[str]] = {}
    for dec in decisions:
        if gq.get(dec.drone_name, False):
            continue
        k = self._dkey(dec)
        buckets.setdefault(k, set()).add(dec.drone_name)

    def visible_for(mu_key: tuple[float, float, float]) -> set[str]:
        return {
            dn
            for dn, pr in d2p.items()
            if not gq.get(dn, False)
            and any((m.x, m.y, m.z) == mu_key for m in pr.marks_coords)
        }

    best_key: Optional[tuple[str, Optional[tuple[float, float, float]]]] = None
    best_score = -1.0

```

```

for key, reporters in buckets.items():
    if key[0] != DECISION_FLY_TO_MARK:
        continue
    mx, my, mz = key[1]
    visible = visible_for((mx, my, mz))
    if not visible:
        continue
    rep_vis = reporters & visible
    if not rep_vis:
        continue

    share_req = share_lo if len(visible) <= a_vismin else share_hi
    if len(visible) >= a_vismin:
        anchors = [dn for dn in rep_vis if tau_of(dn) >= tau_a]
        if len(anchors) < min_a:
            continue

    sum_pos = sum(tau_of(dn) for dn in rep_vis)
    sum_neg = sum(tau_of(dn) for dn in (visible - rep_vis))
    if sum_pos < min_pos:
        continue
    share = sum_pos / (sum_pos + sum_neg) if (sum_pos + sum_neg) > 0 else 0.0
    if share < share_req:
        continue

    score = sum_pos
    mx, my, mz = key[1]
    k3 = (mx, my, mz)
    already_true = k3 in getattr(self, '_ca', «_uniq_true», set())
    novelty_cfg = cfg.novelty
    if not already_true:
        score += float(novelty_cfg.commit_new_bonus)
    elif k3 not in self._committed_keys:
        score += float(novelty_cfg.commit_repeat_bonus)
    if score > best_score:
        best_score = score
        best_key = key

    commit = Decision()
    commit.round_id = round_id
    commit.drone_name = «««
    commit.fraction_to_branch = -1.0

    q_strength = 0.0
    if best_key is not None:

```

```

total_w = sum(tau_of(dn) for dn in visible_for(best_key[1]))
w_yes = sum(tau_of(dn) for dn in buckets.get(best_key, set()))
q_strength = (w_yes / total_w) if total_w > 0.0 else 0.0
if total_w > 0.0 and q_strength >= quorum_cfg.threshold:
    commit.decision = DECISION_FLY_TO_MARK
    mx, my, mz = best_key[1]
    commit.mark_coords = Point(x=mx, y=my, z=mz)
    self._committed_keys.add((mx, my, mz))
    self._key_last_commit_round[(mx, my, mz)] = round_id
    self._last_commit_round_global = round_id
else:
    commit.decision = DECISION_FOLLOW_PATH
    commit.mark_coords = Point(x=math.nan, y=math.nan, z=math.nan)
else:
    commit.decision = DECISION_FOLLOW_PATH
    commit.mark_coords = Point(x=math.nan, y=math.nan, z=math.nan)

supporters = buckets.get(best_key, set())
segs = [d.insertion_segment for d in decisions if d.drone_name in supporters]
commit.insertion_segment = (max(segs) if segs else -1)

fracs = [dec.fraction_to_branch for dec in decisions if not gq.get(dec.drone_name,
False) and self._dkey(dec) == best_key]
commit.fraction_to_branch = max(frac, default=-1.0)

if update_reputation and commit.decision == DECISION_FLY_TO_MARK:
    lam = quorum_cfg.rep_lambda
    peers = self.state.setdefault(«peers», {})
    target_key = self._dkey(commit)
    for dec in decisions:
        if gq.get(dec.drone_name, False):
            continue
        peer = peers.setdefault(dec.drone_name, {«rep_tau»: trust.tau_bootstrap})
        t = 1.0 if self._dkey(dec) == target_key else 0.0
        rt = float(peer.get(«rep_tau», trust.tau_bootstrap))
        peer[«rep_tau»] = self._clip01((1.0 - lam) * rt + lam * t)

    self.apply_commit_feedback(round_id, proposals, commit)

self.logger.info(f»Quorum commit: {commit}»)
self.commit_by_round[round_id] = commit

y = -1
m = None
if commit.decision == DECISION_FLY_TO_MARK:

```

```

        m = (commit.mark_coords.x, commit.mark_coords.y, commit.mark_coords.z)
        y = 1 if m in self._true_marks else 0

    self._ca.update(y, key=m)

    if commit.decision == DECISION_FLY_TO_MARK:
        try:
            gating = self.cfg.gating
            trust = self.cfg.trust
            share_lo = gating.visible_share_low
            share_hi = gating.visible_share_high
            a_vismin = gating.anchor_visible_min
            theta0 = gating.min_trust_sum

            def tau_local(dn: str) -> float:
                return self.state[«peers»].get(dn, {}).get(«tau», trust.tau_bootstrap)

            mx, my, mz = commit.mark_coords.x, commit.mark_coords.y,
commit.mark_coords.z
            d2p: dict[str, Proposal] = {pr.drone_name: pr for pr in proposals}
            reporters = {
                dn for dn, pr in d2p.items()
                if any((m.x, m.y, m.z) == (mx, my, mz) for m in pr.marks_coords)
                and not gq.get(dn, False)
            }
            visible = {dn for dn in d2p if not gq.get(dn, False)}
            pos_set = {dn for dn in reporters if dn != self.drone_name}
            neg_set = {dn for dn in (visible - reporters) if dn != self.drone_name}
            sum_pos = sum(tau_local(dn) for dn in pos_set)
            sum_neg = sum(tau_local(dn) for dn in neg_set)
            denom = (sum_pos + sum_neg)
            share = (sum_pos / denom) if denom > 0 else 0.0
            share_req = share_lo if len(visible) <= a_vismin else share_hi
            k3 = (mx, my, mz)
            tau_anchor = gating.tau_anchor
            anchors_len = sum(1 for dn in reporters if tau_local(dn) >= tau_anchor or dn
== self.drone_name)
            anchors_ratio = anchors_len / max(1.0, len(visible))
            novelty_flag = 1.0 if (k3 not in self._committed_keys and not
self.has_visited(k3)) else 0.0
            visited_flag = 1.0 if self.has_visited(k3) else 0.0
            opp_count = self._opp_counts.get(k3, 0)
            min_opp_req = int(self.cfg.nn.coverage.min_opp_rounds)
            extra_commit = self._nn_extra_features(
                round_id=round_id,

```

```

        key=k3,
        reporters=reporters,
        d2p=d2p,
        opp_count=opp_count,
        min_opp_req=min_opp_req,
    )
    feats_commit = self._build_nn_features(
        k_vis=float(len(visible)),
        k_rep=float(len(reporters)),
        pos_margin=float(sum_pos - theta0),
        neg_ratio=float(sum_neg / (denom + 1e-9)),
        share_surplus=max(0.0, share - share_req),
        share_abs=share,
        anchors_len=anchors_len,
        anchors_ratio=anchors_ratio,
        novelty_flag=novelty_flag,
        visited_flag=visited_flag,
        opp_count=opp_count,
        min_opp_req=min_opp_req,
        extra=extra_commit,
    )
    self._nn_pending.setdefault(k3, []).append({
        «feats»: feats_commit,
        «round_id»: round_id,
    })
    if self.logger:
        self.logger.debug(
            f»[NN] pending store  $\mu$ ={{k3}} feats={{feats_commit}} «
            f»vis={{len(visible)}} rep={{len(reporters)}} pos={{sum_pos:.3f}}
            neg={{sum_neg:.3f}}»
        )
    except Exception as e:
        if self.logger:
            self.logger.warn(f»[NN] pending store failed: {{e}}»)
    cand = self._last_candidate_feats
    quorum_nn = self.cfg.nn.quorum
    if (
        self.theta_nn is not None
        and quorum_nn.enabled
        and commit.decision == DECISION_FLY_TO_MARK
        and cand
        and «key» in cand
        and cand[«key»] is not None
        and cand[«feats»] is not None
        and cand[«key»] == (commit.mark_coords.x, commit.mark_coords.y,

```

```

commit.mark_coords.z)
):
    share_floor = float(quorum_nn.share_floor)
    if q_strength >= share_floor:
        excess = max(0.0, q_strength - share_floor)
        conf = excess / max(1e-6, 1.0 - share_floor)
        pow_k = float(quorum_nn.strength_pow)
        conf = conf ** pow_k
        wq = float(quorum_nn.weight) * conf
        if wq > 0.0:
            self.theta_nn.observe(cand[«feats»], y=1, w=wq, key=cand[«key»])
    return commit

```

def control_action(self, round_id: int) -> Decision:

```

««««
Return the action the local drone should execute for round_id.
If a quorum commit exists, mirror it for this drone.
Otherwise fall back to follow_path.
««««
commit = self.commit_by_round.get(round_id)
if commit is None:
    d = Decision()
    d.round_id = round_id
    d.drone_name = self.drone_name
    d.decision = DECISION_FOLLOW_PATH
    d.mark_coords = Point(x=math.nan, y=math.nan, z=math.nan)
    d.insertion_segment = -1
    return d

return commit

```

def add_visited_mark(self, p: Point) -> None:

```

««««Call this when the local drone actually arrives at a mark.»«««
key = self._pkey(p)
self.visited.add(key)
if self._current_round_id >= 0:
    self._key_last_visit_round[key] = self._current_round_id
    self._ca.record_visit(key, key in self._true_marks)
    self._pending_visit_keys.add(key)
    self._process_pending_visit_labels()
if self.logger:
    self.logger.info(f»Visited mark recorded: {key}»)

```

def on_visit_verdict(
 self,

```

p: tuple[float, float, float],
is_true: bool,
reporters: set[str] | None = None,
) -> None:
    ««««
    Call this *once the swarm has arrived* and you can determine truth with certainty.
    Uses the stored commit-time features for this key (if any) to train the NN online.
    ««««
    k = self._pkey(p)
    if self._current_round_id >= 0:
        self._key_last_visit_round[k] = self._current_round_id
    if not is_true and reporters:
        self._penalize_false_visit(k, reporters)
    if not is_true:
        self._failed_keys.add(k)
        self._opp_counts.pop(k, None)
    else:
        self._failed_keys.discard(k)
    if self.theta_nn is None:
        return
    pending = self._nn_pending.pop(k, [])
    if not pending:
        if self.logger:
            self.logger.debug(f»[Train] No pending samples for visited mark {k}»)
        return
    visit_weights = self.cfg.nn.visit
    w = float(visit_weights.true_weight if is_true else visit_weights.false_weight)
    y = 1 if is_true else 0
    for rec in pending:
        feats = rec.get(«feats»)
        if feats is None:
            continue
        try:
            self.theta_nn.observe(feats, y=y, w=w, key=k)
        except Exception as e:
            if self.logger:
                self.logger.warn(f»[Train] visit update failed for {k}: {e}»)
    if self.logger:
        self.logger.info(f»[Train] visit labels applied: key={k} y={y} n={len(pending)}
w={w}»)

def _process_pending_visit_labels(self) -> None:
    if not self._pending_visit_keys:
        return
    if not self._latest_proposals:

```

```

        return
    decided: list[tuple[float, float, float]] = []
    for key in list(self._pending_visit_keys):
        derived = self._derive_visit_verdict(key, self._latest_proposals,
self._latest_quarantined)
        if derived is None:
            continue
        if self.logger:
            self.logger.info(
                f»Visit verdict for {key}: {'TRUE' if derived[0] else 'FALSE'}»
            )
        self.on_visit_verdict(key, derived[0], derived[1])
        decided.append(key)
    for key in decided:
        self._pending_visit_keys.discard(key)

def _derive_visit_verdict(
    self,
    key: tuple[float, float, float],
    proposals: list[Proposal],
    quarantined: dict[str, bool],
) -> Optional[tuple[bool, set[str]]]:
    visible: set[str] = set()
    reporters: set[str] = set()

    for pr in proposals:
        if quarantined.get(pr.drone_name, False):
            continue
        if any((m.x, m.y, m.z) == key for m in pr.marks_coords):
            visible.add(pr.drone_name)
            reporters.add(pr.drone_name)

    if not visible:
        return None

    positives = len(reporters)
    negatives = len(visible) - positives

    if positives == negatives:
        return None

    return positives > negatives, reporters

def has_visited(self, p: Point | tuple[float, float, float]) -> bool:
    return self._pkey(p) in self.visited

```

```

def clear_visited(self) -> None:
    self.visited.clear()

def get_visited_marks(self) -> list[tuple[float, float, float]]:
    return list(self.visited)

def _penalize_false_visit(
    self, key: tuple[float, float, float], reporters: set[str]
) -> None:
    penalties = self.cfg.penalties
    base_penalty = float(penalties.visit_false_penalty)
    commit_penalty = float(penalties.commit_false_penalty)
    penalty = base_penalty + commit_penalty
    peers = self.state.setdefault(«peers», {})
    for dn in reporters:
        peer = peers.get(dn)
        if peer is None:
            continue
        tau_old = float(peer.get(«tau», self.cfg.trust.tau_bootstrap))
        tau_new = max(0.0, tau_old - penalty)
        peer[«tau»] = tau_new
        if self.logger:
            self.logger.debug(
                f»τ penalty(FALSE visit) for {dn}: {tau_old:.3f} → {tau_new:.3f} (reported
{key})»
            )

def apply_commit_feedback(
    self, round_id: int, proposals: list[Proposal], commit: Decision
) -> None:
    ««««
    Ground-truth aware trust shaping:
    - If the committed mark is TRUE:
        • Penalize peers that could see it but did NOT report it.
        • (Optional) Reward peers that DID report it.
    - If the committed mark is FALSE:
        • Penalize peers that DID report it (i.e., supported a false commit).
        • No change for peers that correctly ignored it.
    Uses exact coordinate equality (no epsilon) – safe with your perfect sensor model.
    ««««
    if commit.decision != DECISION_FLY_TO_MARK:
        if self.logger:
            self.logger.debug(«Commit feedback: no-op (decision = follow_path)»)
        return

```

```

penalties = self.cfg.penalties
pen = penalties.commit_miss_penalty
rew = penalties.commit_hit_reward

mu = commit.mark_coords
key_mu = (mu.x, mu.y, mu.z)

drone2proposal = {pr.drone_name: pr for pr in proposals}

for dn, peer in self.state.setdefault(«peers», {}).items():
    pr = drone2proposal.get(dn)
    if pr is None:
        continue

    has = any((m.x, m.y, m.z) == key_mu for m in pr.marks_coords)
    old = peer.get(«tau», 0.5)
    penal_key = (dn, key_mu)

    if not has:
        continue

    if rew > 0.0:
        new = min(1.0, old + rew)
        peer[«tau»] = new
        if self.logger:
            self.logger.debug(
                f»τ reward for {dn}: {old:.3f} → {new:.3f} (reported committed
{key_mu})»)

def _adapt_theta(self, theta0: float, feats: list[float], key=None) -> float:
    adapter = self.theta_nn
    theta = adapter.adapt_theta(theta0, feats, key=key) if adapter is not None else
theta0
    self.logger.warn(f»A Theta: {theta}»)
    return theta

def on_commit_verdict(self, is_true: bool) -> None:
    return

def _nn_train_once(self, label: int, weight: float = 1.0) -> None:
    adapter = self.theta_nn
    lc = self._last_candidate_feats
    if not (adapter and lc and «feats» in lc and lc[«feats»] is not None):

```

```

        return
    rid = lc.get(«round_id»)
    if not hasattr(self, «_nn_trained_rounds»):
        self._nn_trained_rounds = set()
    if rid in self._nn_trained_rounds:
        return
    self._nn_trained_rounds.add(rid)
    key = lc[«key»] if «key» in lc else None
    adapter.observe(lc[«feats»], int(label), float(weight), key=key)

def log_metrics(self) -> None:
    self.logger.warn(f»PA Metrics: {self._pa.summary()})»)
    self.logger.warn(
        f»CA Metrics: {self._ca.summary()})»
    )

```

ДОДАТОК Б
Список публікацій здобувача

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Меленчуков М. Є., Волокита. А. М., «Нейронні мережі в виявленні атак на розподілені системи,» *Технічні науки та технології*. 2024. 1, №35, сс. 135-145, DOI: 10.25140/2411-5363-2024-1(35)-135-145 (Фахове наукове видання України категорії Б). *Здобувач здійснив аналітичну систематизацію архітектур нейронних мереж для забезпечення безпеки розподілених систем та визначив критерії ефективності інтелектуальних методів ідентифікації аномальної активності; Волокита А. М. здійснив загальне наукове керівництво, визначив пріоритетні напрями дослідження методів машинного навчання у розподілених системах та верифікував результати аналізу.*
2. Меленчуков М. Є., Волокита. А. М., «Дослідження моделей виявлення атак на розподілені системи за допомогою згорткових нейронних мереж,» *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2024. №3, сс. 224-229, DOI: 10.31891/2219-9365-2024-79-29 (Фахове наукове видання України категорії Б). *Здобувач розробив та провів порівняльний аналіз 9 конфігурацій згорткових нейронних мереж для захисту розподілених систем, реалізував препроцесинг та балансування вибірок реальних наборів даних кібератак, а також експериментально обґрунтував вибір оптимальних гіперпараметрів для максимізації точності детекції; Волокита А. М. формалізував процес вибору архітектури нейромережі для забезпечення безпеки розподілених систем.*
3. Меленчуков М. Є., Волокита. А. М., «Метод вибору лідера для підвищення ефективності комунікації в рої БПЛА,» *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025. №3, сс. 409-414, DOI: 10.31891/2219-9365-2025-83-50 (Фахове наукове видання України категорії Б). *Здобувач удосконалив адаптивний метод вибору лідера на основі модифікованого алгоритму зваженої кластеризації з лінійною обчислювальною складністю $O(N)$, що забезпечило підвищення комунікаційної ефективності розподілених*

систем; Волокита А. М. запропонував концептуальну модель ієрархічного управління та децентралізованої координації в динамічних розподілених системах.

4. Меленчуков М. Є., Волокита. А. М., «Адаптивний метод вибору лідера в мобільних розподілених системах на основі інтеграції динамічної довіри,» *Проблеми інформатизації та управління*. 2026. 1, №85, сс. 35-41, DOI: 10.18372/2073-4751.85.21091 (Фахове наукове видання України категорії Б). Здобувач формалізував інтеграцію динамічної довіри в алгоритм вибору лідера та через *Trace-Driven Simulation* підтвердив підвищення безпеки розподілених систем; Волокита А. М. здійснив наукове консультування щодо синтезу механізмів безпеки та ефективності.
5. Меленчуков М. Є., Волокита. А. М., «Метод динамічної оцінки довіри за допомогою модифікованого гістерезису у роях БПЛА,» *Телекомунікаційні та інформаційні технології*. 2026. 1, №90, сс. 218-224, DOI: 10.31673/2412-4338.2026.019021 (Фахове наукове видання України категорії Б). Здобувач обґрунтував метод динамічного оцінювання довіри під час виконання місії рою дронів, реалізувавши механізм безперервного оновлення коефіцієнта довіри; запропонував коефіцієнт динамічного порогу довіри у вигляді модуля модифікованого гістерезису, який забезпечує постійну регуляцію ваги репутації вузлів розподіленої системи, що дозволяє адаптивно виявляти та нейтралізувати вплив підозрілих учасників; Волокита А. М. здійснив наукове керівництво щодо синтезу механізмів безпеки та валідації результатів.

Апробація наукових результатів дисертації:

6. Melenchukov M., Volokyta A. Analysis of defense and attack methods in distributed systems. The International Conference on Security, Fault Tolerance, Intelligence, (Kyiv, June 28, 2024). Kyiv, 2024. URL: <https://icsfti-proc.kpi.ua/issue/view/16681> (дата звернення: 20.03.2026). Здобувач провів класифікацію методів захисту та аналіз вразливостей вузлів у розподілених системах та систематизував

вектори деструктивного впливу кіберзагроз на їх відмовостійкість; Волокита А. М. визначив вектори впливу кіберзагроз на показники безпеки розподілених систем.

7. Melenchukov M. Improving safety and efficiency of distributed systems: a predictive navigation model for uav swarms. The International Conference on Security, Fault Tolerance, Intelligence, (Kyiv, June 30, 2025). Kyiv, 2025. URL: <https://icsfti-proc.kpi.ua/article/view/354612> (дата звернення: 20.03.2026). *Здобувач формалізував децентралізовану архітектуру управління, що інтегрує ймовірнісне 3D-картографування (OctoMap) та предиктивне алгоритмічне планування руху, та експериментально підтвердив підвищення ефективності функціонування системи при одночасному зниженні частоти структурних колізій.*

ДОДАТОК В

Теоретичні основи та комунікаційні обмеження сучасних розподілених систем

Сучасні інформаційно-керуючі комплекси проходять етап глибинної трансформації архітектури, переходячи від монолітних централізованих рішень до складних, географічно розподілених структур. Базовим об'єктом дослідження у цьому контексті виступає розподілена система (*Distributed System*, РС). Згідно з класичним визначенням Таненбаума (*Tanenbaum*) та ван Стіна (*van Steen*) [27], **розподіленою системою** є сукупність незалежних комп'ютерів, що постають перед користувачами як єдина цілісна система. Автори детально описують базові архітектурні рішення, що вирішують проблеми масштабованості та відмовостійкості, проте розкриті класичні підходи не повною мірою покривають жорсткі часові обмеження, характерні для мобільних роїв БПЛА у цій дисертації.

Ключовими характеристиками, що відрізняють РС від традиційних централізованих систем, є:

- Відсутність глобального годинника: Кожен вузол має власний локальний відлік часу, що унеможливорює абсолютну синхронізацію подій без застосування спеціальних алгоритмів.
- Незалежні збої компонентів: Вихід з ладу одного вузла не призводить до зупинки всієї системи, що забезпечує потенційно вищу надійність, але водночас ускладнює діагностику та керування станом.
- Конкурентність виконання: Процеси на різних вузлах виконуються паралельно та асинхронно, взаємодіючи через передачу повідомлень (*message passing*), а не через спільну пам'ять.

Еволюція архітектур розподілених систем пройшла шлях від жорстко зв'язаних кластерів (*Cluster Computing*) та ґрид-систем (*Grid Computing*) до сучасних мобільних та повсюдно розподілених середовищ. Особливої ваги сьогодні набувають системи, де вузли не просто обробляють інформацію, а безпосередньо взаємодіють з фізичним світом.

У цьому контексті доцільно виділити специфічний клас систем – **Розподілені**

Системи Координації Цільової Дії (РС КЦД). На відміну від класичних обчислювальних РС, де головною метою є максимізація пропускну здатності обробки даних (*throughput*), метою РС КЦД є досягнення узгодженого фізичного ефекту (*Coordinated Action*) у реальному часі та просторі.

Типовими прикладами РС КЦД є групи автономних роботів, сенсорні мережі керування промисловими процесами та, зокрема, рої безпілотних літальних апаратів (БПЛА). Для цього класу систем характерні такі специфічні вимоги:

- Жорстка взаємозалежність агентів: Дії одного вузла (наприклад, маневр у строю) фізично обмежують простір рішень для інших вузлів.
- Критичність часових обмежень: Цінність інформації та керуючої команди стрімко деградує з часом; запізнiла реакція може бути небезпечнішою за відсутність реакції.
- Спільний динамічний контекст: Усі агенти повинні підтримувати узгоджене уявлення про стан середовища (*situational awareness*), яке постійно змінюється.

Сучасною технологічною реалізацією РС КЦД є мобільні самоорганізовані мережі (*Mobile Ad-hoc Networks, MANET*) та їхній авіаційний підклас – літаючі ad-hoc мережі (*Flying Ad-hoc Networks, FANET*). У таких мережах відсутня фіксована інфраструктура: вузли вільно переміщуються, динамічно встановлюючи та розриваючи зв'язки. У статті [30] визначено, що відсутність фіксованої інфраструктури забезпечує високу автономність та живучість системи. Автори класифікують моделі мобільності та протоколи, що частково розкриває проблему самоорганізації вузлів. З іншого боку, подані у цій роботі моделі не повною мірою враховують необхідність підтримки довіри в межах рою під час прихованих кібератак (візантійських відмов), що обґрунтовує потребу в розробці методів динамічної оцінки довіри в цій дисертації.

На сучасному етапі розвитку розподілених систем, особливо у сфері автономних роїв безпілотних апаратів (БПЛА) та Інтернету речей (IoT), спостерігається вибухове зростання обсягів інформації, що циркулює в контурах керування. Як зазначено у звіті [28], глобальний мобільний трафік даних

подвоюється із високою швидкістю, і значна частка цього приросту припадає на M2M-комунікації (Machine-to-Machine). Звіт чудово ілюструє загальні тенденції розвитку мереж та прогнозує проблеми масштабування. Проте подана аналітика орієнтується більшою мірою на цивільний сегмент, залишаючи поза увагою специфічні комунікаційні обмеження тактичних мобільних мереж, які усуваються запропонованими методами зменшення службового навантаження. Однак проста екстраполяція обсягів трафіку не відображає повної картини проблем, з якими стикаються системи критичного керування. Ключовим викликом є не лише абсолютний обсяг даних, а й диспропорція між вимогами до пропускної здатності та фізичними обмеженнями каналів зв'язку в умовах високої динаміки

Проблема «комунікаційного вузького горла» (*Communication Bottleneck*).

Сучасні сенсорні платформи, якими оснащуються автономні агенти (камери високої роздільної здатності 4K/8K, мультиспектральні сенсори, *LIDAR*), генерують потоки «сирих» даних щільністю від десятків до сотень мегабіт на секунду. Окрім того, у статті [29] аналізуються різноманітні комунікаційні проблеми мереж БПЛА. Хоча в огляді ґрунтовно систематизовано типові архітектури мереж і проблеми втрати пакетів, там мало уваги приділяється алгоритмічним підходам до підтримання координації за умов деградації якості зв'язку. Водночас пропускна здатність типових бездротових каналів тактичного рівня (наприклад, на базі MANET-радіостанцій або LoRaWAN для телеметрії) є на порядки нижчою, особливо в умовах активної протидії засобами радіоелектронної боротьби (РЕБ) або у складній електромагнітній обстановці, і потребує нових методів керування. Спроба передати весь обсяг даних до центру прийняття рішень неминуче призводить до перевантаження буферів, пакетних втрат і, як наслідок, до деградації якості керування.

Критичність часових затримок (*Latency Criticality*). Для розподілених систем реального часу (*Real-Time Systems*) параметром ефективності №1 стає не ширина смуги пропускання (*Throughput*), а часова затримка доставки повідомлень (*End-to-End Latency*). У контурі керування роєм БПЛА затримка – це інтервал часу між виникненням події (наприклад, виявленням перешкоди одним з агентів) та

моментом, коли команда реакції на цю подію буде отримана й виконана іншими учасниками групи.

Дослідження Юаня та інших, подане у статті [31], підтверджує, що стабільність розподіленого керування на пряму залежить від часових характеристик каналу зворотного зв'язку. Дана робота переконливо аргументує критичність наднизьких затримок (URLLC), але розроблені там оптимізації фокусуються в основному на диспетчеризації радіоресурсів і не розглядають логічну реорганізацію рою, яку ми пропонуємо для підвищення надійності. Якщо сумарна затримка (включаючи час обробки, черги та передачі) перевищує характерний час динаміки об'єкта керування, виникає фазовий зсув у контурі регулювання. Це призводить до розсинхронізації дій агентів, виникнення незатухаючих коливань у строю і, зрештою, до втрати керованості (наприклад, зіткнення дронів під час щільного маневрування).

Особливістю мобільних мереж є висока варіативність затримок (*Jitter*). Час доставки повідомлення може змінюватися на порядки в залежності від поточного розташування вузлів, інтерференції та навантаження на мережу. Тому вибір вузла-координатора (лідера) не може базуватися лише на статичних параметрах (таких як ID вузла або залишковий заряд батареї). Критично важливим стає врахування часу встановлення з'єднання (*Connection Establishment Time*) та поточної оцінки затримок до сусідів. Лідер, який має повний заряд батареї, але знаходиться в зоні «радіотіні» або має перевантажений канал, стає «вузьким місцем», що гальмує реакцію всього рою.

ДОДАТОК Г

Типологія задач керування об'єктами у розподілених системах

Визначення функціонального призначення системи керування розподіленими об'єктами вимагає глибокої декомпозиції, систематизації та детального аналізу задач, які така система повинна вирішувати в автоматичному режимі без участі людини-оператора. Специфіка сучасних розподілених систем (РС), особливо тих, що функціонують у високодинамічних, гетерогенних або навіть ворожих середовищах (до яких належать тактичні мобільні мережі, автономні рої дронів, георозподілені кластери баз даних), полягає у фундаментальній відсутності єдиного джерела істини та глобального синхронізованого часу. Ця архітектурна особливість породжує стан перманентної невизначеності: жоден окремих агент у системі не володіє миттєвим, повним і абсолютно точним знанням про глобальний стан усієї системи. У таких умовах підсистема керування повинна брати на себе відповідальність не лише за виконання прикладної цільової функції групи, а й за підтримку базових властивостей життєздатності (*liveness*) та безпеки (*safety*), балансуючи на межі між доступністю сервісу та узгодженістю даних. Задачі керування утворюють безперервний, замкнений, ітеративний цикл, що охоплює етапи від початкової самоорганізації хаотичної множини вузлів до кризового менеджменту в умовах часткових відмов та відновлення цілісності після збоїв. У ґрунтовній книзі [90] систематизовано фундаментальні алгоритми подолання таких криз. Це чудово пояснює теоретичні основи синхронізації та взаємного блокування (*deadlocks*), проте для реалізації в автономних роях БПЛА викладені алгоритми є надто перевантаженими комунікаційно, що викликає необхідність у їх адаптації.

Фундаментом для побудови будь-якої змістовної розподіленої взаємодії є процес формування та динамічної адаптації топології мережі. Цей базовий шар керування перетворює набір розрізнених обчислювальних вузлів на зв'язну систему і починається з етапу «холодного старту», коли кожен вузол перебуває у стані інформаційної ізоляції. Першочерговою задачею стає ініціалізація та виявлення сусідів (*Neighbor Discovery*), що реалізується через використання

циклічних широкомовних (*broadcast*) або групових (*multicast*) запитів, так званих «Hello»-пакетів. Однак, проста фіксація наявності сигналу є недостатньою; система керування повинна виконувати складний аналіз відповідей для оцінки якості каналів зв'язку, використовуючи метрики *RSSI* (*Received Signal Strength Indicator*), *LQI* (*Link Quality Indicator*) та статистику втрат пакетів. Критично важливим аспектом на цьому етапі є фільтрація «шуму» та випадкових відбиттів сигналу, а також гістерезис при прийнятті рішень про встановлення зв'язку, щоб уникнути ситуації «миготливих» лінків, які дестабілізують маршрутизацію. Лише після побудови коректної таблиці суміжності система може переходити до більш високорівневих задач.

У контексті кіберфізичних систем, таких як групи мобільної робототехніки або БПЛА, задача топології трансформується з суто мережевої у задачу кінематичного керування та підтримки просторового строю (*Formation Control*). Тут керування формулюється як багатофакторна задача оптимізації, де цільова функція включає мінімізацію відхилень агентів від вузлів заданої віртуальної геометричної структури (гратки, клину, колони) та забезпечення умов уникнення взаємних колізій. Для вирішення цієї задачі часто застосовуються методи штучних потенціальних полів, де кожен сусідній агент розглядається як джерело віртуальних сил: сили притягання діють на великих дистанціях для збереження комунікаційної зв'язності, а сили відштовхування активуються на малих дистанціях для забезпечення фізичної безпеки. Щоб забезпечити ефективну трансляцію керуючих команд від логічного центру до периферії в такій динамічній структурі, система повинна постійно підтримувати актуальний оверлейний граф зв'язків, виконуючи динамічну маршрутизацію. Це передбачає побудову та постійну перебудову мінімального остовного дерева (*Minimum Spanning Tree - MST*) або використання проактивних протоколів mesh-маршрутизації (наприклад, *OLSR* або *BATMAN*). У змістовному огляді [46] детально описано, як ці протоколи автоматично адаптують шляхи передачі даних при фізичному переміщенні вузлів, намагаючись мінімізувати кількість транзитних ділянок (хопів) та загальну мережеву затримку. Разом з тим, огляд констатує обмеженість таких протоколів

щодо стійкості до компрометації вузлів, що і розв'язано в даній дисертації шляхом додавання метрики довіри у функції вибору вузлів.

Встановивши надійну фізичну та логічну зв'язність, система стикається з наступним класом викликів – задачами масового оновлення конфігурації та узгодження станів розподілених процесів. Це стає критично важливим при зміні фази місії, оновленні політик безпеки або зміні параметрів цілевказування. Складність полягає в тому, що у ненадійному середовищі проста послідовна розсилка команд не гарантує їх доставки всім адресатам. Тому підсистема керування повинна реалізовувати механізми надійного розповсюдження (*Reliable Broadcasting*). Ефективним підходом, що демонструє високу стійкість до втрат пакетів, є використання епідемічних алгоритмів (*Gossip protocols*). Вони працюють за принципом біологічного поширення вірусів: кожен вузол періодично обирає випадкових сусідів і передає їм нову інформацію, що забезпечує експоненціальне насичення мережі даними навіть при частковій зв'язності.

Однак, простого поширення інформації часто недостатньо; однією з найскладніших проблем на цьому етапі є забезпечення транзакційної атомарності змін у розподіленому середовищі. Неприпустимою є ситуація «часткового застосування», коли через збій зв'язку половина системи переходить на новий протокол взаємодії, а інша половина залишається на старому, що призводить до логічної несумісності та колапсу системи. Для вирішення цієї дилеми використовуються класичні протоколи двофазної (2PC) або трифазної фіксації транзакцій. Як детально пояснюється у підручнику [91], вони гарантують, що глобальне оновлення або застосовується всіма справними вузлами одночасно, або відхиляється повністю, повертаючи систему до попереднього стабільного стану. Ці підходи надійно вирішують задачу консенсусу в стабільних мережах, але їх блокуючий характер при втраті координатора є критичною вразливістю у динамічних радіомережах і вимагає гнучкіших рішень, наведених у нашому дослідженні. Паралельно з цим відбувається постійне фонове узгодження глобальних параметрів, таких як синхронізація логічних годинників (наприклад, годинників Лампорта) для впорядкування подій та справедливий розподіл

спільних ресурсів.

Втім, навіть ідеально спроектована та налаштована система змушена функціонувати в реальних умовах, де апаратні та програмні відмови є нормою, а не винятком. Це висуває на перший план задачі моніторингу здоров'я та ізоляції несправностей. Вузли можуть виходити з ладу миттєво (*crash failures*) або, що гірше, продовжувати працювати некоректно, надсилаючи помилкові дані (візантійські збої). Для своєчасного виявлення таких ситуацій система використовує адаптивні детектори збоїв (*Failure Detectors*). Оскільки в умовах змінного навантаження мережі жорсткі фіксовані таймаути неминуче призводять до хибних спрацювань (оголошення справного вузла мертвим через затримку пакета), застосовуються ймовірнісні алгоритми (наприклад, *ϕ -accrual failure detector*). Вони динамічно накопичують статистику інтервалів прибуття «*heartbeat*»-повідомлень і розраховують поточну ймовірність збою вузла. Після того як підозра перетворюється на впевненість і вузол визнається несправним, ініціюється процедура виключення, відома як *Fencing* або *STONITH* (*Shoot The Other Node In The Head*). Це примусове логічне, а іноді й фізичне відключення або блокування портів проблемного вузла є критично необхідним для запобігання «розщепленню мозку» (*split-brain*), захисту спільних даних від пошкодження та гарантії того, що такий «зомбі»-вузол не впливатиме на прийняття колективних рішень.

Ще серйознішим викликом для стійкості розподіленої системи, ніж відмова окремого вузла, є обробка розривів мережі (*Network Partitioning*), коли через відмову комунікаційного обладнання або радіоперешкоди єдина система розпадається на декілька ізольованих островів. Згідно з фундаментальною теоремою CAP, сформульованою в знаковій праці [32], у такій ситуації система змушена робити жорсткий вибір між доступністю сервісу та узгодженістю даних. Це загальнотеоретичне положення зумовлено математичними законами: хоча компроміс неунікнений, дисертація демонструє, як пріоритезація безпеки та локальних маневрів дозволяє рою уникнути катастрофічних наслідків у моменти розривів. Система керування повинна вміти алгоритмічно розрізняти локальну

втрату зв'язку з одним вузлом та глобальний розкол мережі. Для прийняття автоматичних рішень в умовах ізоляції зазвичай застосовуються кворумні стратегії. Правило просте: лише та підгрупа, що володіє кворумом (зазвичай $2N+1$ голос), отримує мандат на продовження активних операцій та зміну стану системи. Вузли, що опинилися у меншості, повинні автоматично переходити у режим «Read-only» або безпечний режим (fail-safe), припиняючи виконання будь-яких критичних дій, щоб уникнути дивергенції станів. Після відновлення фізичної зв'язності починається складний і ресурсомісткий процес злиття (Merge), що включає обмін журналами операцій, виявлення конфліктів та їх вирішення за допомогою детермінованих правил (наприклад, «Last Write Wins») або використання спеціальних типів даних, що безконфліктно реплікуються (CRDT).

Життєвий цикл розподіленої системи не обмежується лише боротьбою з проблемами та ентропією; він також передбачає розвиток і масштабування через інтеграцію та провізний нових вузлів. Здатність системи до горизонтального масштабування прямо залежить від можливості автоматично, «безшовно» включати нові обчислювальні ресурси без зупинки основного циклу керування. У захищених системах цей процес починається з суворої авторизації та взаємної аутентифікації, де процедура «рукостискання» передбачає криптографічну перевірку цифрових сертифікатів для захисту від підключення ворожих агентів. Після успішного встановлення довіри відбувається передача стану (*State Transfer*), адже новий вузол приєднується до мережі з «чистим» станом і не знає поточної тактичної обстановки. Ключовим технічним викликом тут є забезпечення неблокуючої передачі актуального контексту (електронних карт, списків цілей, значень змінних) через механізми знімків стану (*snapshots*) або відтворення логів команд, дозволяючи системі продовжувати обслуговувати запити під час синхронізації новачка.

Усі вищезгадані процеси – від оновлення конфігурації до відновлення після розривів – значно ускладнюються, а іноді стають алгоритмічно нерозв'язними за прийнятний час, за відсутності єдиного координаційного центру. Це робить задачу виборів та заміни лідера критичним елементом архітектури. У нормальному

режимі роботи всі вузли моніторять активність поточного лідера через механізм періодичних підтверджень. При детектуванні його втрати ініціюється алгоритм виборів, такий як Raft, Paxos або Bully Algorithm. У ході голосування, як класично показано у статті [35], перемагає той кандидат, який має найбільш актуальний журнал подій та зумів зібрати підтримку більшості (кворуму) кластера. Стаття вирішує проблему єдиного координуючого центру для локальних мереж, проте вона зовсім не покриває проблему зловмисних вузлів. Тому ми адаптували механізм виборів так, щоб він зважав на динамічну довіру та фільтрував візантійську поведінку. Щоб уникнути хаосу, кожен новий лідер асоціюється з унікальним, монотонно зростаючим номером епохи (Term ID). Це дозволяє системі миттєво ідентифікувати та відкидати команди від «старих» лідерів, які могли тимчасово втратити зв'язок, але потім повернулися, помилково вважаючи себе досі повноважними координаторами.

Таким чином, детальний аналіз показує, що спроба вирішити весь спектр задач керування у повністю децентралізованому режимі (peer-to-peer) часто призводить до комбінаторного вибуху складності комунікацій та високої ймовірності виникнення тупикових ситуацій (deadlocks). У роботі [92] доведено, що наявність виділеного координатора дозволяє лінеаризувати потік подій, впорядкувати доступ до ресурсів та значно спростити логіку обробки конфліктів. Висновки цієї статті науково обґрунтовують доцільність використання алгоритмів вибору лідера. Однак базові алгоритми консенсусу розроблялися без урахування специфічних швидкостей зміни топології БПЛА, і саме для покриття цієї прогалини в дисертації запропоновано модифіковані протоколи вибору лідера як ключового елемента архітектури високонадійних розподілених систем керування спеціального призначення.