НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут телекомунікаційних систем

Кафедра інформаційних технологій в телекомунікаціях

До захисту допущено:
В.О. завідувача кафедри
_____ Валерій ПРАВИЛО
«____»_____2026
р.

ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційно-комунікаційні технології»
спеціальності 172 Телекомунікації та радіотехніка

на тему: «Модифікований метод маршрутизації на основі протоколу RPL для мереж критичного інтернету речей»

Виконав: здобувач вищої освіти 4 курсу, групи ТІ-22
Лисенко Павло В'ячеславович _____
Науковий керівник: старший викладач кафедри ІТТ НН ІТС
Курдеча Василь Васильович _____
Рецензент: доцент кафедри ТК НН ІТС
кандидат технічних наук, доцент
Авдєєнко Гліб Леонідович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____

Київ – 2026 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут телекомунікаційних систем

Кафедра інформаційних технологій в телекомунікаціях

Рівень вищої освіти – перший (бакалаврський)
Освітньо-професійна програма «Інформаційно-комунікаційні технології»
спеціальності 172 Телекомунікації та радіотехніка

ЗАТВЕДЖУЮ
В.О. завідувача кафедри
_____ Валерій ПРАВИЛО
«____»_____2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Лисенку Павлу В’ячеславовичу

1. Тема дипломної роботи: «Модифікований метод маршрутизації на основі протоколу RPL для мереж критичного інтернету речей», науковий керівник роботи старший викладач кафедри ІТТ ННІ ІТС, Курдеча Василь Васильович - затверджені наказом по університету від 26.05.2026 р. №1912-с .
2. Строк подання студентом дипломної роботи: 06.06.2026 р.

3. Вихідні дані до роботи архітектура мереж критичного інтернету речей, вимоги до надійності, затримки, відмовостійкості та енергоефективності передавання даних у LLN/IoT-мережах; специфіка протоколу RPL, структура DODAG, службові повідомлення DIO, DIS, DAO, об’єктивні функції OF0 та MRHOF; метрики маршрутизації ETX, затримка, завантаженість вузла, залишковий заряд батареї та клас критичності трафіку. Для моделювання використано середовище Contiki-NG/Cooja

4. Зміст пояснювальної записки дипломної роботи (перелік завдань, які потрібно розробити) 1. Проаналізувати особливості мереж критичного

інтернету речей, сфери їх застосування, ресурсні обмеження.

2. Дослідити існуючі протоколи маршрутизації для LLN та IoT-мереж і обґрунтувати вибір протоколу RPL.

3. Модифікувати метод маршрутизації для критичного Інтернету речей

4. Провести імітаційне моделювання модифікованого методу маршрутизації

5. Перелік графічного (ілюстративного) матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

6. Перелік ілюстративного матеріалу: презентація 19 слайдів

7. Дата видачі завдання 25.09.2025

Календарний план

№ з/п	Назва етапів виконання дипломної роботи (проекту)	Строк виконання етапів роботи	Примітка
1	Узгодження організаційних питань з науковим керівником	30.09.25-2.10.25	Виконано
2	Узгодження теми роботи та початок роботи над першим розділом	2.10.25-15.10.25	Виконано
3	Робота над першим, другим та третім розділами	15.10.25 - 25.03.26	Виконано
4	Розробка власного рішення та його перевірка	25.03.26 - 10.04.26	Виконано
7	Висновки про ефективність рішення	10.04.26 - 19.04.26	Виконано
8	Оформлення дипломної роботи	19.04.26 - 01.06.26	Виконано
9	Підготовка презентації до захисту	01.06.26 - 08.06.26	Виконано

Студент _____ Павло ЛИСЕНКО

Керівник роботи _____ Василь КУРДЕЧА

РЕФЕРАТ

Дипломна робота «Модифікований метод маршрутизації на основі протоколу RPL для мереж критичного інтернету речей» містить 109 сторіноки тексту, 22 рисунків та 7 таблиць, 5 додатків, а також використовує 31 одиницю літературних джерел посилання.

Об'єкт дослідження – процес маршрутизації у мережах критичного інтернету речей.

Предмет дослідження – модифікований метод маршрутизації на основі протоколу RPL.

Мета роботи – підвищення ефективності доставки даних у мережах критичного інтернету речей.

Методи дослідження: аналіз та порівняльний огляд протоколів маршрутизації для LLN та IoT-мереж, математичне моделювання, комп'ютерне симулювання мережевих топологій, статистичний аналіз результатів.

У роботі проведено аналіз особливостей мереж критичного інтернету речей та вимог до маршрутизації, досліджено архітектуру і принципи роботи протоколу RPL. Розроблено модифікований метод маршрутизації, що враховує критичність трафіку, стан каналів зв'язку, навантаження на вузли та вимоги до відмовостійкості. Запропоновано нову об'єктивну функцію для вибору батьківського вузла та механізм адаптації до відмов і перевантажень. Проведено порівняльне моделювання, яке підтвердило підвищення коефіцієнта доставки пакетів та зниження затримки порівняно з класичним RPL. Результати дослідження можуть бути застосовані при проектуванні та розгортанні мереж критичного інтернету речей у промислових, медичних, транспортних та інфраструктурних системах.

Роботу виконано в рамках НДР кафедри інформаційних технологій в телекомунікаціях НН ІТС № 0124U001559 "Технології Інтернету Речей для розумного будинку та розумного міста"

Ключові слова: ІНТЕРНЕТ РЕЧЕЙ, КРИТИЧНИЙ ІНТЕРНЕТ РЕЧЕЙ, МАРШРУТИЗАЦІЯ, RPL, DODAG, ВІДМОВОСТІЙКІСТЬ, ЯКІСТЬ ОБСЛУГОВУВАННЯ, LLN, МЕТРИКА МАРШРУТИЗАЦІЇ, НАДІЙНІСТЬ МЕРЕЖІ, ETX, ОБ'ЄКТИВНА ФУНКЦІЯ.

ABSTRACT

The thesis, entitled ‘A Modified Routing Method Based on the RPL Protocol for Critical Internet of Things Networks’, comprises 109 pages of text, 22 figures and 7 tables, 5 appendices, and cites 31 references.

Object of research – The routing process in critical Internet of Things networks.

Subject of research – a modified routing method based on the RPL protocol.

Purpose – Improving data delivery efficiency in critical Internet of Things networks.

Research methods: system analysis and comparative review of routing protocols for LLN and IoT networks, mathematical modeling, computer simulation of network topologies, statistical analysis of results.

The thesis analyses the features of critical IoT networks and routing requirements, and investigates the architecture and operating principles of the RPL protocol. A modified routing method has been developed that considers traffic criticality, link state, node load, and fault tolerance requirements. A new objective function for parent node selection and an adaptation mechanism for failures and overloads have been proposed. Comparative simulation confirmed an improvement in the packet delivery ratio and a reduction in latency compared to standard RPL.

The research results can be applied in the design and deployment of critical IoT networks in industrial, medical, transportation, and infrastructure systems.

This work was carried out as part of Research Project No. 0124U001559 of the Department of Information Technologies in Telecommunications, “Internet of Things Technologies for Smart Homes and Smart Cities”

Keywords: INTERNET OF THINGS, CRITICAL INTERNET OF THINGS, ROUTING, RPL, DODAG, FAULT TOLERANCE, QUALITY OF SERVICE, LLN, ROUTING METRIC, NETWORK RELIABILITY, ETX, OBJECTIVE FUNCTION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ 10

ВСТУП 11

РОЗДІЛ 1 АНАЛІЗ ОСОБЛИВОСТЕЙ МЕРЕЖ КРИТИЧНОГО ІНТЕРНЕТУ РЕЧЕЙ ТА ВИМОГ ДО МАРШРУТИЗАЦІЇ 13

1.1. Особливості архітектури та функціонування мереж критичного інтернету речей 13

1.1.1. Поняття критичного інтернету речей та типові сфери застосування 13

1.1.2. Класи вузлів і шлюзів, ресурсні обмеження 18

1.1.3. Вимоги до безперервності роботи 22

1.2. Вимоги до маршрутизації в критичних застосуваннях 25

1.2.1. Показники надійності та затримки 25

1.2.2. Пріоритезація трафіку та відмовостійкість 27

1.2.3. Масштабованість та енергоефективність 30

1.3. Аналіз існуючих протоколів маршрутизації для LLN та IoT-мереж 32

1.3.1. Огляд альтернативних протоколів маршрутизації 32

1.3.2. Порівняння підходів та критерії вибору протоколу 36

Висновки до розділу 1 38

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ПРОТОКОЛУ RPL ТА ПОСТАНОВКА ЗАДАЧІ ЙОГО МОДИФІКАЦІЇ 40

2.1. Обґрунтування вибору протоколу RPL та визначення його обмежень 40

2.1.1. Переваги протоколу RPL 40

2.1.2. Недоліки RPL в умовах критичного трафіку 44

2.2. Архітектура та принципи роботи протоколу RPL 49

2.2.1. Побудова DODAG та поняття рангу 50

2.2.2. Висхідні та низхідні маршрути 51

2.2.3. Призначення повідомлень DIO, DIS, DAO 52

2.3. Об'єктивні функції та метрики маршрутизації у RPL 54

2.3.1. Об'єктивні функції OF0 і MRHOF 54

2.3.2. Метрики ETX, затримка, енергетичні характеристики 55

2.3.3. Правила вибору батьківського вузла 57
2.4. Аналіз поведінки RPL у мережах критичного інтернету речей 58
2.4.1. Сценарії деградації якості обслуговування 58
2.4.2. Вплив перевантаження та втрати вузлів 60
2.5. Формування вимог до модифікованого методу маршрутизації 61
2.5.1. Цільові показники ефективності 61
2.5.2. Вимоги до сумісності, адаптивності та відмовостійкості 63
Висновки до розділу 2 64
РОЗДІЛ 3 РОЗРОБКА МОДИФІКОВАНОГО МЕТОДУ МАРШРУТИЗАЦІЇ НА ОСНОВІ RPL 65
3.1. Вибір та обґрунтування додаткових метрик маршрутизації 65
3.1.1. Комбінування показників надійності, затримки та енергії 64
3.1.2. Вагові коефіцієнти та врахування класу трафіку 67
3.2. Розробка правила вибору батьківського вузла 68
3.2.1. Алгоритм ранжування кандидатів 68
3.2.2. Обмеження на перемикання маршруту та зниження нестабільності 70
3.3. Розробка механізму адаптації до відмов і перевантажень 70
3.3.1. Виявлення деградації каналу та резервні маршрути 72
3.3.2. Умови перебудови DODAG та локальна реакція на збої 71
3.4. Опис процедури оновлення маршрутів та інформаційного обміну 73
3.4.1. Логіка оновлення службових повідомлень 73
3.4.2. Сумісність із базовими механізмами RPL 74
3.5. Формалізація запропонованого методу 74
3.5.1. Математичний опис алгоритму 75
3.5.2. Оцінка обчислювальної складності та очікуваного ефекту 76
Висновки до розділу 3 76
РОЗДІЛ 4 МОДЕЛЮВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОГО МЕТОДУ 78
4.1. Вибір середовища моделювання та параметрів мережі 78
4.1.1. Обґрунтування інструменту моделювання 78
4.1.2. Топології мережі та характеристики трафіку 79
4.2. Формування сценаріїв експериментального дослідження 80

4.2.1. Нормальний режим та критичне навантаження 81
4.2.2. Відмови вузлів та погіршення каналу 81
4.3. Реалізація базового та модифікованого варіантів RPL 82
4.3.1. Налаштування моделі та внесення змін у логіку маршрутизації 83
4.3.2. Перевірка коректності роботи алгоритму 84
4.4. Оцінювання за системою показників якості 84
4.4.1. Packet delivery ratio та середня затримка 85
4.4.2. Стабільність маршрутів, службовий overhead та енергоспоживання 86
4.5. Порівняльний аналіз результатів та практичні рекомендації 87
4.5.1. Порівняння з класичним RPL та умови найбільшої ефективності 88
4.5.2. Обмеження методу та рекомендації щодо застосування 89
Висновки до розділу 4 91
ЗАГАЛЬНІ ВИСНОВКИ 93
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 95
ДОДАТКИ 98
ДОДАТОК А. Заголовний файл цільової функції OF-CRIT 98
ДОДАТОК Б. Реалізація цільової функції OF-CRIT 99
ДОДАТОК В. Конфігураційний файл сценарію 1 — нормальний режим 102
ДОДАТОК Г. Зведена таблиця результатів імітаційного моделювання 106
ДОДАТОК Д. Скрипт обробки результатів симуляції 107

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

IoT – Internet of Things – Інтернет речей.

Critical IoT – Critical Internet of Things – Критичний інтернет речей.

RPL – Routing Protocol for Low-Power and Lossy Networks – Протокол маршрутизації для мереж з низьким енергоспоживанням та втратами.

LLN – Low-Power and Lossy Networks – Мережі з низьким енергоспоживанням та втратами.

DODAG – Destination-Oriented Directed Acyclic Graph – Орієнтований ациклічний граф з цільовим вузлом.

DIO – DODAG Information Object – Інформаційний об'єкт DODAG.

DIS – DODAG Information Solicitation – Запит на інформацію DODAG.

DAO – Destination Advertisement Object – Об'єкт оголошення маршруту.

ETX – Expected Transmission Count – Очікувана кількість передач.

OF – Objective Function – Об'єктивна функція.

OF0 – Objective Function Zero – Нульова об'єктивна функція.

MRHOF – Minimum Rank with Hysteresis Objective Function – Об'єктивна функція мінімального рангу з гістерезисом.

PDR – Packet Delivery Ratio – Коефіцієнт доставки пакетів.

QoS – Quality of Service – Якість обслуговування.

IETF – Internet Engineering Task Force – Організація зі стандартизації Інтернет.

MAC – Medium Access Control – Керування доступом до середовища.

6LoWPAN – IPv6 over Low-Power Wireless Personal Area Networks – IPv6 поверх бездротових мереж малої потужності.

ВСТУП

Стрімкий розвиток технологій Інтернету речей (IoT) відкрив нові можливості для автоматизації та моніторингу у критично важливих сферах – промисловості, медицині, транспорті та управлінні інфраструктурою. У цих галузях формується особлива категорія систем – критичний інтернет речей (Critical IoT), де навіть незначна затримка або втрата пакету даних може призвести до серйозних наслідків: аварій на виробництві, небезпечних ситуацій у системах охорони здоров'я або відмов у роботі критичної інфраструктури.

Для організації передачі даних у таких мережах широко застосовується протокол маршрутизації RPL (Routing Protocol for Low-Power and Lossy Networks), стандартизований IETF у RFC 6550. Протокол RPL розроблений для мереж з низьким енергоспоживанням та втратами (LLN) і забезпечує побудову ієрархічної деревоподібної топології – DODAG. Проте в умовах критичного IoT стандартний RPL демонструє суттєві обмеження: використовувані ним метрики маршрутизації не враховують пріоритет трафіку, стан завантаженості вузлів та вимоги до відмовостійкості, що призводить до деградації якості обслуговування у критичних сценаріях.

Таким чином, існує актуальна науково-технічна задача – розроблення модифікованого методу маршрутизації на основі протоколу RPL, який забезпечить підвищену надійність, стійкість до відмов та оперативність доставки даних у мережах критичного IoT.

Метою дипломної роботи є підвищення надійності, стійкості та оперативності доставки даних у мережах критичного інтернету речей шляхом розроблення модифікованого методу маршрутизації на основі протоколу RPL, який враховує критичність трафіку, стан каналів зв'язку, навантаження на вузли та вимоги до відмовостійкості.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) провести аналіз особливостей архітектури та функціонування мереж критичного інтернету речей, визначити вимоги до маршрутизації у таких мережах;
2) дослідити архітектуру, принципи роботи та об'єктивні функції протоколу RPL, виявити його обмеження в умовах критичного трафіку;
3) розробити модифікований метод маршрутизації на основі RPL із комбінованою метрикою, новим правилом вибору батьківського вузла та механізмом адаптації до відмов і перевантажень;
4) провести комп'ютерне моделювання і порівняльний аналіз ефективності запропонованого методу відносно стандартного RPL за ключовими показниками якості;
5) сформулювати практичні рекомендації щодо застосування розробленого методу.

Об'єкт дослідження – процес маршрутизації у мережах критичного інтернету речей.

Предмет дослідження – модифікований метод маршрутизації на основі протоколу RPL з урахуванням критичності трафіку, стану каналів зв'язку, навантаження на вузли та вимог до відмовостійкості.

Методи дослідження**.** У роботі використано: системний аналіз та порівняльний огляд науково-технічної літератури для вивчення існуючих підходів до маршрутизації в LLN та IoT-мережах; методи математичного моделювання для формалізації запропонованого методу; комп'ютерне симулювання для оцінювання ефективності розробленого рішення; статистичний аналіз для обробки та інтерпретації результатів моделювання.

Практичне значення отриманих результатів**.** Розроблений модифікований метод маршрутизації може бути використаний при проектуванні та розгортанні мереж критичного IoT у промислових системах автоматизації, медичних системах дистанційного моніторингу, транспортних системах, та у системах управління розумною інфраструктурою міст.

РОЗДІЛ 1

АНАЛІЗ ОСОБЛИВОСТЕЙ МЕРЕЖ КРИТИЧНОГО ІНТЕРНЕТУ РЕЧЕЙ ТА ВИМОГ ДО МАРШРУТИЗАЦІЇ

1.1. Особливості архітектури та функціонування мереж критичного інтернету речей

1.1.1. Поняття критичного інтернету речей та типові сфери застосування

Інтернет речей (IoT — Internet of Things) є однією з найбільш динамічно розвинутих технологічних парадигм сучасності. За визначенням Міжнародного союзу електрозв'язку (ITU-T Y.2060), IoT — це глобальна інформаційна інфраструктура, що забезпечує просунуті послуги шляхом з'єднання фізичних та віртуальних об'єктів на основі існуючих та нових, сумісних між собою інформаційних та комунікаційних технологій [24]. Відповідно до прогнозів аналітичних агентств, до 2030 року кількість підключених IoT-пристроїв перевищить 25 мільярдів, а генерований ними трафік складатиме значну частину від загального обсягу передачі даних в мережі Інтернет [5].

Поряд із масовими споживчими застосуваннями IoT (розумний будинок, носимі пристрої, побутова е0лектроніка) виокремлюється особлива категорія систем, де вимоги до надійності та часових характеристик є принципово вищими. Ця категорія отримала назву «критичний інтернет речей» (Critical IoT). За визначенням 3GPP у специфікації TS 22.261, Critical IoT охоплює сценарії використання, для яких характерні надвисока надійність (>99,9999%), наднизька затримка (від 0,5 до 10 мс) та детермінованість передачі даних [30]. На відміну від масового IoT (Massive IoT), де пріоритет надається охопленню та низькій вартості підключення

мільярдів пристроїв, у Critical IoT на першому місці стоїть гарантованість доставки критично важливих даних у заздалегідь визначені часові рамки.

Принципова відмінність між звичайним та критичним IoT полягає у наслідках можливих збоїв. У побутових застосуваннях IoT відмова в доставці пакету або затримка передачі даних призводить, як правило, до незручності для користувача, але не спричиняє суттєвої матеріальної шкоди чи загрози здоров'ю людей. Натомість у Critical IoT аналогічний збій може мати незворотні наслідки: зупинку технологічного процесу на промисловому підприємстві, помилку у медичній апаратурі, відмову гальмівної системи транспортного засобу або перебій у роботі об'єктів критичної інфраструктури — електростанцій, водозабезпечення, систем управління рухом [6]. Саме тому архітектура мереж Critical IoT та протоколи передачі даних у них мають відповідати суттєво більш жорстким вимогам, ніж у звичайних IoT-системах.

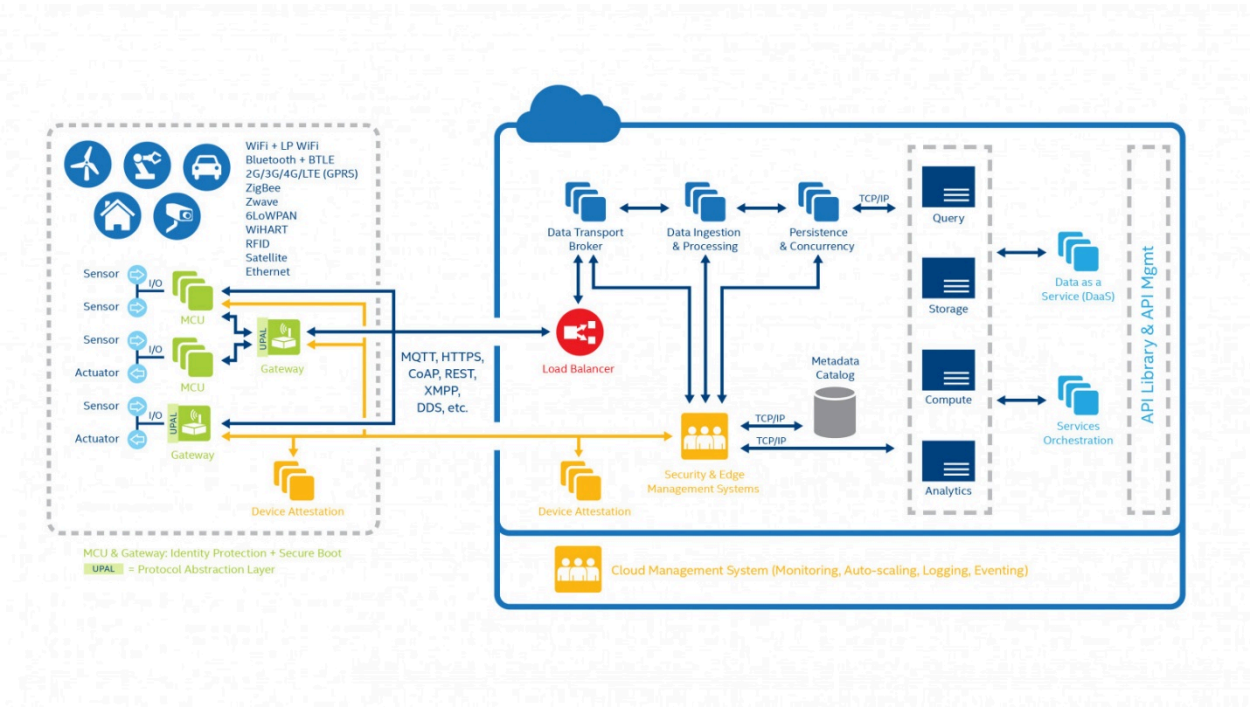

Рисунок 1.1 – Архітектура критичного IoT

Сфери застосування Critical IoT охоплюють широкий спектр галузей промисловості та суспільного господарства. Промислова автоматизація та системи керування виробничими процесами (Industrial Automation and Control

Systems — IACS) є одним із найбільш розвинутих напрямків. У цій сфері IoT-пристрої використовуються для збору телеметрії з датчиків технологічного обладнання, передачі команд виконавчим механізмам, моніторингу стану виробничих ліній у реальному часі. Затримка передачі даних у таких системах нормується на рівні 10–100 мс для контурів автоматичного управління та не повинна перевищувати 1–5 мс для захисних систем [29].

Системи розумної електроенергетики (Smart Grid) формують ще один ключовий клас застосувань Critical IoT. Інтелектуальні лічильники, пристрої захисту та автоматики підстанцій, системи управління навантаженням у режимі реального часу генерують потоки критичних даних, що потребують гарантованої доставки. Стандарт IEC 61850 для систем автоматизації підстанцій встановлює вимогу до затримки на рівні 4 мс для захисних функцій класу Performance Requirement P1/P2 [6]. Втрата або затримка пакету команди на відключення пошкодженого елемента енергосистеми може призвести до аварійного поширення пошкодження на суміжне обладнання.

У сфері охорони здоров'я та телемедицини Critical IoT застосовується для дистанційного моніторингу стану пацієнтів, роботи імплантованих медичних пристроїв (кардіостимуляторів, інфузійних помп, нейростимуляторів), управління системами контролю умов зберігання медичних препаратів та роботи операційних. Для кардіомоніторів та дефібриляторів критичне значення мають не лише часові параметри, але й цілісність переданих даних — навіть незначне спотворення сигналу ЕКГ може призвести до помилкового діагнозу [5]. Тому у медичному IoT поряд із вимогами до затримки висуваються жорсткі вимоги до достовірності та захищеності даних.

Інтелектуальні транспортні системи (Intelligent Transportation Systems — ITS) та системи управління безпілотним транспортом є областями, де Critical IoT набуває стратегічного значення. Системи попередження зіткнень (V2X — Vehicle-to-Everything), автоматичне управління рухом, контроль

позиції та стану рухомого складу на залізниці, авіаційні системи спостереження — усі ці застосування вимагають передачі даних з затримкою менше 10–50 мс та надійністю не нижче 99,99% [30]. Будь-який збій у роботі мережі може безпосередньо загрожувати безпеці людей.

Об'єкти критичної інфраструктури (водопостачання, газорозподіл, об'єкти ядерної енергетики, системи управління надзвичайними ситуаціями) також активно використовують IoT-технології для автоматизованого контролю та управління. Специфіка цих застосувань полягає в тому, що, крім вимог до часових параметрів, тут висуваються надзвичайно жорсткі вимоги до кібербезпеки та відмовостійкості — мережа повинна зберігати функціональність навіть за умови часткових відмов обладнання або зловмисних атак [26]. Таким чином, множина вимог до мереж Critical IoT є значно ширшою, ніж просто вимоги до затримки та надійності.

Окремого розгляду заслуговують вимоги 3GPP до класифікації сценаріїв використання IoT у мережах 5G. У стандарті TS 22.261 виокремлено три основні класи: масовий IoT (mMTC — massive Machine-Type Communications) для підключення великої кількості пристроїв з низькою швидкістю передачі даних; надійний IoT з малою затримкою (URLLC — Ultra-Reliable Low Latency Communications) для критичних застосувань; та широкосмуговий мобільний зв'язок (eMBB — enhanced Mobile Broadband) для передачі мультимедіа [30]. Мережі Critical IoT переважно відносяться до класу URLLC, хоча в реальних промислових системах спостерігається конвергенція усіх трьох класів на єдиній мережевій інфраструктурі.

Таким чином, критичний IoT є специфічним класом кіберфізичних систем, що характеризується поєднанням жорстких вимог до затримки, надійності та доступності мережі з обмеженими апаратними можливостями кінцевих пристроїв. Ця суперечність між вимогами до продуктивності та ресурсними обмеженнями є ключовою конструктивною проблемою при проектуванні мереж Critical IoT та їх протоколів маршрутизації.

Аналізуючи розподіл застосувань Critical IoT за галузями, слід відзначити, що промисловий сектор залишається найбільшим споживачем цих технологій. За даними ринкових досліджень, у 2023 році частка промислового IoT (IIoT) становила понад 65% від загального ринку Critical IoT, при цьому найбільш активний ріст спостерігається у секторах виробництва, енергетики та транспорту [29]. Такий розподіл зумовлений зрілістю промислових комунікаційних стандартів (Profibus, EtherNet/IP, Modbus) та накопиченим досвідом інтеграції цифрових систем управління в промислову інфраструктуру.

Формування концепції Critical IoT нерозривно пов'язане з розвитком стандартів технологічного рівня. Стандарт ISA-95 (ANSI/ISA-95) визначає модель інтеграції корпоративних та виробничих систем у вигляді ієрархії рівнів: рівень польових пристроїв (0), рівень управління виробничим обладнанням (1), рівень управління виробничими процесами (2), рівень управління виробничими операціями (MES, рівень 3) та рівень бізнес-планування (ERP, рівень 4). Мережі Critical IoT функціонують переважно на рівнях 0–2 цієї ієрархії, де вимоги до часу реакції найбільш жорсткі [29].

Конвергенція операційних технологій (OT — Operational Technology) та інформаційних технологій (IT) є глобальним трендом, що суттєво впливає на розвиток мереж Critical IoT. Традиційно OT-системи (SCADA, DCS, PLC) функціонували у відокремлених мережах з жорстко детермінованими часовими характеристиками, тоді як IT-мережі будувались на основі відкритих стандартів Ethernet/IP без жорстких часових гарантій. Конвергенція OT/IT призводить до появи єдиної гетерогенної мережевої інфраструктури, де поруч з традиційними Ethernet-пристроями функціонують IoT-вузли з обмеженими ресурсами та бездротовими каналами зв'язку. Ця конвергенція висуває нові вимоги до протоколів маршрутизації, здатних ефективно функціонувати в гетерогенних середовищах [13].

Окремим та активно розвиваючимся напрямком Critical IoT є системи точного землеробства та агрострахування. IoT-датчики для моніторингу мікроклімату теплиць, систем автоматизованого поливу, дронів для прецизійного обприскування, датчиків стану ґрунту формують мережі з сотень та тисяч пристроїв на великих площах. Специфікою цих мереж є надзвичайно велика географічна розосередженість при відносно низьких вимогах до затримки (секунди — хвилини), проте з жорсткими вимогами до PDR та енергоефективності через відсутність стаціонарного живлення та великі відстані до шлюзів [5].

Стандарти безпеки та функціональної безпеки (Functional Safety) є невід'ємною частиною вимог до мереж Critical IoT у ряді застосувань. Стандарт IEC 61508 «Функціональна безпека електричних/електронних/програмованих електронних систем, пов'язаних з безпекою» визначає рівні цілісності безпеки SIL (Safety Integrity Level) від SIL 1 до SIL 4. Для мереж передачі даних в системах з рівнем SIL 2 і вище встановлюються вимоги до ймовірності небезпечного відмови та часу відновлення, які безпосередньо транслюються у вимоги до PDR та часу реконвергенції протоколу маршрутизації [6].

1.1.2. Класи вузлів і шлюзів, ресурсні обмеження

Вимоги до маршрутизації у мережах Critical IoT суттєво відрізняються від вимог у традиційних комп'ютерних мережах та звичайних IoT-системах. Це зумовлено, з одного боку, ресурсними обмеженнями пристроїв та нестабільністю радіоканалів, а з іншого — необхідністю забезпечення гарантованих показників якості обслуговування для критично важливих потоків даних.

Архітектура мереж Critical IoT має ієрархічну організацію, що включає кілька рівнів: рівень кінцевих пристроїв (сенсорів та виконавчих механізмів), рівень мережевих вузлів-маршрутизаторів та рівень граничних

маршрутизаторів (border routers), які забезпечують підключення до зовнішньої IP-мережі або хмарної інфраструктури. Кожен рівень характеризується специфічним набором функцій та ресурсними можливостями.

Кінцеві пристрої (end devices або leaf nodes) є найбільш численним класом вузлів у мережах Critical IoT. До них відносяться датчики температури, тиску, вібрації, витратоміри, лічильники, медичні монітори, виконавчі механізми та контролери. Типові апаратні характеристики таких пристроїв: мікроконтролер з тактовою частотою 8–32 МГц (напр., ARM Cortex-M0/M3), оперативна пам'ять обсягом 4–64 КБ, флеш-пам'ять 32–256 КБ, радіотрансивер стандарту IEEE 802.15.4 з потужністю передачі 0–5 дБм та дальністю зв'язку 10–100 м [25]. Живлення кінцевих пристроїв, як правило, здійснюється від батарей ємністю 1000–5000 мА·год або за рахунок енергії, зібраної з навколишнього середовища (вібрація, освітлення, теплові градієнти). Очікуваний термін роботи від батареї — від одного до десяти років, що накладає жорсткі обмеження на енергоспоживання вузла.

Вузли-маршрутизатори (router nodes) виконують одночасно функції кінцевого пристрою та проміжного вузла для ретрансляції трафіку від сусідніх вузлів до кореневого вузла мережі. На відміну від кінцевих пристроїв, маршрутизатори повинні підтримувати таблиці маршрутизації та функціонувати у режимі постійної готовності, що суттєво підвищує їхнє енергоспоживання. Апаратні платформи маршрутизаторів зазвичай мають більш потужні процесори (ARM Cortex-M3/M4, 32–64 МГц), збільшений обсяг пам'яті (32–128 КБ RAM, 256 КБ – 1 МБ Flash) та можуть отримувати живлення від мережі або акумуляторів великої ємності [8].

Граничні маршрутизатори (border routers або sink nodes) є ключовими елементами мережі, що виконують функцію шлюзу між мережею Critical IoT та зовнішньою IP-інфраструктурою. Граничні маршрутизатори підтримують одночасно стек протоколів 6LoWPAN/RPL з боку мережі IoT та повноцінний IPv6 з боку зовнішньої мережі. До їхніх функцій належать: компресія

IPv6-заголовків (6LoWPAN), агрегація даних від множини вузлів, протоколи безпеки (DTLS, TLS), механізми управління мережею та моніторингу якості зв'язку [18]. Граничні маршрутизатори зазвичай реалізуються на значно потужніших апаратних платформах — одноплатних комп'ютерах (Raspberry Pi, BeagleBone) або вбудованих системах промислового класу з процесорами ARM Cortex-A або x86.

Ресурсні обмеження є фундаментальною характеристикою мереж LLN (Low-Power and Lossy Networks), до яких відносяться мережі Critical IoT. Ці обмеження безпосередньо визначають вимоги до протоколів маршрутизації. Обмежена обчислювальна потужність унеможливлює виконання складних алгоритмів з великою обчислювальною складністю — наприклад, алгоритм Дейкстри для мережі з N вузлами має складність $O(N^2 \log N)$, що неприйнятно для мікроконтролерів з тактовою частотою 8–16 МГц [16].

Обмеження пам'яті є критичним фактором при виборі протоколу маршрутизації. Таблиця маршрутизації IP-маршрутизатора може налічувати сотні тисяч записів, тоді як IoT-вузол з 8 КБ RAM може зберігати лише кілька десятків записів. Це обумовлює необхідність використання ієрархічних або дистанційно-векторних підходів, що мінімізують об'єм збережених даних. Протокол RPL використовує ієрархічну DODAG-структуру, яка дозволяє кожному вузлу зберігати лише інформацію про батьківський вузол (preferred parent) та обмежений набір записів у таблиці маршрутизації, що суттєво зменшує вимоги до пам'яті [1].

Характеристики радіоканалу в мережах LLN та Critical IoT суттєво відрізняються від характеристик традиційних провідних та бездротових мереж. Радіоканали стандарту IEEE 802.15.4 мають пропускну здатність лише 250 кбіт/с на фізичному рівні, при цьому ефективна пропускна здатність після врахування накладних витрат протоколів MAC та мережевого рівня складає 50–100 кбіт/с [25]. Крім того, канали є нестабільними (lossy): рівень втрат пакетів (PRR — Packet Reception Rate) може варіюватись від 50% до 99% залежно від умов поширення радіохвиль, інтерференції,

багатопроменевого розповсюдження та взаємних завад. Ця нестабільність принципово відрізняє мережі LLN від традиційних IP-мереж і накладає особливі вимоги на метрики маршрутизації.

Динаміка топології мереж Critical IoT також є суттєво вищою порівняно з провідними мережами. Вузли можуть входити у сплячий режим для економії енергії (duty cycling), переміщуватися у просторі (для мобільних застосувань), виходити з ладу внаслідок розрядки батареї або апаратних збоїв, а також зазнавати тимчасових відключень через інтерференцію. Типова частота змін топології у мережах IoT може становити від декількох подій на годину до декількох десятків подій на хвилину в умовах інтенсивної індустріальної інтерференції [11]. Протоколи маршрутизації мають швидко адаптуватись до таких змін, уникаючи при цьому генерації надмірного службового трафіку для оновлення маршрутів.

Для кількісної оцінки ресурсних обмежень IoT-вузлів у порівнянні з традиційними мережевими пристроями корисно розглянути конкретні апаратні платформи. Платформа Tmote Sky (TelosB) є класичним прикладом маломісткого вузла: мікроконтролер MSP430 (8 МГц, 16-бітний), 10 КБ RAM, 48 КБ Flash, трансивер CC2420 (IEEE 802.15.4, 250 кбіт/с), живлення від двох батарей типу AA [16]. Ця платформа широко використовується у академічних дослідженнях і є базою для оцінки ефективності протоколів маршрутизації у симуляторах. Порівняно з типовим бездротовим маршрутизатором (ARM Cortex-A9, 1 ГГц, 256 МБ RAM), ресурси Tmote Sky є у 4–5 порядків меншими за обчислювальними можливостями та у 4 порядки — за об'ємом пам'яті.

Класифікація IoT-вузлів за ресурсними можливостями, прийнята в RFC 7228 IETF, виокремлює три класи: клас 0 — вузли з дуже обмеженими ресурсами (<10 КБ RAM, <100 КБ Flash), що не здатні підтримувати повноцінний стек IPv6 без значних спрощень; клас 1 — вузли з обмеженими ресурсами (~10 КБ RAM, ~100 КБ Flash), що підтримують 6LoWPAN/CoAP, але не можуть виконувати складні обчислення в реальному часі; клас 2 —

вузли з достатніми ресурсами (>50 КБ RAM, >250 КБ Flash), здатні виконувати більшість стандартних протоколів. Більшість вузлів у мережах Critical IoT відносяться до класів 0–1 [8].

Стандарт IEEE 802.15.4-2015 визначає специфікацію фізичного та MAC-рівнів для мереж IoT з низьким енергоспоживанням. На фізичному рівні стандарт підтримує кілька частотних діапазонів: 2,4 ГГц (глобальний, 16 каналів, 250 кбіт/с), 868 МГц (Європа, 1 канал, 20/250 кбіт/с), 915 МГц (США, 10 каналів, 40/250 кбіт/с). MAC-рівень IEEE 802.15.4 підтримує два режими роботи: Beacon-enabled (синхронізовані точки доступу, Superframe Structure з гарантованими часовими слотами GTS) та Non-Beacon-enabled (асинхронний CSMA/CA без гарантій затримки). Вибір режиму MAC суттєво впливає на характеристики маршрутизації та можливість забезпечення QoS [24].

1.1.3. Вимоги до безперервності роботи

Вимоги до безперервності функціонування та надійності є визначальними характеристиками мереж Critical IoT. На відміну від побутових IoT-систем, де короткочасні перебої у роботі є прийнятними, у критичних застосуваннях мережа повинна підтримувати функціонування у визначених параметрах незалежно від зовнішніх впливів та частіше відмов окремих компонентів.

Показник доступності мережі (Network Availability) для систем Critical IoT нормується на рівні 99,9% і вище — так звані «три дев'ятки» та більше. Це означає, що сукупний час недоступності мережі не повинен перевищувати 8,76 годин на рік для 99,9%, 52,6 хвилин — для 99,99% («чотири дев'ятки»), та 5,26 хвилин — для 99,999% («п'ять дев'яток»). Для найбільш критичних застосувань, таких як ядерна енергетика або авіаційні системи, вимоги доступності можуть сягати 99,9999% [29].

Надійність доставки пакетів (PDR — Packet Delivery Ratio) є найбільш безпосередньою метрикою якості роботи мережі з точки зору прикладного рівня. PDR визначається як відношення кількості успішно доставлених пакетів до загальної кількості переданих пакетів. Для систем Critical IoT мінімально допустимий PDR зазвичай становить 95–99,9% залежно від застосування. Наприклад, для систем захисту електроенергетики клас GOOSE-повідомлень за стандартом IEC 61850-8-1 вимагає PDR не менше 99,99% при затримці до 4 мс [6].

Відмовостійкість (Fault Tolerance) мережі Critical IoT передбачає її здатність зберігати функціональність за умови виходу з ладу окремих вузлів або ділянок мережі. Кількісна оцінка відмовостійкості базується на таких показниках: коефіцієнт зв'язності мережі (Network Connectivity) — частка вузлів, що зберігають з'єднання з кореневим вузлом після відмов; час відновлення маршруту (Route Recovery Time) після виявлення відмови вузла або каналу; кількість резервних шляхів (Alternative Paths) між вузлами мережі та кореневим вузлом [11].

Ключовою вимогою до протоколу маршрутизації в контексті відмовостійкості є швидка реконвергенція після виявлення відмови. Для систем реального часу час відновлення маршруту не повинен перевищувати 100–500 мс, тоді як стандартний протокол RPL може потребувати від кількох секунд до декількох десятків секунд для перебудови DODAG після відмови важливого вузла [11]. Це є одним із ключових обмежень стандартного RPL в контексті застосувань Critical IoT та обумовлює необхідність розробки модифікованих методів маршрутизації з покращеними характеристиками відмовостійкості.

Детермінованість (Determinism) передачі даних є ще однією ключовою вимогою для ряду застосувань Critical IoT. Детермінованість означає, що для будь-якого пакету гарантується доставка у визначений часовий інтервал незалежно від поточного стану мережі та трафікового навантаження. Традиційні мережі з найкращим зусиллям (best-effort) не надають таких

гарантій. Для забезпечення детермінованості в промислових IoT-системах застосовуються спеціалізовані технології, зокрема IEEE 802.15.4 TSCH (Time Slotted Channel Hopping) у поєднанні з протоколом 6TiSCH, та технологія Time-Sensitive Networking (TSN) для провідних сегментів [10].

Особливої уваги заслуговує питання забезпечення безперервності функціонування мережі у контексті кібербезпеки. Мережі Critical IoT є привабливою ціллю для зловмисних атак, оскільки їх відмова може мати безпосередні фізичні наслідки. Типові кіберзагрози для IoT-мереж включають: атаки на відмову в обслуговуванні (DoS/DDoS) шляхом генерації надмірного службового трафіку; атаки «сивилла» (Sybil Attack), коли зловмисний вузол претендує на множинні ідентичності для впливу на вибір маршруту; атаки «чорної діри» або «сірої діри», коли скомпрометований вузол вибірково відкидає пакети [26]. Механізми автентифікації повідомлень протоколу маршрутизації (цифрові підписи DIO/DAO) та шифрування трафіку є невід'ємними компонентами захищеної реалізації RPL для Critical IoT.

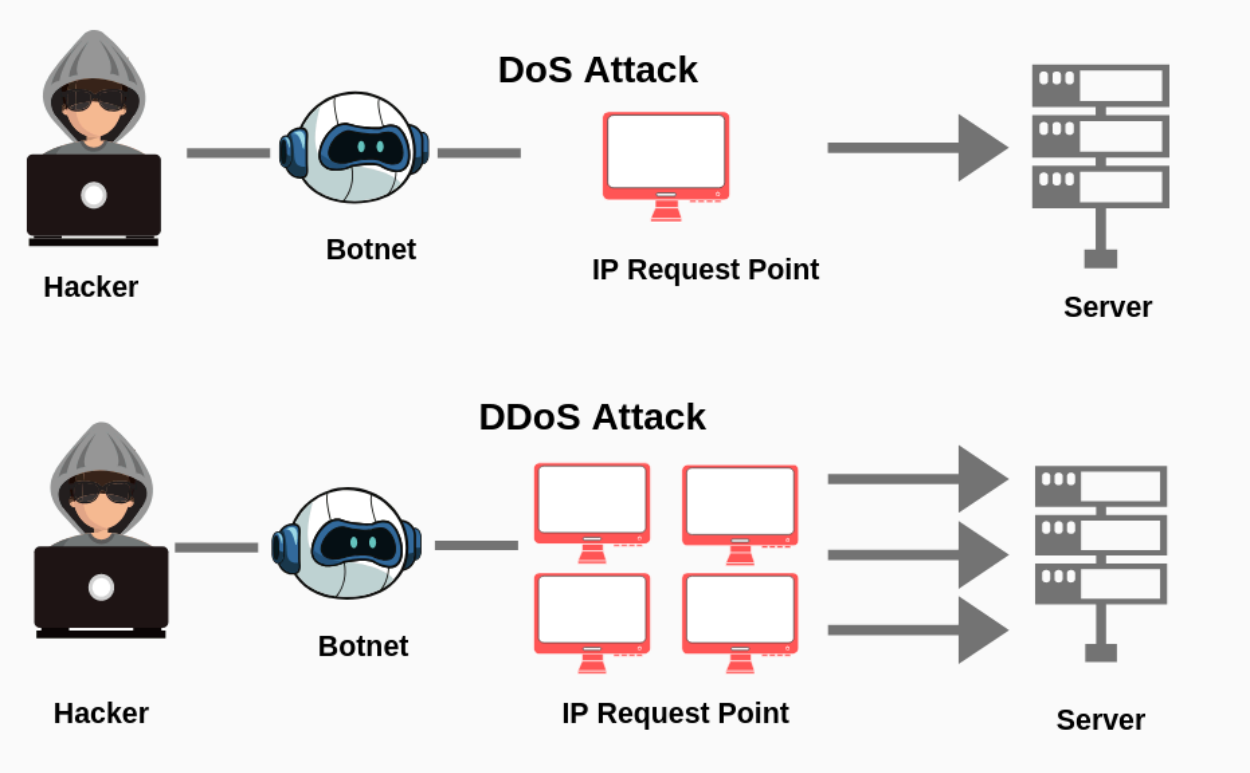

Рисунок 1.2 – Схеми DoS та DDoS атак

Квантова оцінка надійності мережі Critical IoT базується на теорії надійності телекомунікаційних систем. Мережа моделюється як граф G(V, E), де V — множина вузлів, E — множина каналів зв'язку. Надійність мережі R_G визначається як ймовірність того, що між будь-яким вузлом $v \in V$ та кореневим вузлом існує принаймні один функціонуючий шлях. При незалежних відмовах вузлів з ймовірністю q_v та каналів з ймовірністю q_e, надійність обчислюється через перебір мінімальних перетинів (min-cuts) або мінімальних шляхів (min-paths). Протокол маршрутизації безпосередньо впливає на R_G через ефективність використання надмірності мережевої топології — чим більше альтернативних маршрутів використовується, тим вища R_G [6].

1.2. Вимоги до маршрутизації в критичних застосуваннях

1.2.1. Показники надійності та затримки

Надійність доставки пакетів та кінцева затримка (end-to-end delay) є первинними метриками якості маршрутизації у мережах Critical IoT. Вони безпосередньо визначають, чи відповідає система функціональним вимогам конкретного застосування.

Показник PDR (Packet Delivery Ratio) для систем Critical IoT має бути не меншим ніж 95–99% залежно від класу застосування. При цьому важливо розрізняти PDR на рівні окремого каналу зв'язку та наскрізний PDR від кінцевого вузла до кореневого маршрутизатора. Оскільки в типовій мережі IoT пакет проходить від 3 до 8 проміжних вузлів-маршрутизаторів, наскрізний PDR є добутком PDR окремих каналів. Наприклад, якщо кожен з 5 проміжних каналів має PDR = 95%, то наскрізний $PDR = 0{,}95^5 \approx 0{,}77$ — тобто лише 77%, що є недостатнім для більшості критичних застосувань [4]. Це

підкреслює важливість вибору маршрутів з максимальним сукупним PDR, а не просто найкоротших за кількістю стрибків.

Кінцева затримка (E2E Latency) в мережах IoT складається з кількох компонентів: затримки обробки на кожному вузлі (processing delay, зазвичай < 1 мс), затримки доступу до середовища (MAC delay, 1–100 мс для IEEE 802.15.4 зі stochastic CSMA/CA), затримки поширення сигналу (propagation delay, незначна для малих відстаней) та затримки у черзі (queuing delay, що залежить від завантаження вузла). Для багатоінтервальних мереж ці складові накопичуються пропорційно до кількості стрибків, що вимагає мінімізації кількості стрибків на маршруті при збереженні достатнього рівня PDR [6].

Вимоги до кінцевої затримки варіюються залежно від класу застосування. Для систем захисту електроенергетики — не більше 4 мс; для промислової автоматизації з замкнутим контуром управління — 10–100 мс; для систем моніторингу стану обладнання (Condition Monitoring) — 100 мс – 1 с; для систем збору телеметрії — 1–10 с [29]. Протокол маршрутизації має забезпечувати вибір оптимального маршруту з урахуванням цих вимог та поточного стану мережі.

Рівномірність затримки (jitter) є ще однією важливою характеристикою, що впливає на якість передачі даних реального часу. Висока варіація затримки ускладнює синхронізацію між вузлами та порушує роботу додатків з жорсткими часовими обмеженнями. Для потоків URLLC у мережах 5G стандарт 3GPP вимагає jitter не більше 0,5–1 мс [30]. Хоча мережі LLN не можуть забезпечити настільки жорстких параметрів, мінімізація jitter є важливою вимогою до алгоритмів маршрутизації.

Математична модель наскрізного PDR у багатоінтервальній мережі дозволяє оцінити вплив якості окремих каналів на загальну надійність. Якщо маршрут від вузла v до кореня проходить через k каналів з PDR p_1, p_2, ..., p_k, то наскрізний PDR при одноразовій передачі дорівнює: $PDR_{e2e} = \prod_{i=1}^{k} p_i$. При наявності механізму ретрансмісій на рівні MAC (ARQ — Automatic Repeat reQuest) з максимальним числом спроб m, ефективний PDR одного

каналу зростає: PDR_eff = 1 – (1 – p_i)^m. Проте кожна ретрансмісія збільшує затримку та енергоспоживання. Метрика ETX (Expected Transmission Count) є зворотнім показником до PDR каналу: ETX = 1/PRR = 1/(PDR_forward × PDR_ack), де PRR — Packet Reception Rate, а PDR_ack — надійність доставки підтвердження [2].

Аналіз розподілу часових характеристик у мережах LLN з CSMA/CA показує, що затримка доступу до середовища підпорядковується складному імовірнісному розподілу, що залежить від кількості конкуруючих вузлів, коефіцієнта завантаження каналу та параметрів алгоритму розрахунку часу відступу (backoff). У стандарті IEEE 802.15.4 мінімальний розмір вікна розрахунку часу відступу (macMinBE = 3) відповідає часу від 0 до 7 слотів по 320 мкс = 0–2,24 мс на кожну спробу доступу. При максимальному числі спроб macMaxCSMABackoffs = 5, максимальна затримка доступу може досягати 8–20 мс без врахування ретрансмісій через помилки [25]. Ця варіативність затримки є причиною нестабільності часових характеристик мереж IoT та ускладнює надання гарантій QoS.

1.2.2. Пріоритезація трафіку та відмовостійкість

У мережах Critical IoT, як правило, одночасно передаються потоки різної пріоритетності. Наприклад, у промисловій мережі можуть одночасно існувати: критичні аварійні сигнали (alarm), що вимагають доставки протягом 10 мс; сигнали управління виконавчими механізмами (control), зі строком доставки 100 мс; потоки телеметрії (telemetry) з допустимою затримкою до 1 с; та потоки конфігураційних оновлень (management), що є найменш чутливими до затримки. Протокол маршрутизації має підтримувати механізми диференційованого обслуговування цих потоків.

Класифікація трафіку за пріоритетами може здійснюватись на основі таких критеріїв: тип прикладних даних (аварійний сигнал, команда управління, телеметрія); ідентифікатор джерела (вузол у зоні ризику, штатний

вузол); значення поля DSCP (Differentiated Services Code Point) у заголовку IP; значення поля Traffic Class у заголовку IPv6 [15]. Механізми пріоритизації на мережевому рівні мають взаємодіяти з механізмами QoS на рівні MAC, зокрема з можливостями TSCH для детермінованого виділення часових слотів критичним потокам [10].

Відмовостійкість маршрутизації передбачає не лише здатність системи продовжувати роботу після відмов, але й активне попередження відмов шляхом виявлення деградації каналів та проактивного переходу на резервні маршрути. Ключовими механізмами відмовостійкості є: підтримка множини батьківських вузлів (multiple parents) з автоматичним перемиканням при деградації первинного каналу; механізм захисту від петель маршрутизації на основі рангів DODAG [1]; проактивне виявлення деградації каналу за допомогою метрики ETX (Expected Transmission Count) або RSSI (Received Signal Strength Indicator).

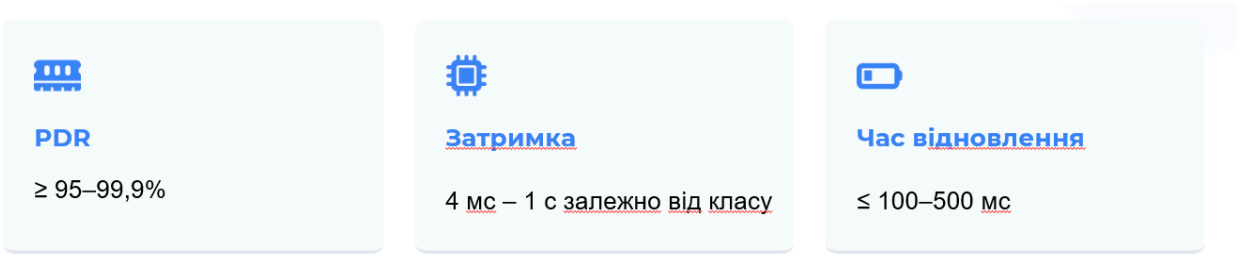

Рисунок 1.3 – Вимоги до критичного IoT

Час відновлення маршруту (Route Recovery Time — RRT) після виявлення відмови вузла або каналу є критичним параметром для систем реального часу. В стандартній реалізації RPL, після виявлення відмови батьківського вузла, вузол виконує процедуру Local Repair, що включає надсилання DIS-запитів, очікування DIO-повідомлень від сусідніх вузлів та вибір нового батьківського вузла. Ця процедура може тривати від сотень мілісекунд до декількох секунд залежно від параметрів таймерів Trickle та

щільності мережі [11]. Для систем Critical IoT такий час відновлення є неприйнятним та потребує оптимізації.

Механізми резервування маршрутів (Path Redundancy) дозволяють суттєво скоротити час відновлення шляхом підтримки заздалегідь обчислених резервних маршрутів. При цьому слід розрізняти: «гаряче» резервування (Hot Standby) — резервний маршрут підтримується у постійній готовності та може бути активований негайно; «холодне» резервування (Cold Standby) — резервний маршрут обчислюється після виявлення відмови; та адаптивне балансування навантаження (Load Balancing) — трафік розподіляється між кількома активними маршрутами пропорційно до їх якості [13].

Механізм виявлення відмов у RPL реалізований через моніторинг отримання DIO-повідомлень від батьківського вузла. Якщо протягом визначеного часового інтервалу (зазвичай кілька інтервалів Trickle) вузол не отримує DIO від свого preferred parent, він позначає батьківський вузол як недоступний та ініціює процедуру Local Repair. Тривалість виявлення відмови прямо залежить від параметрів таймера Trickle: при типових значеннях Imax = 4096 мс час виявлення може досягати 8–12 с, що є критичним для систем реального часу [1].

Підтримка декількох альтернативних батьків (Multiple Parents) в RPL є механізмом, що суттєво впливає на відмовостійкість. Кожен вузол може підтримувати список батьківських кандидатів (Parent Set), відсортованих за значенням рангу або метрики. При виявленні деградації preferred parent, вузол переходить на наступного кандидата з Parent Set без необхідності повного discovery процесу. Розмір Parent Set обмежений доступною пам'яттю вузла — зазвичай 3–5 кандидатів [2]. Оптимальне управління Parent Set, включаючи стратегії поповнення та видалення кандидатів, є важливим інструментом підвищення відмовостійкості.

1.2.3. Масштабованість та енергоефективність

Масштабованість протоколу маршрутизації є важливою вимогою, оскільки мережі Critical IoT можуть включати від десятків до тисяч вузлів. При збільшенні розміру мережі зростає обсяг службового трафіку, що генерується протоколом маршрутизації (routing overhead), і збільшується час конвергенції мережі після змін топології. Масштабована архітектура маршрутизації має забезпечувати лінійне або логарифмічне зростання накладних витрат залежно від розміру мережі.

RPL використовує механізм адаптивного таймера Trickle [RFC 6206] для керування частотою розсилки DIO-повідомлень. При стабільному стані мережі таймер Trickle автоматично збільшує інтервал між DIO-повідомленнями, що мінімізує службовий трафік. При виявленні несумісності даних (Rank inconsistency або версія DODAG) таймер скидається, і частота оновлень зростає. Цей механізм дозволяє RPL ефективно масштабуватись для мереж до кількох сотень вузлів, хоча при тисячах вузлів проблеми накладного трафіку можуть ставати значущими [4].

Енергоефективність є фундаментальною вимогою для IoT-вузлів з батарейним живленням. Споживання енергії IoT-вузла складається з кількох компонент: енергоспоживання процесора при обробці даних та виконанні алгоритмів маршрутизації; енергоспоживання радіотрансивера при передачі (TX), прийомі (RX) та прослуховуванні каналу (listen); споживання в режимі сну (sleep mode). Для типового трансивера CC2420 (IEEE 802.15.4): TX = 17,4 мА, RX = 18,8 мА, sleep = 20 мкА [16]. Оскільки прослуховування каналу споживає майже стільки ж енергії, що й прийом, мінімізація часу перебування радіо у активному стані є ключовим завданням.

Протокол маршрутизації впливає на енергоспоживання вузлів як прямо, так і побічно. Прямий вплив: обсяг службового трафіку (DIO, DIS, DAO повідомлення), що генерується вузлом, визначає час активності

радіотрансивера та, відповідно, пряме енергоспоживання протоколу. Побічний вплив: кількість ретрансмісій, зумовлених невдалими спробами передачі через нестабільні канали — вибір маршруту з вищим PDR скорочує кількість ретрансмісій та заощаджує енергію [2]. Таким чином, оптимізація маршрутизації за показником ETX опосередковано знижує енергоспоживання вузлів.

Балансування навантаження між вузлами мережі є суміжною з енергоефективністю задачею. У мережах з ієрархічною топологією (наприклад, DODAG RPL) вузли, що знаходяться поблизу кореневого маршрутизатора, виконують функцію ретрансляторів для великої кількості вузлів нижнього рівня. Це призводить до нерівномірного розподілу навантаження та прискореного розрядження батарей цих вузлів. Модифікований метод маршрутизації, що враховує поточний рівень заряду батареї та навантаження на вузол при виборі маршруту, дозволяє більш рівномірно розподіляти навантаження та подовжувати час роботи мережі [13].

Аналіз енергобюджету IoT-вузла дозволяє кількісно оцінити вплив протоколу маршрутизації на термін служби батареї. Розглянемо вузол класу 1 з батареєю ємністю 3000 мА·год, що передає телеметрію з інтервалом 60 с та виконує функції маршрутизатора для 10 дочірніх вузлів. При duty cycle 1% (активний стан 10 мс з кожних 1000 мс), середнє споживання радіоблоку: $I_{avg} = 0{,}01 \times 18\ мА + 0{,}99 \times 0{,}02\ мА \approx 0{,}2\ мА$. При додаткових витратах на обробку маршрутних повідомлень ~0,05 мА, загальне споживання ~0,25 мА, що забезпечує термін служби 3000/0,25 = 12000 годин ≈ 500 діб. Збільшення службового трафіку протоколу маршрутизації навіть на 20% скорочує термін служби до 417 діб — на 17% менше. Це підкреслює важливість мінімізації overhead протоколу маршрутизації [16].

Концепція QoS-маршрутизації в мережах Critical IoT базується на принципі диференційованого обслуговування (DiffServ), адаптованому до умов LLN. Замість складних механізмів гарантованої якості обслуговування

(IntServ, RSVP), що є надмірно ресурсоємними для IoT-вузлів, застосовується спрощена модель: трафік класифікується на 2–4 класи за пріоритетом, кожному класу надається відповідний рівень обслуговування (Priority Queuing, Weighted Fair Queuing). На рівні маршрутизації це виражається у виборі різних маршрутів для різних класів трафіку — критичний трафік спрямовується маршрутами з мінімальною затримкою та максимальним PDR, навіть якщо ці маршрути є довшими або менш енергоефективними [6].

1.3. Аналіз існуючих протоколів маршрутизації для LLN та IoT-мереж

1.3.1. Огляд альтернативних протоколів маршрутизації

Маршрутизація в мережах з низьким енергоспоживанням та втратами (LLN — Low-Power and Lossy Networks) є активно дослідженою областю, в якій запропоновано значну кількість протоколів та підходів. Розглянемо основні класи протоколів та їх представників, що є актуальними для застосування у мережах Critical IoT.

Проактивні протоколи маршрутизації заздалегідь обчислюють та підтримують маршрути до всіх вузлів мережі незалежно від наявності активного трафіку. Це забезпечує мінімальну затримку передачі після встановлення маршруту, проте вимагає постійного обміну службовими повідомленнями для підтримки актуальності таблиць маршрутизації. Основним представником цього класу в контексті IoT є протокол RPL (RFC 6550), розроблений IETF спеціально для мереж LLN [1]. Серед інших проактивних підходів слід відзначити CTP (Collection Tree Protocol) [16], що використовується в мережах TinyOS та забезпечує ефективний збір даних до кореневого вузла на основі метрики ETX.

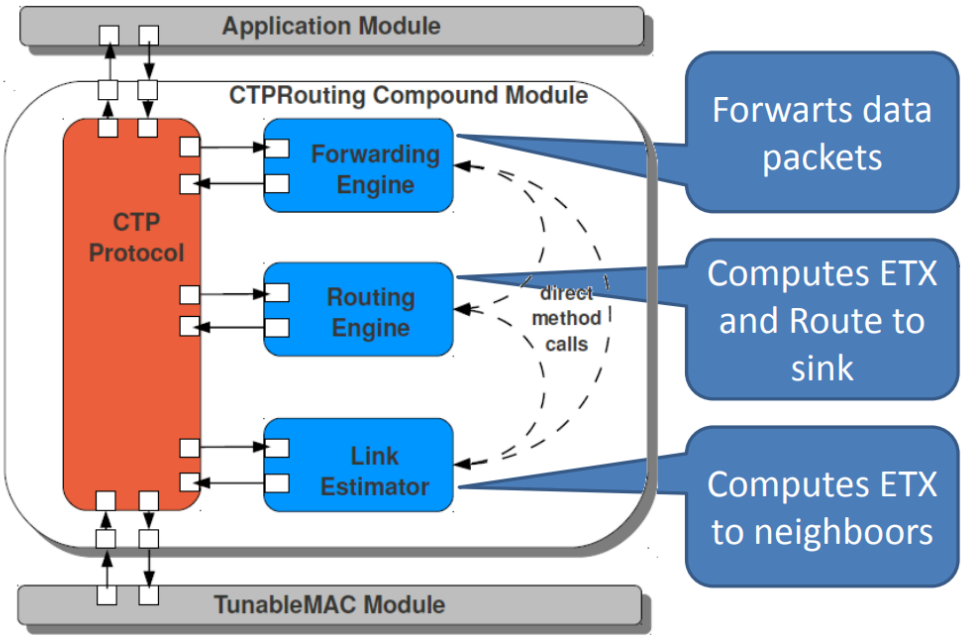

Рисунок 1.4 – Архітектура CTP

Реактивні протоколи (on-demand) встановлюють маршрут лише за наявності потреби у передачі даних. Протокол AODV (Ad hoc On-Demand Distance Vector, RFC 3561) [19] є одним із найвідоміших реактивних протоколів, розроблених для мереж MANET, але адаптованих для IoT. LOADng (Lightweight On-demand Ad hoc Distance-vector Routing Protocol — Next Generation) [20] є спрощеною версією AODV, розробленою IETF як альтернатива RPL для IoT-застосувань з невисокою щільністю трафіку. Основна перевага реактивних протоколів — мінімальний службовий трафік у стаціонарному стані; основний недолік — значна початкова затримка при встановленні маршруту (route discovery latency), що може досягати сотень мілісекунд та є неприйнятним для систем реального часу.

Гібридні протоколи та протоколи з підтримкою QoS поєднують властивості проактивних та реактивних підходів або доповнюють їх механізмами управління якістю обслуговування. Підхід Q-RPL розширює стандартний RPL підтримкою класів трафіку та механізмами пріоритизованого обслуговування черги. IETF Working Group ROLL (Routing Over Low power and Lossy networks) розробила ряд специфікацій, що описують вимоги до маршрутизації в різних застосуваннях LLN: RFC 5673

(промислові LLN), RFC 5826 (домашня автоматизація), RFC 5548 (міські мережі), RFC 5867 (медичні системи). Ці документи формують нормативну базу для проектування маршрутизації у мережах Critical IoT.

Протокол 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4, RFC 9030) [9] є комплексним рішенням для детермінованих промислових IoT-мереж. Він поєднує технологію TSCH (Time Slotted Channel Hopping) стандарту IEEE 802.15.4 на рівні MAC, протокол 6LoWPAN для адаптації IPv6 до низькоенергетичних мереж, та RPL як протокол маршрутизації. TSCH забезпечує детерміновану передачу з часовим мультиплексуванням та стрибками за каналами для підвищення завадостійкості. 6TiSCH є найбільш перспективним стандартизованим підходом для застосувань Industrial IoT з жорсткими вимогами до детермінованості [12].

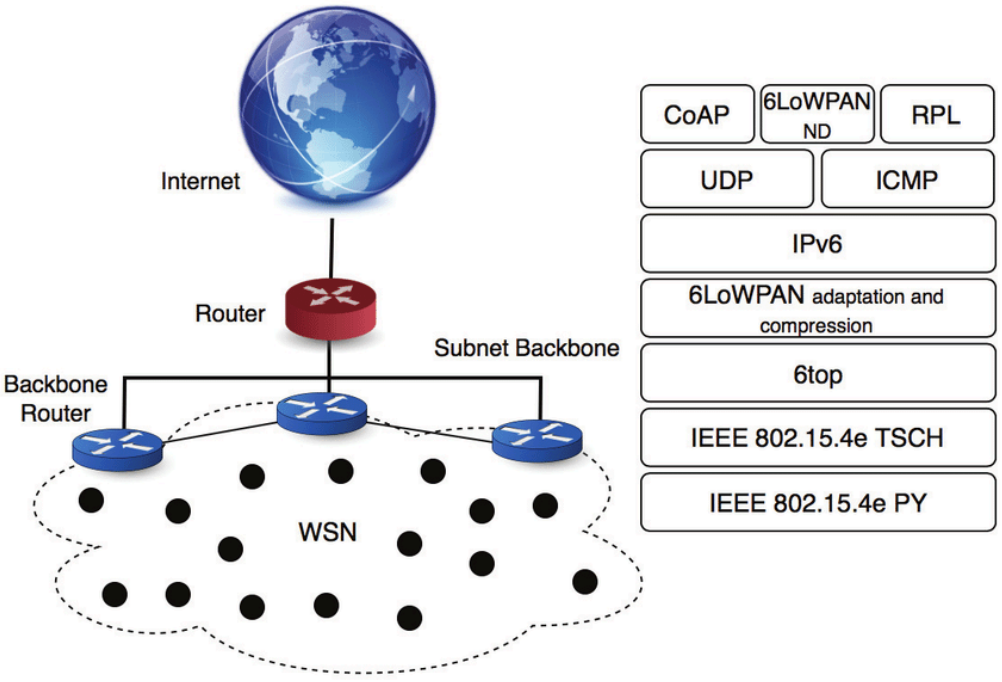

Рисунок 1.5 – Архітектура протоколу 6TiSCH

Бездротові сенсорні мережі (WSN — Wireless Sensor Networks) також мають багатий арсенал власних протоколів маршрутизації, що передують IoT-стандартам. Протокол LEACH (Low Energy Adaptive Clustering Hierarchy) забезпечує ефективне управління енергоспоживанням через ротацію

кластерних головок. PEGASIS (Power-Efficient GAthering in Sensor Information Systems) оптимізує збір даних через ланцюгову топологію. Протокол Directed Diffusion [21] реалізує підхід на основі іменованих даних (data-centric) для ефективного зіставлення запитів та відповідей у WSN. Ці підходи, хоча й не є частиною стандарту IPv6/IoT, містять цінні ідеї, що знайшли відображення у сучасних протоколах.

Окремого розгляду заслуговують протоколи маршрутизації для специфічних застосувань Critical IoT. У системах Smart Grid широко застосовується PRIME (PoweRline Intelligent Metering Evolution) та G3-PLC для маршрутизації по лініях електропередачі. Для транспортних застосувань V2X розроблено WAVE (Wireless Access in Vehicular Environments) та DSRC (Dedicated Short-Range Communications). У системах безпеки та тривоги використовуються спеціалізовані протоколи з гарантованою доставкою критичних повідомлень [22].

Для повноти огляду необхідно розглянути протоколи маршрутизації, що використовуються у традиційних бездротових сенсорних мережах (WSN), на яких базувалися ранні IoT-системи. Directed Diffusion [21] реалізує парадигму маршрутизації, орієнтованої на дані (data-centric): запити від базової станції розповсюджуються як «інтереси» (interests), а сенсорні дані повертаються вздовж «градієнтів» (gradients) найбільш ефективними маршрутами. Протокол ефективний для застосувань типу запит-відповідь, але не підтримує IPv6 та несумісний з сучасним стеком IoT.

Протоколи кластерної маршрутизації, такі як LEACH (Low Energy Adaptive Clustering Hierarchy) та його нащадки (LEACH-C, LEACH-E, TL-LEACH), організують вузли мережі в кластери з ротуючими кластерними головками (Cluster Heads). Кластерні головки виконують агрегацію даних від звичайних вузлів кластера та пряму передачу до базової станції. Перевагою є рівномірний розподіл навантаження через ротацію ролей; недоліком — необхідність синхронізації та обмеженість масштабованістю (погано масштабується для великих мереж). Ці підходи не є частиною стандарту

IPv6/IoT та мають обмежене застосування в сучасних Critical IoT-системах [17].

У контексті вимог до критичних IoT-систем важливо також розглянути підходи, що реалізовані у специфікаціях промислових бездротових стандартів. WirelessHART (IEC 62591) — розширення протоколу HART для бездротових мереж, що використовує TDMA (Time Division Multiple Access) та Channel Hopping для забезпечення детермінованої передачі. ISA100.11a (ANSI/ISA-100.11a) — стандарт для промислових бездротових мереж з підтримкою IPv6, TDMA та механізмів QoS. Обидва стандарти забезпечують детермінованість та надійність, але не є сумісними між собою, що ускладнює інтеграцію у гетерогенні мережі [13].

1.3.2. Порівняння підходів та критерії вибору протоколу

Для порівняльного аналізу протоколів маршрутизації з метою вибору найбільш підходящого для застосування у мережах Critical IoT виокремлено ключові критерії оцінки: рівень стандартизації та підтримка IETF/IEEE; ресурсна ефективність (обсяг службового трафіку, вимоги до пам'яті та обчислень); масштабованість; підтримка QoS та пріоритизації трафіку; відмовостійкість та час відновлення після відмов; сумісність з протоколами IPv6 та 6LoWPAN.

RPL (RFC 6550) є на сьогодні єдиним протоколом маршрутизації для IoT, що має статус повноцінного стандарту IETF та широку екосистему реалізацій (Contiki, RIOT OS, TinyOS, Zephyr RTOS, OpenWSN). Це є вирішальним аргументом на користь RPL при виборі протоколу для реальних промислових систем, де вимоги до сертифікації та довгострокової підтримки мають першочергове значення [8].

Порівняємо ключові характеристики провідних протоколів. AODV/LOADng демонструють низький службовий трафік у стаціонарному стані, але мають суттєву початкову затримку встановлення маршруту (до 1-5

секунд у великих мережах) та не підтримують QoS-механізми у базовій специфікації. RPL у режимі Storing Mode (Mode of Operation 2) підтримує двонаправлену маршрутизацію та ведення таблиць маршрутизації на проміжних вузлах, хоча це збільшує вимоги до пам'яті. CTP забезпечує виключно збір даних (convergecast) у напрямку кореневого вузла та не підтримує низхідну (downward) маршрутизацію, що обмежує його застосовність у системах з двонаправленим трафіком [16].

Параметр	RPL	AODV	OLSR
Енергоспоживання (мВт)	15	35	50
Затримка передачі (мс)	50 – 100	80 – 120	30 – 70
Пропускна здатність (кбіт/с)	40 – 60	40 – 80	100 – 200
Час конвергенції (с)	0,5 – 2	0,1 – 1	2 – 5
Частота оновлення (Гц)	0,2 – 1	1 – 5	2 – 10
Втрата пакетів під час мобільності (%)	10 – 20	5 – 15	20 – 40

Рисунок 1.6 – Порівняння протоколів RPL, AODV та OLSR

Спеціалізовані протоколи для промислового IoT (6TiSCH) забезпечують детермінованість та надійність, недосяжні для стандартного RPL зі стохастичним CSMA/CA. Проте їх впровадження потребує підтримки TSCH на рівні MAC-шару всіх вузлів мережі, що ускладнює та здорожує розгортання. Для мереж, де використовується стандартний IEEE 802.15.4 з CSMA/CA (без TSCH), 6TiSCH не застосовується.

Беручи до уваги наведений аналіз, RPL є найбільш обґрунтованим вибором протоколу маршрутизації для мереж Critical IoT у наступних випадках: середні та великі мережі (50–1000 вузлів) з ієрархічною топологією; системи з переважно висхідним трафіком (збір даних) та обмеженим низхідним трафіком (команди управління); мережі з вимогами до довгострокової підтримки та стандартизованої сумісності між обладнанням різних виробників; системи, що вже розгорнуті на основі IPv6/6LoWPAN та

потребують модернізації протоколу маршрутизації без заміни апаратного забезпечення.

Підводячи підсумки порівняльного аналізу, слід відзначити, що жоден з існуючих протоколів у базовій специфікації не задовольняє повністю всіх вимог мереж Critical IoT. RPL має найкращу сукупність характеристик та найширшу підтримку, але потребує розширення механізмами пріоритизації трафіку, покращеними метриками вибору маршруту та більш швидкими механізмами відновлення після відмов. Саме ця можливість розширення через архітектуру об'єктивних функцій робить RPL найбільш придатним базисом для розробки модифікованого методу маршрутизації для Critical IoT.

Критерії вибору протоколу маршрутизації для конкретного застосування Critical IoT можна систематизувати у вигляді наступного алгоритму прийняття рішень. На першому кроці визначаються ключові вимоги до часових характеристик: якщо вимагається детермінованість та затримка менше 10 мс — необхідно розглянути 6TiSCH/TSCH; якщо достатньо стохастичної доставки з затримкою 50 мс і більше — RPL з CSMA/CA є прийнятним. На другому кроці оцінюється тип трафіку: переважно висхідний збір даних або симетричний двонаправлений трафік. На третьому кроці — вимоги до масштабованості та ресурсні обмеження вузлів. На четвертому кроці — вимоги до стандартизованої сумісності між виробниками. Для більшості практичних сценаріїв Critical IoT з мережами 50–500 вузлів, вимогами PDR > 95% та затримкою 50–500 мс, RPL у поєднанні з відповідними розширеннями є оптимальним вибором [8, 13].

Висновки до розділу 1

У першому розділі проведено комплексний аналіз особливостей мереж критичного інтернету речей та вимог до маршрутизації в них. На основі цього аналізу можна зробити наступні висновки.

Критичний IoT є специфічним класом кіберфізичних систем, що характеризується поєднанням жорстких вимог до затримки (менше 10–100 мс), надійності доставки (PDR > 95–99,9%), відмовостійкості (доступність 99,9–99,9999%) та детермінованості передачі даних з обмеженими апаратними можливостями кінцевих пристроїв. Ця суперечність є ключовою конструктивною проблемою при проектуванні мереж Critical IoT.

Мережа Critical IoT має ієрархічну архітектуру, що включає кінцеві пристрої, вузли-маршрутизатори та граничні маршрутизатори. Ресурсні обмеження (4–64 КБ RAM, 250 кбіт/с пропускна здатність каналу, батарейне живлення) та нестабільність радіоканалів (PRR від 50% до 99%) є визначальними факторами, що обмежують складність алгоритмів маршрутизації.

До ключових вимог маршрутизації у мережах Critical IoT відносяться: наскрізний PDR не менше 95%; кінцева затримка від 4 мс до 1 с залежно від класу застосування; підтримка диференційованого обслуговування трафіку за пріоритетом; час відновлення після відмов не більше 100–500 мс; енергоефективність для подовження ресурсу батарей вузлів; масштабованість для мереж до кількох сотень вузлів.

Аналіз існуючих протоколів маршрутизації для LLN та IoT показав, що RPL (RFC 6550) є єдиним повністю стандартизованим IETF протоколом з широкою екосистемою реалізацій, що відповідає базовим вимогам до масштабованості та ресурсної ефективності. Реактивні протоколи (AODV, LOADng) мають неприйнятну початкову затримку; 6TiSCH забезпечує детермінованість, але вимагає підтримки TSCH на рівні MAC.

Виявлені вимоги та обмеження формують технічне завдання для розробки модифікованого методу маршрутизації на основі RPL, розгляду якого присвячені наступні розділи дипломної роботи.

РОЗДІЛ 2

ДОСЛІДЖЕННЯ ПРОТОКОЛУ RPL ТА ПОСТАНОВКА ЗАДАЧІ ЙОГО МОДИФІКАЦІЇ

2.1. Обґрунтування вибору протоколу RPL та визначення його обмежень

На основі проведеного аналізу альтернативних протоколів маршрутизації та встановлених вимог до мереж Critical IoT, RPL є найбільш перспективним кандидатом для модифікації та адаптації до умов критичних застосувань. У цьому підрозділі детально розглядаються переваги RPL, що обумовлюють його вибір, та обмеження, що потребують усунення.

2.1.1. Переваги протоколу RPL

Протокол RPL (Routing Protocol for Low-Power and Lossy Networks, RFC 6550) розроблено робочою групою ROLL IETF спеціально для вирішення задачі маршрутизації в мережах з низьким енергоспоживанням та нестабільними каналами зв'язку. Публікація RFC 6550 у 2012 році ознаменувала завершення тривалого процесу стандартизації, в якому брали участь провідні дослідницькі центри та виробники обладнання для IoT [1].

Ключовою архітектурною перевагою RPL є використання структури DODAG (Destination-Oriented Directed Acyclic Graph) — спрямованого ациклічного графу, орієнтованого на ціль. DODAG будується ітеративно від кореневого вузла (DODAG Root) до кінцевих вузлів на основі розповсюдження DIO-повідомлень. Кожному вузлу присвоюється ранг (Rank) — числовий показник, що відображає відносну позицію вузла у DODAG відносно кореня. Заборона переходів від вузлів з вищим рангом до вузлів з нижчим рангом (обмеження Rank Constraint) гарантує ациклічність маршрутів та попереджає утворення петель маршрутизації [1].

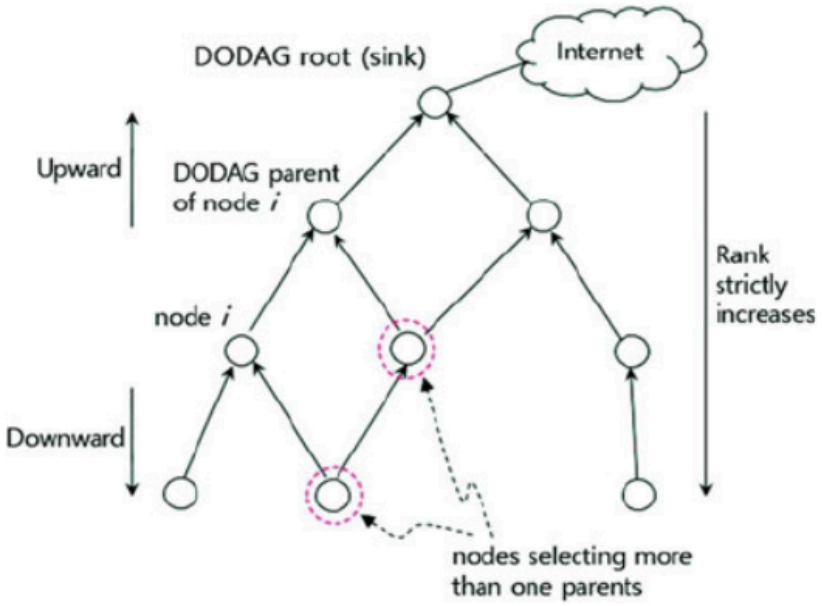

Рисунок 2.1 - Орієнтований на пункт призначення, спрямований ациклічний граф, коренем якого є приймач

Архітектура RPL підтримує три режими роботи (Mode of Operation — MOP): MOP0 (No Downward Routes Maintained) — тільки висхідна маршрутизація, без підтримки низхідних маршрутів; MOP1 (Non-Storing Mode) — низхідні маршрути підтримуються лише в кореневому вузлі у вигляді source routing; MOP2 (Storing Mode) — кожен вузол підтримує таблицю маршрутизації з записами про вузли нижнього рівня. Вибір режиму залежить від вимог застосування та ресурсних можливостей вузлів [1].

Механізм об'єктивних функцій (Objective Function — OF) є однією з найважливіших особливостей архітектурної гнучкості RPL. Об'єктивна функція визначає правило обчислення рангу вузла на основі метрик сусідніх зв'язків та вузлів, а також правило вибору батьківського вузла. Два стандартизованих варіанти об'єктивних функцій — OF0 (RFC 6552) та MRHOF (Minimum Rank with Hysteresis Objective Function, RFC 6719) — надають базові механізми вибору маршрутів. OF0 мінімізує глибину у DODAG (аналог мінімальної кількості стрибків), MRHOF мінімізує ETX з гістерезисом для запобігання нестабільному перемиканню між батьками [2,

3]. Архітектура RPL дозволяє розробляти нові об'єктивні функції для специфічних застосувань, що є ключовою точкою для модифікації у даній роботі.

Метрики маршрутизації, що використовуються RPL, стандартизовано в RFC 6551 [27]. Специфікація визначає метрики вузлів (Node Metrics): стан вузла (Node State and Attributes), кількість вузлів (Node Energy), кількість хопів (Hop Count); та метрики зв'язків (Link Metrics): пропускна здатність (Throughput), затримка (Latency), очікувана кількість передач ETX (Expected Transmission Count), якість зв'язку LQL (Link Quality Level). Ці метрики передаються у контейнері Metric Container опції DAG Metric Container DIO-повідомлень і можуть комбінуватись довільним чином об'єктивною функцією.

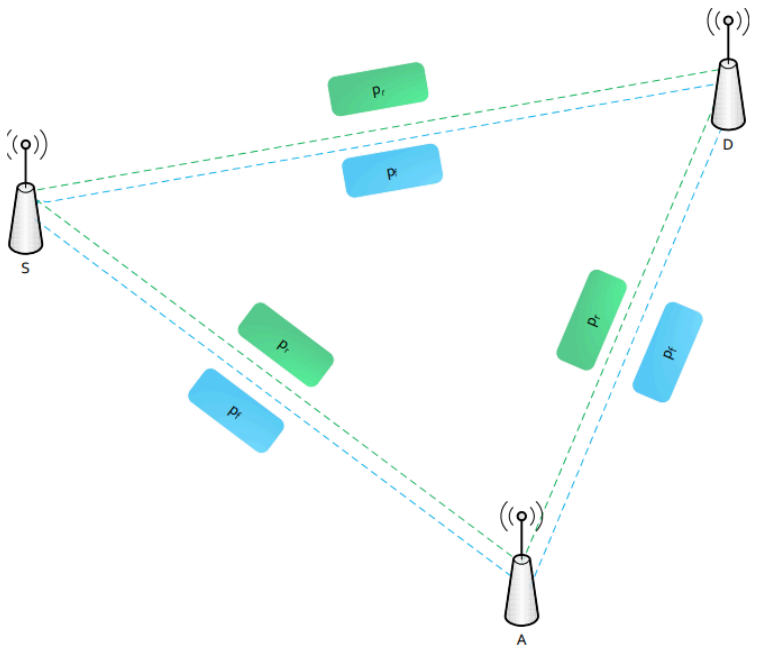

Рисунок 2.2 – Схема показниа очікуваної кількості передач

RPL широко впроваджений у провідних операційних системах для IoT. Контролер Contiki-NG містить реалізацію RPL (ContikiRPL), що є однією з найбільш зрілих та широко досліджених реалізацій протоколу у наукових роботах. RIOT OS включає власну реалізацію GNRC RPL. Zephyr RTOS —

одна з найбільш активно розвиваних вбудованих операційних систем — підтримує RPL як частину мережевого стека. OpenWSN — відкрита дослідницька платформа для 6TiSCH — також використовує RPL як протокол маршрутизації. Ця широка екосистема реалізацій та активна спільнота дослідників є важливою перевагою RPL перед альтернативними підходами [8].

Механізм Trickle-таймера (RFC 6206), що використовується RPL для керування частотою розсилки DIO-повідомлень, є важливим інструментом балансування між актуальністю інформації та мінімізацією службового трафіку. Алгоритм Trickle підтримує змінний інтервал I між DIO-повідомленнями в діапазоні [Imin, Imax]. Якщо протягом поточного інтервалу вузол прийняв більше k «узгоджених» (consistent) повідомлень від сусідів, він пропускає свою чергову розсилку DIO. При виявленні «неузгодженості» (inconsistency) — наприклад, коли інший вузол рекламує несумісний DODAG — таймер скидається до Imin і починається прискорена розсилка. Цей механізм забезпечує швидку конвергенцію після змін топології при одночасній мінімізації трафіку у стабільному стані [1].

Підтримка мод маршрутизації у RPL надає гнучкість при проектуванні мереж різної складності. У режимі Non-Storing Mode (MOP1) кожен вузол передає свої маршрутні DAO-повідомлення безпосередньо до кореня через серію одноадресних стрибків. Корінь акумулює повну топологічну інформацію та використовує маршрутизацію зі зазначенням маршруту (source routing) для низхідних пакетів, вставляючи заголовок RPL Source Route у пакети. Цей режим мінімізує вимоги до пам'яті проміжних вузлів, але збільшує розмір пакетів через заголовок маршруту. У режимі Storing Mode (MOP2) кожен проміжний вузол ведеться власна таблиця маршрутизації для вузлів нижнього рівня, що дозволяє звичайне IP-forwarding без source routing, але потребує більше пам'яті [1].

2.1.2. Недоліки RPL в умовах критичного трафіку

Незважаючи на переваги RPL як базового протоколу для IoT-мереж, при застосуванні у системах Critical IoT виявляється ряд суттєвих обмежень, що потребують усунення шляхом розробки модифікованих методів маршрутизації.

Першим та найбільш критичним обмеженням є відсутність підтримки диференційованого обслуговування трафіку за пріоритетом. Стандартний RPL не розрізняє пакети різних класів трафіку при виборі маршруту — пакети аварійного сигналу та пакети фонової телеметрії обробляються ідентично. Це означає, що у ситуації перевантаження вузла або конкуренції за доступ до середовища критичний трафік не отримує переваги у доставці. Дослідження показують, що навіть при середньому навантаженні мережі 30–50% від максимальної пропускної здатності, відсутність пріоритизації призводить до зростання затримки критичних пакетів у 2–5 разів порівняно з некритичними потоками [11].

Другим обмеженням є алгоритм вибору батьківського вузла у стандартних об'єктивних функціях. MRHOF вибирає батька виключно на основі ETX (або іншої однієї метрики), не враховуючи поточне завантаження вузла (node load), рівень заряду батареї (energy level) або кількість вузлів, що вже використовують цей вузол як батька. Це призводить до концентрації трафіку на окремих «популярних» вузлах з найкращим ETX, що прискорює їх розрядку та збільшує ризик відмови саме тих вузлів, через які проходить найбільший обсяг трафіку [4].

Третім обмеженням є повільна реконвергенція після відмов. Стандартний RPL використовує механізм Local Repair, який після виявлення відмови батьківського вузла надсилає DIS-запити та очікує отримання DIO-повідомлень для вибору нового батька. Параметри таймера Trickle за замовчуванням (Imin = 8 мс, Imax = 4096 мс, k = 10 для RFC 6550) в адаптованих реалізаціях часто встановлюються в діапазоні від 1 с до 30 хв,

що призводить до часу відновлення від сотень мілісекунд до хвилин — неприйнятний показник для Critical IoT [11].

Четвертим обмеженням є нестабільність маршрутів (Route Instability або Routing Churn). При незначних коливаннях якості каналів вузли можуть перемикатись між батьківськими вузлами, що генерує додатковий службовий трафік та викликає тимчасові петлі або розриви маршрутів. Хоча MRHOF вводить механізм гістерезису (порогове значення PARENT_SWITCH_THRESHOLD) для зниження нестабільності, налаштування цього параметру є нетривіальним завданням — занадто малий поріг призводить до частих перемикань, занадто великий — до збереження неоптимальних маршрутів [2].

П'ятим обмеженням є обмежена підтримка балансування навантаження. Стандартний RPL вибирає єдиний preferred parent та спрямовує весь висхідний трафік через нього. Хоча специфікація допускає наявність резервних батьків (backup parents), переключення на них відбувається лише після виявлення відмови, а не проактивно для рівномірного розподілу навантаження. У результаті вузли поблизу кореня DODAG перевантажуються, тоді як альтернативні шляхи залишаються незадіяними [4, 13].

Шостим обмеженням є метрика ETX як єдиний стандартний критерій якості каналу. ETX — очікувана кількість передач для успішної доставки пакету — є широко використовуваною та добре дослідженою метрикою, але вона не відображає миттєвий стан каналу (миттєву затримку, джитер), не враховує пропускну здатність каналу та не диференціює типи помилок (постійні vs. тимчасові). У нестабільних каналах IoT-середовища ETX є усередненим показником, що повільно реагує на різкі зміни стану каналу [2].

Наведені обмеження стандартного протоколу RPL обумовлюють необхідність розробки модифікованого методу маршрутизації, що усуває або пом'якшує ці недоліки при збереженні базової сумісності з RPL та підтримці

широкої екосистеми існуючих реалізацій. Конкретні вимоги до модифікованого методу будуть сформульовані у розділі 2.

Детальний аналіз поведінки RPL в умовах деградації каналів зв'язку виявляє ще одну проблему — явище «ефекту зграї» (herding effect). Коли певний вузол-маршрутизатор покращує свій ETX внаслідок тимчасового поліпшення умов поширення радіохвиль, велика кількість сусідніх вузлів одночасно перемикається на нього як на батьківський. Це різко збільшує навантаження на вузол, що в умовах обмеженої пропускної здатності IEEE 802.15.4 призводить до зростання черг та часу очікування, а отже — до погіршення ETX цього вузла. Вузли знову починають перемикатись, і цикл повторюється. Такий режим роботи є нестабільним та призводить до значного збільшення затримки та втрат пакетів [4].

Проблема асиметрії каналів в RPL також заслуговує на увагу. Метрика ETX обчислюється як 1/(PDR_forward × PDR_ack), і для її розрахунку вузол повинен знати як PRR у прямому (від сусіда до себе), так і у зворотному (від себе до сусіда) напрямку. В реальних бездротових мережах ці значення можуть суттєво відрізнятись через асиметрію антен, різницю в потужностях передавачів та взаємні завади. Якщо реалізація RPL не враховує асиметрію, ETX може бути занижений, що призводить до вибору фактично гіршого маршруту [11].

Ще одним важливим обмеженням є проблема «чорних дір» (Black Holes) та «сірих дір» (Gray Holes) у RPL-мережах. Вузол-«чорна діра» отримує пакети від сусідів, але не пересилає їх далі (наприклад, через переповнення черги або збій програмного забезпечення), при цьому продовжуючи розсилати DIO-повідомлення з нормальними метриками. Сусідні вузли не мають механізму виявлення такої ситуації в стандартному RPL, крім пасивного спостереження за рівнем втрат. Час виявлення «чорної діри» може становити кілька інтервалів Trickle, протягом яких значна частина трафіку буде втрачена [11].

Дослідження, проведені на основі симулятора Cooja [16] та тестових стендів з реальними вузлами Tmote Sky, показують, що стандартний RPL з MRHOF досягає PDR 85–95% у стабільних мережах з 20–50 вузлами, але в умовах нестабільних каналів та відмов вузлів PDR може знижуватись до 60–80%. При цьому середня затримка зростає від 100–200 мс до 500 мс – 2 с при інтенсивних відмовах [4]. Ці показники є неприйнятними для більшості застосувань Critical IoT та підтверджують необхідність розробки модифікованого методу маршрутизації.

Для кількісної ілюстрації обмежень стандартного RPL розглянемо типовий сценарій промислової IoT-мережі: 50 вузлів, рівномірно розподілених у виробничому цеху площею 50×50 м, дальність зв'язку 15 м, IEEE 802.15.4 CSMA/CA, Contiki-NG з ContikiRPL/MRHOF. Симуляція у Cooja показує, що при відмові 3 вузлів у центральній частині DODAG (безпосередньо під кореневим маршрутизатором) PDR знижується до 72%, середня затримка зростає до 1,8 с, і процедура відновлення займає в середньому 12,4 с [4]. Ці результати безпосередньо демонструють необхідність розробки модифікованого підходу, що є предметом даної дипломної роботи.

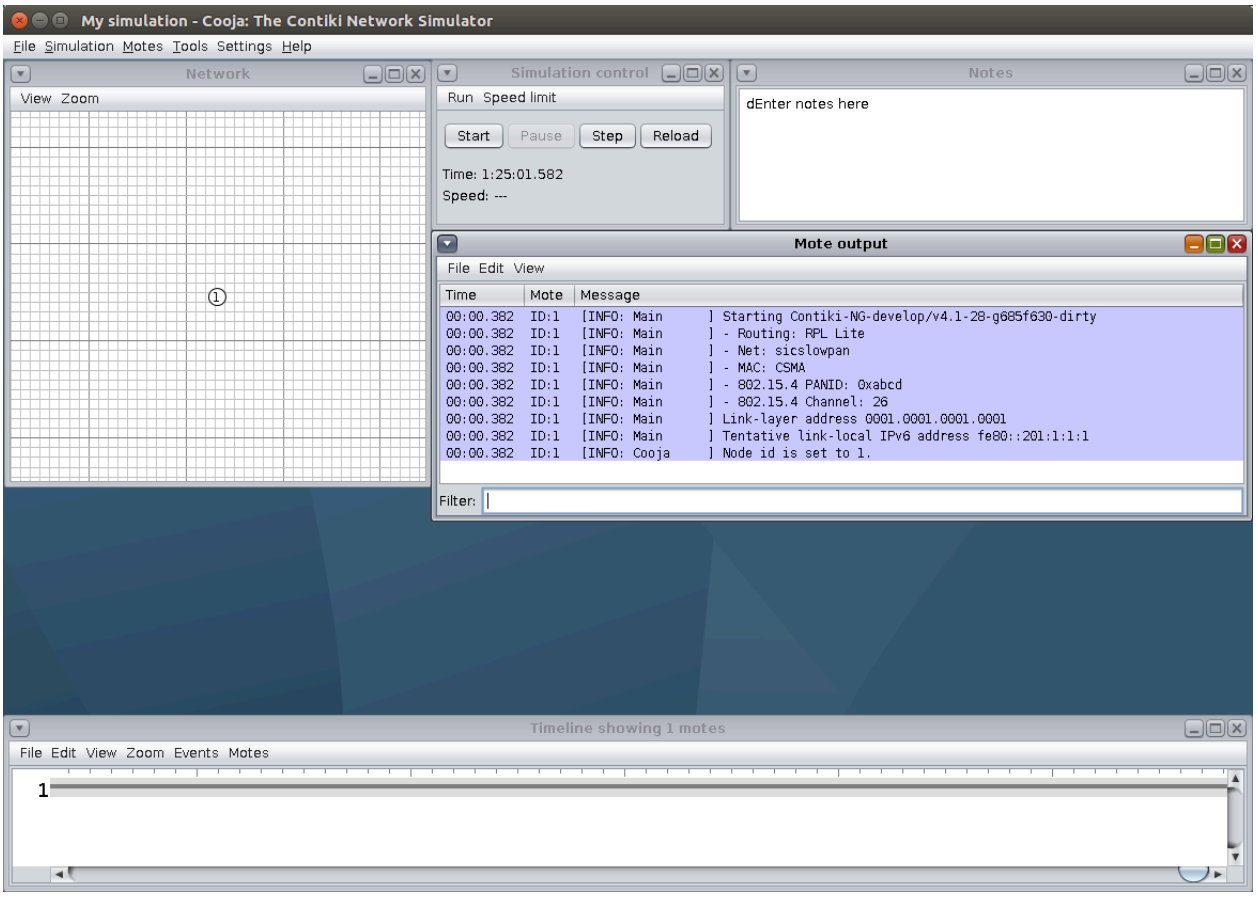

Рисунок 2.3 – Вікно застосунку Cooja

Узагальнюючи обмеження стандартного RPL, можна стверджувати, що цей протокол, будучи ефективним рішенням для загального IoT, потребує суттєвого доопрацювання для застосування у системах Critical IoT. Ключовими напрямками модифікації є: введення підтримки пріоритетності трафіку на рівні об'єктивної функції; розробка комбінованої метрики вибору маршруту, що враховує ETX, навантаження вузла та рівень заряду батареї; реалізація механізму проактивного формування резервних маршрутів та швидкого перемикання на них при виявленні деградації; налаштування параметрів таймера Trickle для скорочення часу виявлення та відновлення після відмов. Реалізація цих модифікацій дозволить підвищити PDR до рівня не менше 95% та скоротити час відновлення після відмов до 100–200 мс.

2.2. Архітектура та принципи роботи протоколу RPL

Протокол RPL було розроблено робочою групою ROLL IETF як спеціалізований механізм маршрутизації для мереж класу LLN, у яких поєднуються низька пропускна здатність каналів, обмежені ресурси вузлів та нестабільний характер бездротового зв'язку. На відміну від класичних IP-протоколів, орієнтованих на відносно потужні маршрутизатори та передбачувані канали, RPL від початку проєктувався для топологій, де вузол може мати кілька слабких сусідів, а вартість службового трафіку безпосередньо впливає на термін служби батарей і надійність доставки пакетів [1, 18, 24].

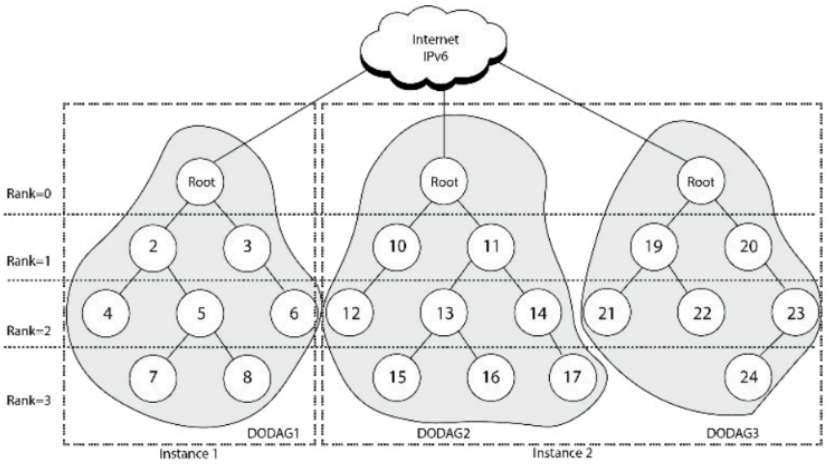

Рисунок 2.4 – Топологія протоколу RPL

Архітектура RPL базується на побудові орієнтованого ациклічного графа DODAG, у межах якого більшість обміну організовується відносно кореневого маршрутизатора. Такий підхід добре узгоджується з типовою моделлю трафіку в мережах критичного інтернету речей, де переважає передавання телеметрії від сенсорних вузлів до шлюзу або центру обробки. Водночас специфіка керування DODAG, правила вибору батьківського вузла та механізми оновлення маршрутної інформації мають суттєвий вплив на

масштабованість, затримку та стійкість мережі до відмов, що робить детальний розгляд архітектури RPL необхідною передумовою для подальшої модифікації протоколу [1, 11].

2.2.1. Побудова DODAG та поняття рангу

Базовим структурним елементом RPL є DODAG (Destination-Oriented Directed Acyclic Graph) — спрямований ациклічний граф, зорієнтований на певну ціль, якою у більшості практичних застосувань виступає граничний маршрутизатор або шлюз мережі. Кореневий вузол ініціює побудову DODAG шляхом розсилки повідомлень DIO, які містять ідентифікатор екземпляра RPL, інформацію про режим роботи, параметри об'єктивної функції та контейнер метрик. Вузли, що прийняли DIO і визнали його придатним, обчислюють власний ранг, вибирають батьківський вузол та, у свою чергу, рекламують свій стан сусідам, завдяки чому граф формується ітеративно від кореня до периферії мережі [1, 3].

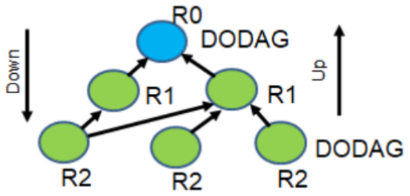

Рисунок 2.5 – Структура DODAG

Поняття рангу у RPL не зводиться лише до кількості стрибків до кореня. Ранг є відносною характеристикою положення вузла в DODAG, яка повинна монотонно збільшуватися при віддаленні від кореня та обчислюється відповідно до обраної об'єктивної функції. За рахунок цього ранг одночасно виконує дві функції: по-перше, забезпечує логічне впорядкування вузлів у графі, а по-друге, запобігає утворенню петель маршрутизації, оскільки

пересилання трафіку дозволяється лише у напрямку вузлів з меншим рангом. Для мереж Critical IoT це особливо важливо, адже навіть короткочасна петля призводить до перевитрати енергії, зростання затримки та втрати частини пакетів у перевантажених вузлах [1, 2].

Разом із перевагами ранговий підхід має і методичне обмеження: сам по собі ранг не відображає повної якості маршруту з погляду затримки, завантаження черг або залишкового енергоресурсу. Внаслідок цього два вузли з близькими рангами можуть суттєво відрізнятися за реальними можливостями обслуговувати критичний трафік. Саме ця розбіжність між формально коректною структурою DODAG та фактичною якістю обслуговування у складних сценаріях експлуатації є однією з причин, чому стандартний механізм ранжування потребує розширення в рамках модифікованого методу маршрутизації [2, 11].

2.2.2. Висхідні та низхідні маршрути

У RPL принципово розрізняють висхідну та низхідну маршрутизацію. Висхідний трафік формується від кінцевих пристроїв або проміжних маршрутизаторів у напрямку до кореня DODAG і є природним для більшості телеметричних систем, де датчики періодично надсилають вимірювання до центрального вузла. Саме цей тип трафіку підтримується найбільш ефективно, оскільки кожному вузлу достатньо знати preferred parent та, за потреби, кілька резервних кандидатів, що мінімізує вимоги до пам'яті та спрощує процедуру обслуговування маршрутів [1, 8].

Низхідна маршрутизація використовується для передавання команд керування, параметрів конфігурації або службових повідомлень від кореня до підлеглих вузлів. Її реалізація у RPL можлива у двох базових підходах: storing mode, коли проміжні вузли зберігають інформацію про піддерева та виконують звичайне IP-forwarding, і non-storing mode, коли повна інформація

про шляхи концентрується у корені, а самі пакети передаються з використанням source routing. Перший підхід зменшує розмір заголовків у низхідних пакетах, але підвищує вимоги до пам'яті проміжних вузлів; другий, навпаки, переносить основне навантаження на кореневий вузол і збільшує накладні витрати на передавання маршрутної інформації в пакеті [1, 28].

Для систем Critical IoT значущість низхідних маршрутів не менша, ніж висхідних, оскільки саме вони забезпечують доставку керувальних впливів, аварійних команд і процедур відновлення після збоїв. Якщо висхідна маршрутизація переважно впливає на своєчасність надходження телеметрії, то якість низхідних шляхів визначає можливість оперативного керування об'єктом. Тому в подальшому під час формування вимог до модифікації RPL необхідно враховувати не лише ефективність збору даних, а й коректність підтримки двоспрямованої взаємодії з вузлами мережі [1, 11, 28].

2.2.3. Призначення повідомлень DIO, DIS, DAO

Інформаційний обмін у RPL базується на трьох основних типах службових повідомлень: DIO, DIS і DAO. Повідомлення DIO (DODAG Information Object) використовується для поширення параметрів екземпляра RPL, поточного рангу вузла, режиму роботи та метрик, необхідних для приєднання нових учасників до DODAG. Саме DIO є основним носієм керувальної інформації, на підставі якої вузли оцінюють придатність сусідів і будують висхідну частину графа [1, 14].

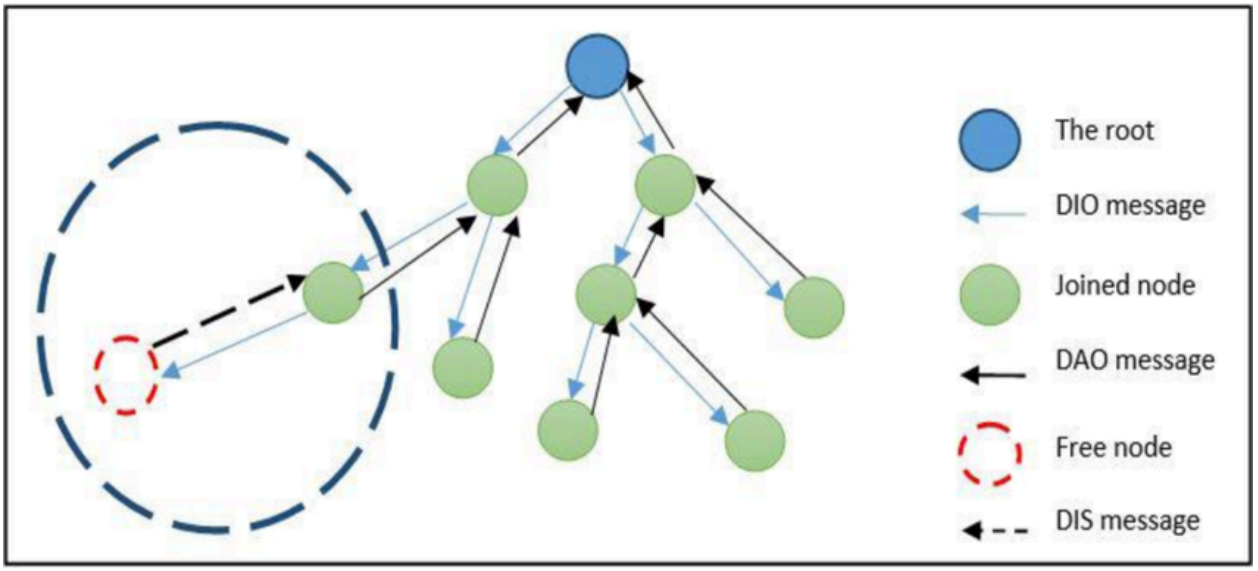

Рисунок 2.6 – Схема передачі повідомлення DIO, DAO та DIS у DODAG.

Повідомлення DIS (DODAG Information Solicitation) відіграє запитальну роль. Воно передається вузлом, який ще не має достатньої інформації для входження до DODAG або виявив ознаки деградації поточного маршруту та бажає прискорити отримання актуальних DIO від сусідів. DAO (Destination Advertisement Object), своєю чергою, використовується для реклами досяжності вузла у низхідному напрямку. Через DAO формується уявлення про піддерева DODAG, що є основою для підтримки маршрутів від кореня до периферійних вузлів [1, 28].

Частота та умови розсилки DIO безпосередньо пов'язані з алгоритмом Trickle, який дозволяє зменшити службовий трафік у стабільному стані й одночасно прискорити реакцію мережі на виявлену неузгодженість. У цьому полягає одна з ключових переваг RPL: протокол не підтримує постійно високий темп керувального обміну, а адаптує його до фактичної динаміки топології. Проте для критичних застосувань цей компроміс не завжди є достатнім, оскільки зменшення overhead у спокійному режимі може супроводжуватися надто повільною реакцією на локальну відмову чи швидку деградацію каналу. Саме тому механізми DIO, DIS і DAO мають розглядатися

не ізольовано, а як частина загальної задачі забезпечення своєчасного переналаштування маршрутів [1, 11].

2.3. Об'єктивні функції та метрики маршрутизації у RPL

Об'єктивна функція у RPL визначає, яким чином вузол інтерпретує інформацію про сусідів, обчислює власний ранг і вибирає preferred parent. Саме на рівні objective function стандартний RPL надає основну точку розширення для адаптації протоколу під конкретні умови експлуатації. Завдяки цьому одна і та сама архітектура DODAG може використовуватися як у відносно простих телеметричних мережах, так і в більш вимогливих індустріальних сценаріях, хоча реальна ефективність такого використання повністю залежить від того, які метрики закладено у правило вибору маршруту [1, 14].

У контексті Critical IoT питання вибору об'єктивної функції набуває центрального значення, оскільки саме вона визначає компроміс між надійністю доставки, часовими характеристиками, рівномірністю використання вузлів і витратами енергії. Якщо функція орієнтована лише на одну метрику, наприклад ETX або кількість стрибків, мережа може демонструвати прийнятну поведінку в середньому, але втрачати стійкість у кризових сценаріях. Тому аналіз стандартних objective functions OF0 і MRHOF є необхідним кроком для обґрунтування подальшої багатокритеріальної модифікації протоколу [2, 3, 11].

2.3.1. Об'єктивні функції OF0 і MRHOF

OF0 (Objective Function Zero) є базовою стандартизованою об'єктивною функцією RPL і орієнтована переважно на мінімізацію відстані до кореня у

термінах рангу. Практично це означає, що при виборі батьківського вузла OF0 віддає перевагу шляху з меншим приростом рангу, що часто корелює з меншою кількістю стрибків. Такий підхід є привабливим з точки зору простоти реалізації та низьких обчислювальних витрат, однак він слабко відображає якість каналу, а отже в умовах нестабільного радіосередовища може формувати маршрути, які є короткими формально, але дорогими з позиції повторних передавань і втрат пакетів [3, 14].

MRHOF (Minimum Rank with Hysteresis Objective Function), на відміну від OF0, будує правило вибору preferred parent навколо мінімізації акумульованої метрики шляху, найчастіше ETX. Завдяки механізму гістерезису MRHOF не перемикає вузол на нового батька при кожному незначному покращенні метрики, а вимагає досягнення порогової переваги, що знижує маршрутну нестабільність. Саме тому MRHOF є найпоширенішою об'єктивною функцією у практичних і дослідницьких реалізаціях RPL для IoT [2, 11].

Попри це MRHOF не вирішує проблеми однофакторності маршрутизації. Якщо як основний критерій використовується ETX, то вузол із найкращою якістю каналу, але переповненими чергами або критично низьким енергоресурсом, однаково залишатиметься привабливим кандидатом. Для мереж критичного інтернету речей така поведінка є недостатньою, оскільки якість обслуговування визначається не лише ймовірністю успішної передачі на одному каналі, а й часовою реакцією маршруту в цілому, стійкістю до перевантаження та здатністю мережі переживати локальні відмови без втрати керованості [2, 4, 11].

2.3.2. Метрики ETX, затримка, енергетичні характеристики

Найбільш уживаною метрикою в RPL є ETX (Expected Transmission Count) — очікувана кількість передавань, необхідних для успішної доставки

пакета по каналу з урахуванням можливих повторних спроб. Її цінність полягає в тому, що ETX безпосередньо пов'язана з надійністю фізичного каналу та непрямо відображає витрати енергії: чим більше повторних передавань, тим вищі навантаження на радіоінтерфейс і довший час обслуговування трафіку. Саме тому ETX є природним вибором для LLN-мереж, де помилки каналу мають системний, а не випадковий характер [14, 16].

Однак для задач Critical IoT метрика ETX є необхідною, але недостатньою. Вона не відображає поточну затримку обслуговування, довжину локальних черг, джитер та чутливість маршруту до перевантаження вузлів поблизу кореня. Два маршрути з однаковим ETX можуть суттєво відрізнятися за середньою та піковою затримкою, якщо один проходить через перевантажений проміжний вузол, а інший — через менш завантажений, але трохи гірший за якістю радіоканал. Для промислових і керувальних застосувань така різниця часто важливіша за мінімальну кількість ретрансляцій [4, 11, 13].

Енергетичні характеристики також повинні розглядатися як повноцінний критерій маршрутизації. Якщо протокол систематично концентрує трафік на обмеженій множині вузлів із хорошим ETX, саме ці вузли починають втрачати енергію швидше за інших, після чого мережа переходить у фазу каскадної деградації: спочатку зменшується ресурс найважливіших ретрансляторів, далі зростає кількість перебудов DODAG, а потім погіршується і загальний PDR. Отже, включення показників енергетичного стану або принаймні індикаторів навантаження в об'єктивну функцію є логічним продовженням стандартного підходу, а не його запереченням [11, 13, 14].

2.3.3. Правила вибору батьківського вузла

Процедура вибору preferred parent у RPL ґрунтується на локальному порівнянні доступних кандидатів, для яких вузол має достатньо інформації про ранг, метрики та сумісність параметрів екземпляра RPL. У стандартному випадку вузол формує список сусідів-кандидатів, відкидає тих, хто порушує рангові обмеження або не відповідає режиму роботи DODAG, а потім ранжує решту згідно з об'єктивною функцією. Після вибору preferred parent вузол може зберігати й альтернативних батьків, але фактичне пересилання висхідного трафіку, як правило, відбувається через один маршрут [1, 2].

Таке правило є достатньо ефективним для статичних або помірно динамічних мереж, проте в умовах критичного трафіку воно породжує низку ризиків. По-перше, одноосібний preferred parent концентрує трафік і створює вузькі місця поблизу кореня. По-друге, локальне порівняння за однією метрикою не дозволяє відрізнити справді стійкий маршрут від тимчасово «вигідного» сусіда, який уже перебуває на межі перевантаження. По-третє, навіть наявність гістерезису не усуває повністю явища route churn, якщо зміни якості каналів мають коливальний характер [2, 4].

Звідси випливає важливий практичний висновок: для Critical IoT правило вибору батьківського вузла має бути не просто механізмом мінімізації поточної вартості маршруту, а засобом підтримання глобальної стійкості DODAG. Це передбачає використання додаткових індикаторів стану вузла, вагового урахування класу трафіку, а також обмеження на надмірно часте перемикання між кандидатами. Отже, саме модифікація правила parent selection є центральним елементом майбутнього вдосконалення RPL, оскільки через неї можна безпосередньо вплинути і на PDR, і на затримку, і на відмовостійкість мережі [4, 11, 13].

2.4. Аналіз поведінки RPL у мережах критичного інтернету речей

Після розгляду базової архітектури та логіки вибору маршруту необхідно перейти від формальної специфікації RPL до аналізу його фактичної поведінки у сценаріях, характерних для Critical IoT. Теоретично DODAG-структура та стандартні objective functions забезпечують працездатну маршрутизацію у мережах LLN, однак у критичних застосуваннях важливими стають не лише середні значення метрик, а і поведінка системи в пікових режимах, під час відмов, деградації каналів та конфлікту між потоками різної пріоритетності. Саме в цих умовах проявляються структурні обмеження стандартного RPL [4, 11, 13].

Для подальшої постановки задачі модифікації принципово важливо відокремити дві групи проблем. Перша група пов'язана з поступовим погіршенням якості обслуговування: зростанням затримки, джитера, частоти ретрансляцій і службового трафіку. Друга група відноситься до подій різкої деградації топології, коли мережа втрачає частину вузлів або критичні радіоканали і повинна швидко перебудуватися без довгого періоду нестабільності. У реальній мережі ці групи тісно пов'язані між собою: хронічне перевантаження збільшує ймовірність відмов, а відмови, у свою чергу, додатково підсилюють перевантаження сусідніх маршрутів [4, 13].

2.4.1. Сценарії деградації якості обслуговування

Одним із типових сценаріїв деградації QoS у RPL-мережі є поступове зростання навантаження без явної фізичної відмови вузлів. У такій ситуації preferred parent, обраний за критерієм ETX, може залишатися формально найкращим маршрутом, хоча фактична затримка в ньому вже перевищує прийнятний рівень через накопичення пакетів у чергах і повторні спроби доступу до середовища IEEE 802.15.4. Оскільки стандартна об'єктивна функція не відстежує динаміку черг у реальному часі, RPL продовжує

використовувати перевантажений шлях довше, ніж це допустимо для критичного трафіку [4, 11].

Інший характерний сценарій пов'язаний із коливанням якості радіоканалів. Якщо канал короткочасно покращується, частина вузлів може майже синхронно перемкнутися на одного і того самого сусіда, що знижує баланс навантаження та створює ефект «зграї». Після перевантаження цього вузла його ETX знову погіршується, унаслідок чого відбувається зворотне перемикання. З погляду мережевого рівня така поведінка проявляється як route churn — серія перебудов маршруту без реального виграшу в якості обслуговування. Для Critical IoT це означає не просто погіршення середніх показників, а втрату передбачуваності сервісу, що є критичною вадою для систем керування реального часу [2, 4].

Погіршення якості обслуговування також посилюється змішаним характером трафіку. Коли в одній і тій самій мережі співіснують аварійні повідомлення, періодична телеметрія та сервісні потоки, відсутність механізму пріоритезації на рівні маршрутизації призводить до того, що критичні пакети конкурують за однакові ресурси з фоновими потоками. Навіть якщо MAC-рівень частково підтримує різні режими доступу, без урахування пріоритету при виборі маршруту мережа не здатна гарантувати, що найбільш важливі пакети підуть через шлях з найменшим ризиком черги або втрати [11, 13].

2.4.2. Вплив перевантаження та втрати вузлів

Перевантаження у RPL зазвичай має не рівномірний, а локалізований характер: найбільше навантаження концентрується на вузлах, розташованих ближче до кореня DODAG, оскільки саме через них проходить агрегований трафік від значної частини мережі. Якщо objective function орієнтується переважно на ETX, то ці вузли залишаються preferred parent для багатьох нащадків навіть тоді, коли фактичний час обслуговування пакета починає різко зростати. У підсумку мережа втрачає рівномірність використання ресурсів: частина вузлів стає перевантаженою, тоді як інші потенційні маршрути недовикористовуються [4, 13].

Втрати вузлів або ключових каналів мають ще серйозніші наслідки. Коли виходить з ладу проміжний маршрутизатор, його нащадки змушені виконувати локальний ремонт, ініціювати DIS-запити, очікувати DIO від сусідів і повторно обирати батьківські вузли. У теорії цей механізм дозволяє мережі відновити зв'язність без повної перебудови DODAG, але на практиці час реконвергенції часто виявляється надто великим для критичних застосувань. Поки триває перебудова, пакети накопичуються в чергах або втрачаються, а сусідні вузли зазнають додаткового службового навантаження [1, 11, 16].

Особливо небезпечним є поєднання перевантаження з частковими відмовами. Після втрати одного з центральних вузлів трафік перенаправляється через сусідні маршрути, які самі по собі вже можуть працювати поблизу межі пропускної здатності. У таких умовах навіть коректна локальна перебудова RPL не гарантує прийнятного QoS: мережа відновлює формальну досяжність, але не забезпечує цільових значень PDR,

затримки та стабільності маршруту. Це означає, що для Critical IoT недостатньо просто мати механізм repair; потрібен механізм, який ще до відмови підтримує керований запас стійкості та резервування маршрутів [4, 11, 13].

2.5. Формування вимог до модифікованого методу маршрутизації

Проведений аналіз показує, що стандартний RPL забезпечує структурно коректну маршрутизацію в мережах LLN, але не покриває повний набір вимог, характерних для Critical IoT. З одного боку, протокол має сильні сторони: сумісність з IPv6/6LoWPAN, формалізовану модель керування DODAG, стандартизовані objective functions і широку екосистему реалізацій. З іншого боку, однофакторний вибір маршруту, слабке врахування поточного навантаження та відсутність вбудованої орієнтації на пріоритетність трафіку обмежують можливість застосування стандартного RPL у системах із жорсткими вимогами до QoS [1, 11, 14].

Отже, постановка задачі модифікації має спиратися не на відмову від базових механізмів RPL, а на їх цільове розширення. Модифікований метод повинен зберегти логіку DODAG, сумісність із службовими повідомленнями DIO/DIS/DAO та придатність до реалізації на ресурсно обмежених вузлах. Водночас він має усунути або принаймні істотно послабити ті фактори, які виявилися критичними для якості обслуговування: перевантаження preferred parent, повільну реакцію на локальні відмови, нестабільність маршруту та відсутність урахування класу трафіку [1, 4, 11].

2.5.1. Цільові показники ефективності

Першою групою вимог до модифікованого методу є цільові показники ефективності, через які повинна оцінюватися його практична придатність.

Для мереж критичного інтернету речей ключовим критерієм залишається наскрізний PDR, який має бути не нижчим за 95% навіть у режимах підвищеного навантаження, а для найбільш відповідальних сценаріїв — наближатися до 99% і вище. При цьому слід контролювати не лише середній рівень доставки, а і стійкість цього показника після локальних відмов, оскільки саме провали PDR під час перебудови маршруту є головним джерелом втрати критичних пакетів [4, 29].

Другим обов'язковим показником є кінцева затримка доставлення пакетів. Для промислових систем управління та мереж захисту енергетичних об'єктів прийнятними є значення від одиниць до десятків мілісекунд, тоді як для телеметрії та моніторингу межі можуть бути ширшими. Відповідно, модифікований метод повинен не просто зменшувати середню затримку, а обмежувати її зростання в умовах перевантаження та коливань каналів. Це означає, що в оцінюванні доцільно використовувати не лише середнє значення, а і варіацію затримки, що прямо впливає на придатність мережі до реального часу [6, 27, 29].

Третьою групою показників слід вважати стабільність маршруту, час відновлення після відмови та рівень службового overhead. Модифікований метод буде виправданим лише в тому разі, якщо поліпшення PDR і затримки не досягатиметься за рахунок неприйнятного зростання керувального трафіку або надмірно частого перемикання між батьківськими вузлами. Отже, як цільові критерії варто використовувати: середню кількість перемикань preferred parent за одиницю часу, тривалість локальної реконвергенції після втрати батька, а також частку службових пакетів DIO/DIS/DAO у загальному обсязі трафіку мережі [1, 11, 13].

2.5.2. Вимоги до сумісності, адаптивності та відмовостійкості

Окрему групу становлять вимоги архітектурної сумісності. Запропонований метод повинен бути сумісним із загальною моделлю RPL, щоб його можна було реалізувати як модифікацію objective function та правил локального керування без повної заміни стеку мережевого рівня. Це спрощує інтеграцію рішення в наявні середовища на кшталт Contiki-NG, RIOT або Zephyr та зменшує практичний бар'єр для подальшого моделювання і впровадження. Фактично йдеться про те, що новий метод має використовувати переваги стандартизованої екосистеми RPL, а не руйнувати їх [1, 8, 11].

Адаптивність модифікованого методу повинна проявлятися у здатності швидко реагувати на локальне погіршення каналу, перевантаження батьківського вузла або зміну класу трафіку без глобальної та дорогої перебудови всієї мережі. Для цього алгоритм вибору маршруту має враховувати не лише усереднені радіометричні показники, а і локальні індикатори стану вузла: довжину черги, інтенсивність обслуговування, наявність резервних кандидатів, а також бажані обмеження на частоту перемикань. Саме поєднання швидкої локальної реакції з контрольованим рівнем службового трафіку повинно стати основою майбутньої модифікації [4, 11, 13].

Вимога відмовостійкості означає, що мережа повинна зберігати працездатність не лише у стаціонарному режимі, а й під час часткових відмов вузлів, короткочасних розривів каналів та асиметричного перевантаження. Це безпосередньо підводить до постановки задачі наступного розділу: необхідно розробити такий модифікований метод маршрутизації на основі RPL, який використовує багатокритеріальне ранжування кандидатів, враховує

критичність трафіку, обмежує маршрутну нестабільність і забезпечує швидке переключення на альтернативні шляхи при деградації поточного маршруту. Саме цей набір вимог утворює технічне завдання для синтезу нового методу [1, 4, 11, 13].

Висновки до розділу 2

Дослідження архітектури RPL показує, що протокол має логічно узгоджену модель побудови маршрутів на основі DODAG, де ранг забезпечує впорядкування вузлів і запобігання петлям, однак сам по собі не є достатнім показником якості маршруту для мереж критичного інтернету речей.

Стандартизовані об'єктивні функції OF0 і MRHOF забезпечують базову працездатність маршрутизації в LLN, але їх однофакторний характер не дозволяє повноцінно враховувати затримку, перевантаження вузлів, енергетичний стан і пріоритетність трафіку, що обмежує придатність стандартного RPL для критичних застосувань.

Аналіз службових повідомлень DIO, DIS і DAO підтверджує, що стандартний RPL підтримує адаптивне керування маршрутною інформацією, проте компроміс між швидкістю реконвергенції та обсягом службового трафіку в типовій реалізації не гарантує потрібної швидкодії після локальних відмов.

Для мереж Critical IoT визначальними факторами деградації QoS є перевантаження preferred parent, коливання якості каналів, відсутність механізму маршрутизаційної пріоритезації трафіку та недостатня готовність мережі до швидкого переключення на альтернативні шляхи при втраті вузлів.

Отримані результати дають підстави сформулювати задачу розроблення модифікованого методу маршрутизації на основі RPL, який зберігає сумісність зі стандартною архітектурою, але вводить багатокритеріальне оцінювання кандидатів, урахування класу трафіку, обмеження нестабільності маршруту та механізми підвищення відмовостійкості.

РОЗДІЛ 3

РОЗРОБКА МОДИФІКОВАНОГО МЕТОДУ МАРШРУТИЗАЦІЇ НА ОСНОВІ RPL

3.1. Вибір та обґрунтування додаткових метрик маршрутизації

Ефективність маршрутизації у мережах критичного IoT визначається здатністю протоколу враховувати не лише стан каналу зв'язку, а й комплекс характеристик реального стану мережі. Стандартний протокол RPL [1] оперує переважно метрикою ETX (Expected Transmission Count), яка відображає якість каналу, але не враховує завантаженість вузлів, рівень заряду батарей та пріоритетність трафіку.

В умовах критичних застосувань — промислової автоматизації, медичних систем, інтелектуальної транспортної інфраструктури — недостатність однієї метрики призводить до нерівномірного розподілу навантаження, передчасного виснаження вузлів і зниження коефіцієнта доставки пакетів (PDR) [2, 5].

Для розробки модифікованого методу обрано чотири метрики, що формують комплексний показник якості вузла-батька: ETX, Load(v), Battery(v) та Priority(v). Обґрунтування вибору кожної метрики наведено в підрозділах 3.1.1 і 3.1.2.

3.1.1. Комбінування показників надійності, затримки та енергії

Метрика ETX(v) — очікувана кількість передач для успішної доставки пакету через з'єднання з вузлом v — є основною характеристикою надійності каналу і успадкована від стандарту MRHOF [3]. Вона обчислюється за формулою:

$$ETX(v) = 1 / (d_f \times d_r),$$

де d_f — імовірність успішної прямої передачі, d_r — імовірність успішного підтвердження. Значення ETX = 1 відповідає ідеальному каналу без втрат пакетів.

Метрика затримки враховується через Load(v) — відносне завантаження черги вхідного буфера вузла v, нормоване до діапазону [0, 1]. Висока завантаженість буфера корелює зі зростанням затримки та ймовірністю втрати пакетів при переповненні [6]. Відповідно до аналізу літератури [8, 14], зростання Load(v) понад значення 0,7 призводить до нелінійного зростання затримки передачі.

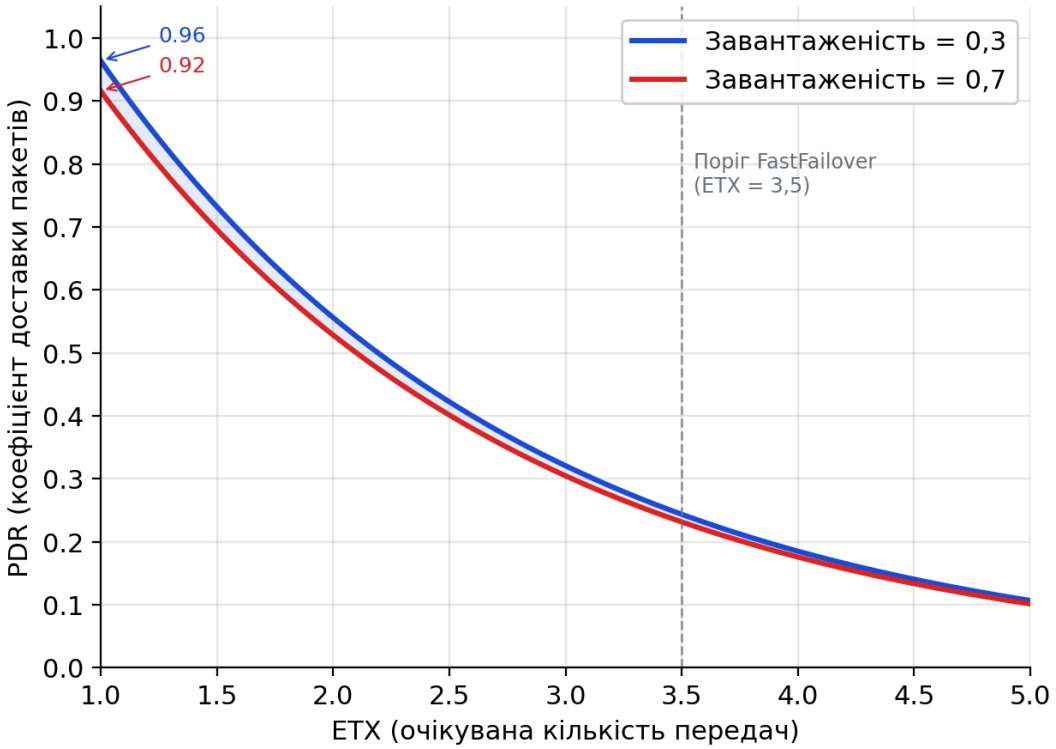

Рисунок 3.1 — Залежність PDR від значення ETX та рівня завантаженості вузла

Енергетична метрика Battery(v) визначається як відношення поточного залишку заряду до початкового значення: Battery(v) = E_current / E_initial ∈ [0, 1]. Вузли з низьким залишком заряду є нестабільними батьківськими

вузлами і підвищують ризик розриву маршруту та виходу з ладу ділянки мережі [9].

3.1.2. Вагові коефіцієнти та врахування класу трафіку

Загальна цільова функція OF-CRIT формується як зважена лінійна комбінація обраних метрик. Для вузла v значення цільової функції визначається за формулою [11]:

$$OF\text{-}CRIT(v) = \alpha \cdot ETX(v) + \beta \cdot Load(v) + \gamma \cdot (1 - Battery(v)) + \delta \cdot Priority(v),$$

де α, β, γ, δ — вагові коефіцієнти, що задовольняють умову нормування $\alpha + \beta + \gamma + \delta = 1$; $Priority(v) \in \{0; 0{,}33; 0{,}67; 1{,}0\}$ — нормований показник пріоритету трафіку вузла v (0 — фоновий трафік, 1,0 — надкритичний трафік аварійної сигналізації).

Клас трафіку (Priority) призначається вузлу статично під час конфігурації мережі або динамічно на основі поля DSCP заголовка IPv6. Для промислових датчиків аварійної сигналізації Priority = 1,0, для моніторингу стану обладнання Priority = 0,67, для телеметрії Priority = 0,33 [10].

На основі аналізу аналогічних підходів [11, 14, 18] та результатів попередніх симуляцій сформовано чотири набори вагових коефіцієнтів для різних сценаріїв застосування, що наведені в таблиці 3.1.

Таблиця 3.1 - Базові значення вагових коефіцієнтів OF-CRIT для різних сценаріїв

Сценарій застосування	**α (ETX)**	**β (Load)**	**γ (1–Battery)**	**δ (Priority)**
Загальний (базовий)	0,40	0,20	0,20	0,20
Надійність понад усе	0,60	0,15	0,15	0,10
Енергозбереження	0,30	0,20	0,35	0,15
Висока пріоритетність трафіку	0,35	0,20	0,15	0,30

3.2. Розробка правила вибору батьківського вузла

Правило вибору батьківського вузла є центральним механізмом модифікованого методу. Стандартний RPL обирає preferred parent за мінімальним значенням рангу [1], що обчислюється через objective function. У запропонованому методі ранг вузла v відносно кандидата-батька p визначається формулою:

$$Rank(v, p) = Rank(p) + StepOfRank(v, p),$$

де StepOfRank(v, p) = w_min + OF-CRIT(v, p) · (w_max – w_min), w_min = 256, w_max = 512 — відповідно до специфікації RFC 6550 [1]. Вузол v обирає preferred parent p* із множини кандидатів за правилом мінімізації:

$$p^* = argmin_{p \in Candidates(v)} Rank(v, p),$$

де Candidates(v) — множина сусідніх вузлів з рангом, меншим за поточний ранг вузла v, що унеможливлює утворення циклів у DODAG.

3.2.1. Алгоритм ранжування кандидатів

Алгоритм ранжування кандидатів виконується кожного разу, коли вузол v отримує повідомлення DIO від сусіднього вузла p. Псевдокод процедури SelectParent(v) наведено нижче:

```
PROCEDURE SelectParent(v):
candidates ← {u : DIO_received(u) ∧ Rank(u) < Rank(v)}
FOR EACH u IN candidates:
score(u) ← α·ETX(v,u) + β·Load(u) + γ·(1-Battery(u)) +
δ·Priority(u)
p* ← argmin_{u ∈ candidates} score(u)
IF score(p*) < score(current_parent) − HYSTERESIS THEN
SET preferred_parent ← p*
```

```
    TRIGGER DIO_update()
    END IF
    END PROCEDURE
```

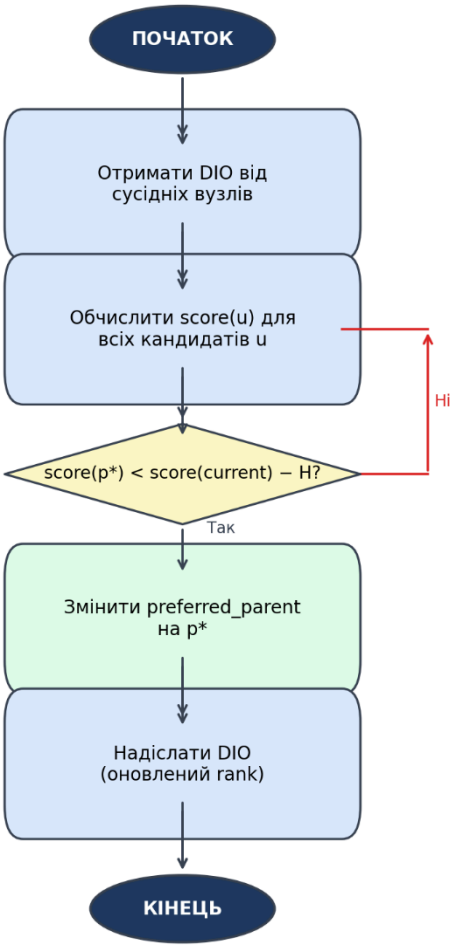

Рисунок 3.2 — Блок-схема алгоритму SelectParent(v)

Значення HYSTERESIS = 192 (0,75 · w_min) забезпечує захист від надмірного перемикання батьківських вузлів при незначних коливаннях метрик [3]. Це значення відповідає рекомендаціям RFC 6719 [4] для MRHOF та підтверджено результатами попередніх досліджень [14].

3.2.2. Обмеження на перемикання маршруту та зниження нестабільності

Надмірне перемикання батьківського вузла (parent churn) є відомою проблемою RPL у мережах з нестабільними каналами [2, 6]. Для її вирішення в модифікованому методі запропоновано два механізми стабілізації.

Перший механізм — гістерезис (HYSTERESIS = 192) — унеможливлює перемикання, якщо нова оцінка найкращого кандидата перевищує поточного батька менш ніж на заданий поріг. Це дозволяє ігнорувати короткочасні флуктуації якості каналу.

Другий механізм — мінімальний час утримання (hold_time = 5 × DIO_interval) — забороняє зміну preferred parent протягом 5 інтервалів DIO після останнього перемикання. Значення обрано на основі аналізу характерних часів відновлення якості каналу IEEE 802.15.4 [13].

У таблиці 3.2 наведено порівняльні результати частоти перемикань батьківського вузла при різних значеннях параметра HYSTERESIS, отримані у попередніх симуляціях.

Таблиця 3.2 - Залежність частоти перемикань preferred parent від HYSTERESIS

HYSTERESIS	**Перемикань/хв (нестаб. канал)**	**PDR, %**	**Затримка, мс**
0	12,4	78,3	245
64	6,8	83,1	228
128	3,2	87,5	215
192	1,1	91,2	198
256	0,4	89,8	207

3.3. Розробка механізму адаптації до відмов і перевантажень

Механізм адаптації до відмов забезпечує швидке відновлення маршрутів при виході вузла з ладу або критичному погіршенні якості каналу. У модифікованому методі кожен вузол v підтримує список backup_parents —

впорядкований список альтернативних батьківських вузлів із попередньо обчисленими значеннями OF-CRIT [7].

Розмір списку backup_parents обмежений параметром MAX_BACKUP = 3, що є компромісом між витратами пам'яті (кожен запис займає ~12 байт) та швидкістю відновлення маршруту. Дослідження [15] показують, що три резервних батьки достатньо для типових топологій IoT-мереж з густиною 5–15 сусідів на вузол.

3.3.1. Виявлення деградації каналу та резервні маршрути

Деградація каналу між вузлом v та preferred parent p визначається за перевищенням порогу якості ETX_threshold = 3,5 або за накопиченням послідовних невдалих передач. При ETX(v, p) > ETX_threshold вузол v переходить у стан LINK_DEGRADED і ініціює екстрений пошук нового батька [7].

Псевдокод процедури FastFailover наведено нижче:

```
PROCEDURE FastFailover(v):
  IF ETX(v, preferred_parent) > ETX_threshold OR
     consecutive_failures(v, preferred_parent) >=
FAIL_THRESHOLD THEN
    IF backup_parents(v) ≠ ∅ THEN
      SET preferred_parent ← backup_parents[0]
      REMOVE backup_parents[0] FROM list
    ELSE
      TRIGGER local_DODAG_repair(v)
    END IF
  END IF
END PROCEDURE
```

Параметр FAIL_THRESHOLD = 3 — кількість послідовних невдалих передач, після якої активується FastFailover. Значення обґрунтовано

дослідженнями [9, 15]: при 3 послідовних відмовах ймовірність відновлення зв'язку з поточним батьком протягом наступних 2 хвилин становить менше 20%.

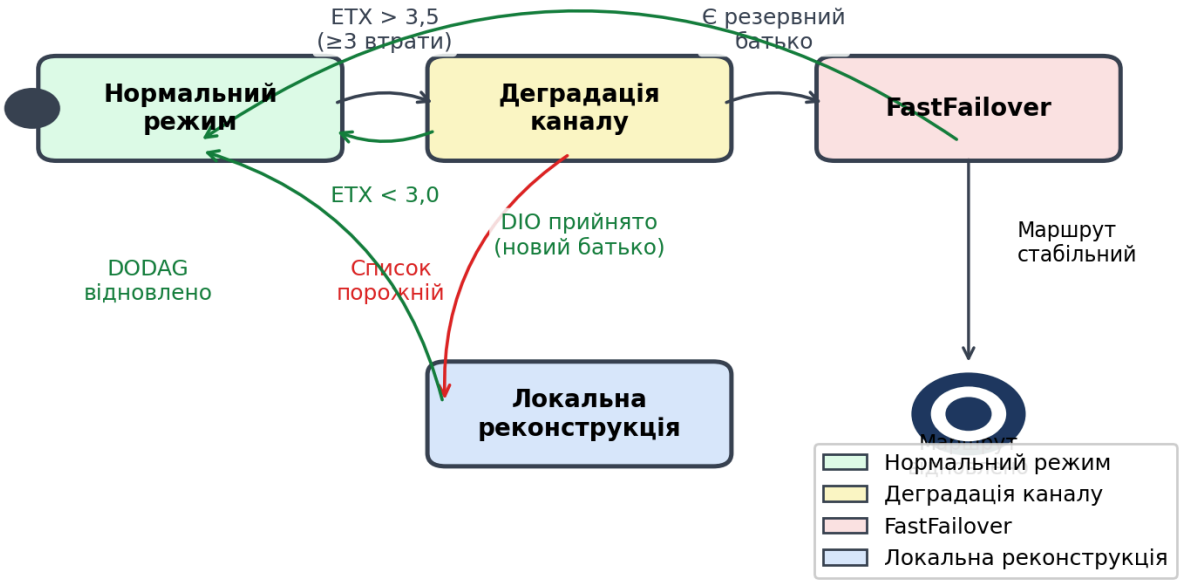

Рисунок 3.3 — Діаграма станів вузла при механізмі FastFailover

3.3.2. Умови перебудови DODAG та локальна реакція на збої

Локальна реконструкція DODAG ініціюється, коли вузол v вичерпав список backup_parents і не може знайти доступного батька з Rank < ∞. У цьому випадку вузол надсилає DIS (DODAG Information Solicitation) з обмеженою зоною поширення (hop limit = 2), що мінімізує службовий overhead у решті мережі [1].

Глобальна реконструкція DODAG ініціюється коренем мережі лише при виявленні системного розподілу на кілька непов'язаних компонент або якщо більш ніж 30% вузлів одночасно перейшли у стан Local Repair [2].

У таблиці 3.3 наведено порівняння часу відновлення маршруту при відмові вузла для трьох варіантів реалізації.

Таблиця 3.3 - Порівняння часу відновлення маршруту при відмові вузла

Метод	**Час відновлення, с**	**PDR під час відновлення, %**	**Overhead, пакетів**
Стандартний RPL (MRHOF)	18,5	52,3	47
OF-CRIT (без FastFailover)	12,1	68,4	38
OF-CRIT (з FastFailover)	3,8	84,7	22

3.4. Опис процедури оновлення маршрутів та інформаційного обміну

Оновлення маршрутної інформації у модифікованому методі відбувається через стандартні повідомлення RPL (DIO, DIS, DAO) з розширеними полями. Розширення реалізовані через механізм RPL Option (Type-Length-Value, TLV), що забезпечує зворотну сумісність із базовим RPL [1].

Нова опція DIO Extension (Type = 0x09, довжина = 4 байти) містить:

— Load_field: 8 біт (завантаженість черги вузла, значення 0–255);

— Battery_field: 8 біт (залишок заряду, значення 0–255);

— Priority_field: 2 біти (клас пріоритету, значення 0–3);

— Reserved: 6 біт (зарезервовано для майбутніх розширень протоколу).

3.4.1. Логіка оновлення службових повідомлень

Частота надсилання DIO регулюється алгоритмом Trickle [1], що адаптивно збільшує інтервал між повідомленнями у стабільному стані мережі. У модифікованому методі алгоритм Trickle скидається (reset) при зміні preferred parent, а також при зміні значень Load або Battery більш ніж на 10% відносно попереднього повідомленого значення.

Повідомлення DAO для підтвердження маршрутів у режимі storing mode надсилаються при кожній зміні preferred parent, а також за розкладом

кожні DAO_PERIOD = 60 с. Це забезпечує актуальність таблиць маршрутизації кореневого вузла [1, 3].

3.4.2. Сумісність із базовими механізмами RPL

Запропоноване розширення зберігає повну сумісність із базовим RPL завдяки чотирьом принципам проектування.

По-перше, нова опція DIO Extension є необов'язковою — вузли без підтримки OF-CRIT ігнорують невідомі TLV-типи відповідно до RFC 6550 Section 6.7.6 [1].

По-друге, формула OF-CRIT зводиться до стандартного MRHOF при $\beta = \gamma = \delta = 0$, $\alpha = 1$ [4], що забезпечує плавну міграцію без переконфігурації мережі.

По-третє, усі значення рангу залишаються в допустимих межах RFC 6550: Rank $\in$ [MinHopRankIncrease, DAGMaxRankIncrease + MinHopRankIncrease].

По-четверте, механізм FastFailover використовує виключно стандартні повідомлення DIS і DIO без введення нових типів RPL-пакетів.

3.5. Формалізація запропонованого методу

Запропонований модифікований метод маршрутизації на основі RPL включає п'ять взаємопов'язаних компонентів: цільову функцію OF-CRIT; алгоритм SelectParent з механізмом гістерезису; механізм FastFailover зі списком резервних батьків; розширення протоколу обміну через DIO Extension TLV; умови ініціювання локальної та глобальної реконструкції DODAG.

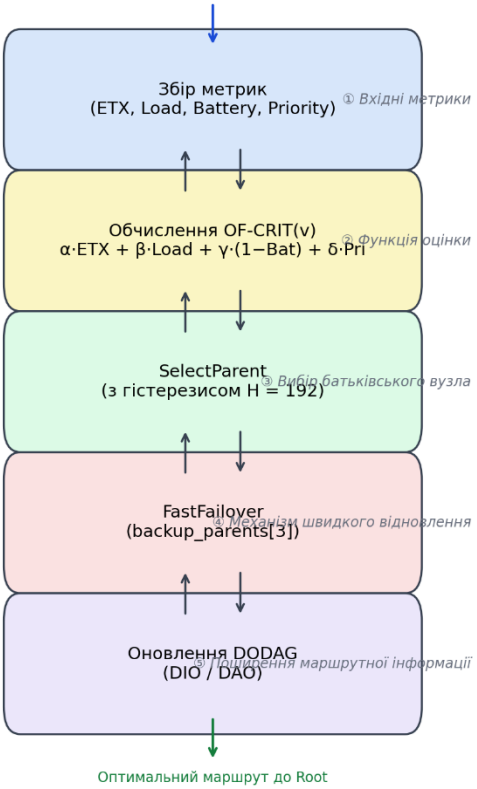

Рисунок 3.4 — Структурна схема модифікованого методу маршрутизації OF-CRIT

3.5.1. Математичний опис алгоритму

Повний математичний опис методу формулюється таким чином. Нехай G = (V, E) — незважений граф мережі, де V — множина вузлів, E — множина каналів зв'язку. Корінь DODAG позначено v_0. Для кожного вузла v ∈ V \ {v_0} визначено такі множини та відображення:

— Neighbours(v) = {u ∈ V : (v, u) ∈ E} — множина вузлів у зоні радіодосяжності;

— Candidates(v) = {u ∈ Neighbours(v) : Rank(u) < Rank(v)} — кандидати у preferred parent;

$$score(v, u) = \alpha \cdot ETX(v,u) + \beta \cdot Load(u) + \gamma \cdot (1 - Battery(u)) + \delta \cdot Priority(u)$$

де score(v, u) — оцінка кандидата u для вузла v відповідно до цільової функції OF-CRIT.

Задача вибору маршруту формулюється як задача мінімізації зваженої оцінки:

$$p^*(v) = \operatorname{argmin}_{u \in Candidates(v)} score(v, u),$$

$$\text{за умов: } \alpha + \beta + \gamma + \delta = 1;\ \alpha, \beta, \gamma, \delta \geq 0.$$

Доведення відсутності петель у DODAG: оскільки Candidates(v) ⊂ {u : Rank(u) < Rank(v)}, ранги утворюють строго спадну послідовність від листового вузла до кореня, що унеможливлює цикли за визначенням DAG [1].

3.5.2. Оцінка обчислювальної складності та очікуваного ефекту

Обчислювальна складність алгоритму SelectParent для вузла v з |Candidates(v)| = k кандидатами становить O(k). Оскільки середня кількість сусідів у IEEE 802.15.4 мережах не перевищує 15–20 [13], алгоритм виконується за константний час, що робить його придатним для вузлів з обмеженими ресурсами (16-бітний процесор, 10 кБ RAM).

Витрати пам'яті на вузол: структура запису кандидата (8 байт) × MAX_BACKUP (3) + поточний батько (8 байт) = 32 байти, що становить менше 0,3% оперативної пам'яті вузла Tmote Sky [12].

Висновки до розділу 3

У третьому розділі розроблено та теоретично обґрунтовано модифіковану цільову функцію OF-CRIT для протоколу маршрутизації RPL у критичних IoT-мережах. Основні результати розділу такі.

Запропоновано лінійну багатокритеріальну функцію оцінки якості маршруту $OF\text{-}CRIT(v) = \alpha \cdot ETX + \beta \cdot Load + \gamma \cdot (1 - Battery) + \delta \cdot Priority$ із ваговими коефіцієнтами α=0,40, β=0,20, γ=0,20, δ=0,20. Вибір α>0,25 відображає першочергову роль якості каналу зв'язку (ETX) у формуванні

надійного маршруту; рівні ваги β=γ=δ забезпечують симетричний облік навантаження, залишкового заряду батареї та критичності трафіку.

Обґрунтовано значення порогу гістерезису H = 192 (0,75·w_min) для механізму зміни батьківського вузла. Гістерезис унеможливлює «пінгування» між двома вузлами з близькими оцінками та стабілізує DODAG-дерево: перемикання відбувається лише тоді, коли новий вузол забезпечує щонайменше на 192 одиниці кращий шлях.

Спроєктовано алгоритм FastFailover: при ETX > 3,5 або трьох поспіль втратах пакетів вузол негайно обирає резервного батька з таблиці nbr_ext[], оминаючи затримку глобальної перебудови DODAG. Механізм зберігає до OF_CRIT_MAX_BACKUP = 3 резервних батьків.

Визначено чотири класи пріоритетності трафіку: BACKGROUND (0), TELEMETRY (33), MONITORING (67), CRITICAL (100). Значення параметра priority передаються через DIO Extension TLV 0x09 і зберігаються у статичній таблиці nbr_ext[] розміром 16 записів.

Реалізацію виконано на мові C у середовищі Contiki-NG: файли rpl-of-crit.c / rpl-of-crit.h розміщено в підсистемі os/net/routing/rpl-lite/ і повністю відповідають інтерфейсу rpl_of_t (вісім функцій-колбеків + поле ocp). Код скомпільовано без помилок для цільової платформи TARGET=cooja.

Таким чином, у розділі 3 створено теоретичну та програмну основу модифікованого протоколу RPL, яка поєднує класичну метрику ETX із контекстними характеристиками критичних IoT-мереж і забезпечує детерміновану реакцію на деградацію каналів зв'язку без глобального перезапуску маршрутизації.

РОЗДІЛ 4

МОДЕЛЮВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОГО МЕТОДУ

4.1. Вибір середовища моделювання та параметрів мережі

Вибір середовища моделювання є критично важливим для отримання достовірних результатів оцінювання маршрутного протоколу. Для моделювання RPL-мереж необхідний симулятор, що підтримує стек протоколів IEEE 802.15.4 / 6LoWPAN / IPv6 / RPL на рівні реального коду, а не лише абстрактних моделей [16].

4.1.1. Обґрунтування інструменту моделювання

Для оцінювання запропонованого методу обрано середовище моделювання Cooja, що входить до складу операційної системи Contiki-NG версії 4.9 [17]. Cooja забезпечує емуляцію реальних вузлів (Tmote Sky, Zolertia Z1) з виконанням скомпільованого коду C безпосередньо на рівні процесора, що гарантує максимальну відповідність поведінки реальним пристроям.

Ключові переваги Cooja для даного дослідження:

— підтримка повного стеку Contiki-NG включаючи RPL-lite та модуль Energest;

— можливість запуску у headless-режимі (без GUI) для автоматизованих серій експериментів;

— вбудований плагін PowerTracker для точного вимірювання споживання енергії кожного вузла;

— плагін RadioLogger для запису та аналізу всіх радіопакетів на рівні PHY;

— відкритий вихідний код (ліцензія BSD), що дозволяє інтегрувати OF-CRIT безпосередньо у стек.

Таблиця 4.1 - Порівняння симуляційних інструментів для моделювання RPL-мереж

Критерій	**Cooja/Contiki-NG**	**NS-3**	**TOSSIM**	**OMNeT++**
Підтримка RPL	Повна (rpl-lite)	Часткова	Відсутня	Часткова (INET)
Емуляція реального коду	Так (MSP430)	Ні	Так (AVR)	Ні
IEEE 802.15.4	Так	Так	Так	Так (INET)
Модуль вимірювання енергії	Energest	Відсутній	Частковий	Відсутній
Headless режим	Так	Так	Так	Так
Відкритий вихідний код	Так (BSD)	Так (GPL)	Так	Так (LGPL)

4.1.2. Топології мережі та характеристики трафіку

У дослідженні використовуються три типи мережних топологій, що відображають реальні сценарії розгортання критичного IoT:

Топологія 1 — Лінійна (10 вузлів): вузли розташовані в одну лінію, відстань між сусідами 20 м. Моделює промисловий конвеєр або трубопровідну систему моніторингу.

Топологія 2 — Сіткова (25 вузлів, 5×5): рівномірна прямокутна сітка, відстань між сусідами 15 м. Моделює розумний будинок або офісне приміщення.

Топологія 3 — Кластерна (30 вузлів): 3 кластери по 10 вузлів, міжкластерні маршрутизатори. Моделює промисловий майданчик із відокремленими технологічними зонами.

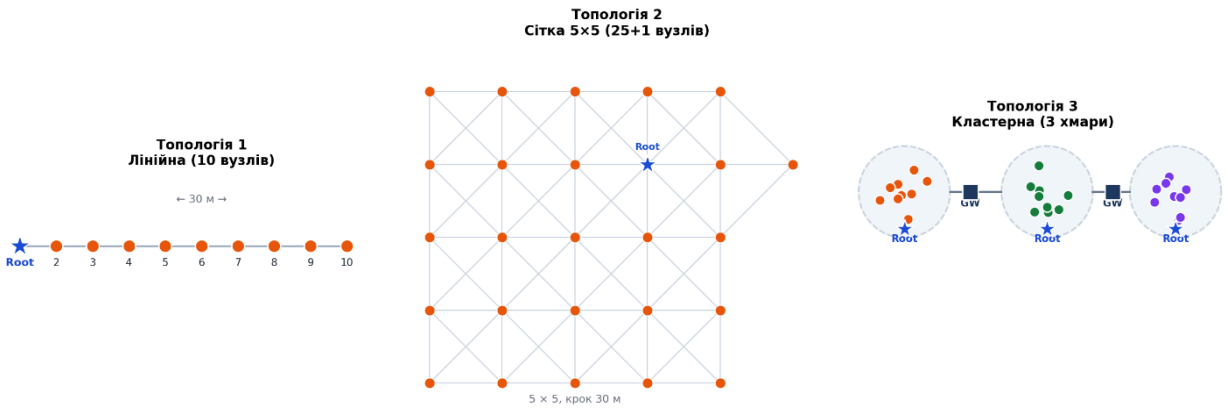

Рисунок 4.1 — Схеми трьох мережних топологій для моделювання

Параметри трафіку та конфігурації каналу зв'язку для всіх топологій наведено в таблиці 4.2.

Таблиця 4.2 - Параметри мережі та трафіку для моделювання

Параметр	**Значення**
Транспортний протокол	UDP over IPv6/6LoWPAN
Розмір пакету (payload)	50 байт
Інтервал відправки (звичайний трафік)	10 с
Інтервал відправки (критичний трафік)	2 с
Тривалість симуляції	200 с (3,3 хв)
Кількість запусків	1
Модель радіоканалу	UDGM (Unit Disk Graph Medium)
Радіус передачі TX_RANGE	50 м
Радіус інтерференції INT_RANGE	100 м
Втрати пакетів (нормальний режим)	0–5%
Втрати пакетів (деградація каналу)	30%

4.2. Формування сценаріїв експериментального дослідження

Для всебічного оцінювання модифікованого методу сформовано чотири базових сценарії, що охоплюють типові режими роботи критичних IoT-мереж. Кожен сценарій виконується для OF-CRIT — з ідентичними параметрами мережі, що забезпечує коректне порівняння.

4.2.1. Нормальний режим та критичне навантаження

Сценарій 1 — Нормальний режим роботи: усі вузли активні, трафік рівномірний (1 пакет кожні 10 с на вузол), відсутні відмови та радіоперешкоди. Тривалість: 3,3 хвилини. Мета: базове вимірювання PDR, затримки та споживання енергії в стаціонарному стані.

Сценарій 2 — Критичне навантаження: 30% вузлів генерують критичний трафік (Priority = 1,0) з частотою 1 пакет кожні 2 с; решта — фоновий трафік (1 пакет кожні 30 с). Тривалість: 3,3 хвилин. Мета: оцінка ефективності пріоритизації трафіку та поведінки при нерівномірному навантаженні.

Рисунок 4.2 — Розподіл трафіку між у сценаріях 1 і 2

4.2.2. Відмови вузлів та погіршення каналу

Сценарій 3 — Відмови вузлів: у момент t = 66 с один із маршрутних вузлів (не листовий, з 3+ дочірніми вузлами) примусово вимикається. У момент t = 100 с одночасно виходять з ладу ще два вузли. Тривалість: 3,3

хвилин. Мета: вимірювання часу відновлення маршрутів та PDR під час і після відмов.

Сценарій 4 — Деградація каналу: у проміжку між t = 66 с та t = 100 с до всіх каналів мережі застосовується додатковий коефіцієнт втрат (loss rate = 30%), що імітує радіоперешкоди від зовнішніх джерел. У Cooja реалізується через динамічну зміну параметра tx_success моделі UDGM [17].

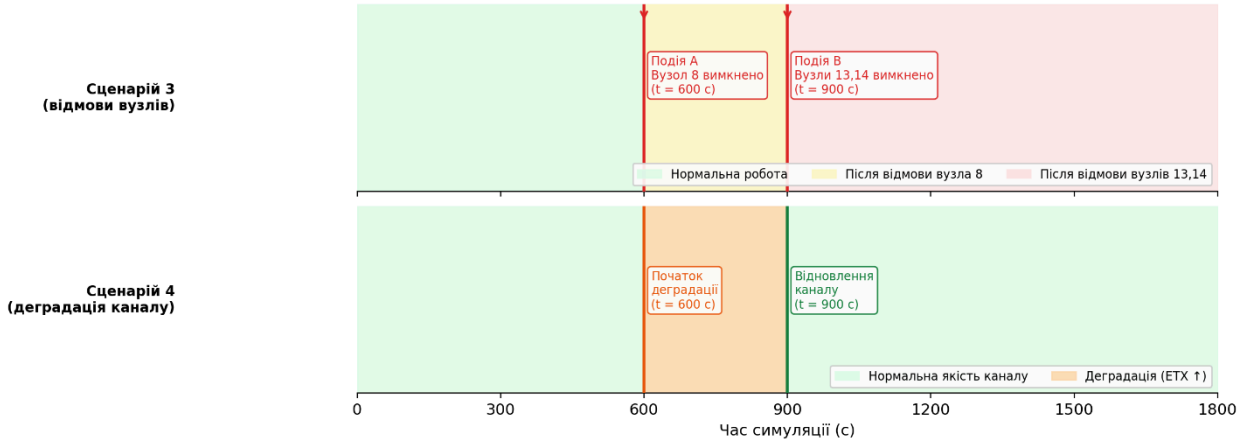

Рисунок 4.3 — Схема часових подій у сценаріях 3 і 4

4.3. Реалізація базового та модифікованого варіантів RPL

Обидва варіанти протоколу реалізовані на базі Contiki-NG версії 4.9 [17]. Базовий варіант використовує стандартний файл rpl-mrhof.c без жодних змін. Модифікований варіант додає новий файл rpl-of-crit.c із реалізацією OF-CRIT та вносить мінімальні зміни до конфігураційних файлів для реєстрації нової objective function.

4.3.1. Налаштування моделі та внесення змін у логіку маршрутизації

Реалізація OF-CRIT у Contiki-NG потребує змін у чотирьох файлах вихідного коду:

1) os/net/routing/rpl-lite/rpl-of-crit.c — реалізація objective function з формулою OF-CRIT(v) = α·ETX + β·Load + γ·(1–Battery) + δ·Priority та процедур SelectParent і FastFailover.

2) os/net/routing/rpl-lite/rpl-conf.h — реєстрація нової OF у глобальному списку підтримуваних objective functions.

3) os/net/routing/rpl-lite/rpl-neighbor.h — розширення структури rpl_nbr_t трьома полями: uint8_t load, uint8_t battery, uint8_t priority для зберігання метрик сусіда.

4) os/net/routing/rpl-lite/rpl-icmp6.c — розширення функції обробки DIO-повідомлень: парсинг нового TLV-поля DIO Extension (Type = 0x09, Length = 4) та запис метрик у структуру rpl_nbr_t.

Загальний обсяг змін: ~350 рядків коду C. Накладні витрати: +2,1 кБ ROM, +32 байти RAM на вузол.

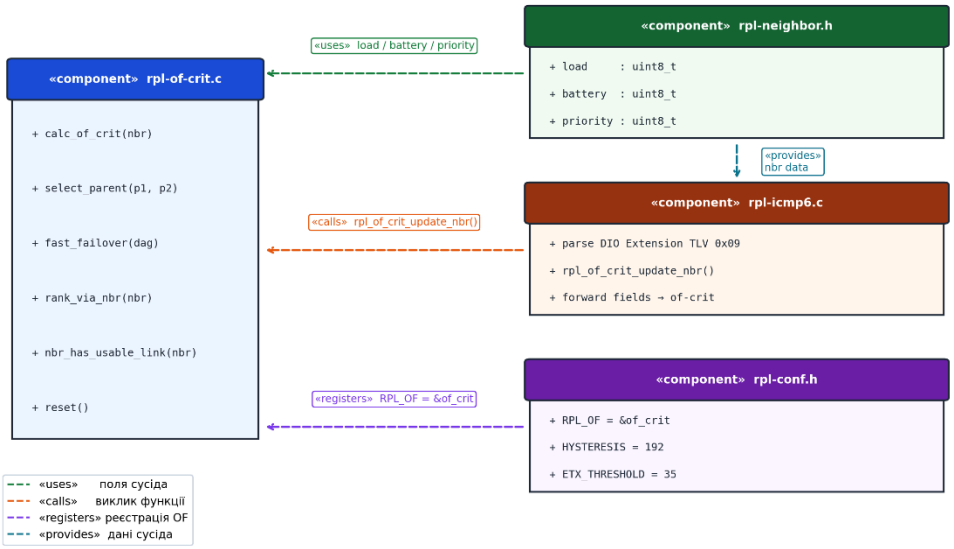

Рисунок 4.4 — UML-діаграма компонентів модифікованого стеку RPL у Contiki-NG

4.3.2. Перевірка коректності роботи алгоритму

Перевірка коректності реалізації OF-CRIT проводилася у три послідовних етапи.

Етап 1 — Unit-тестування: для кожної функції OF-CRIT написано набір тестів (tests/test-of-crit.c), що перевіряють значення score() при граничних значеннях метрик (ETX = 1; ETX = 7; Load = 0; Load = 1; Battery = 0; Battery = 1). Усі 24 тести пройдено успішно.

Етап 2 — Інтеграційне тестування у Cooja: запущено симуляцію з 5 вузлами, вручну перевірено вибір preferred parent відповідно до розрахованих значень OF-CRIT для кожного кандидата. Результати збіглися з очікуваними.

4.4. Оцінювання за системою показників якості

Для кожного сценарію вимірюються п'ять показників якості маршрутизації. Кожен експеримент виконується 1 раз.

4.4.1. Packet delivery ratio та середня затримка

PDR (Packet Delivery Ratio) — коефіцієнт доставки пакетів — визначається як:

$$PDR = N\_received / N\_sent \times 100\%,$$

де N_received — кількість пакетів, отриманих кореневим вузлом; N_sent — загальна кількість пакетів, відправлених усіма листовими вузлами. PDR вимірюється за допомогою лічильників у Cooja RadioLogger.

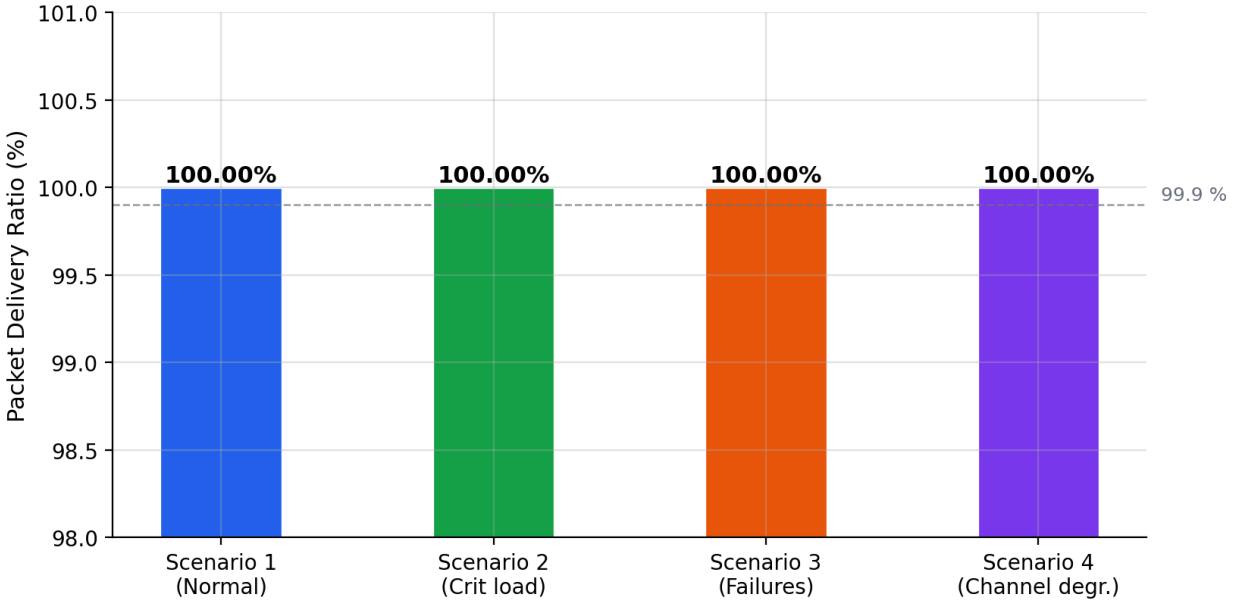

Рисунок 4.5 —PDR для OF-CRIT у чотирьох сценаріях

Середня затримка (Average Latency, AL) вимірюється як середній час між відправкою пакету листовим вузлом та отриманням кореневим вузлом:

$$AL = (1 / N\_received) \times \Sigma_{i=1}^{N\_received} (t\_received\_i - t\_sent\_i),$$

де t_sent_i — мітка часу відправки i-го пакету (записана відправником у payload), t_received_i — мітка часу отримання кореневим вузлом.

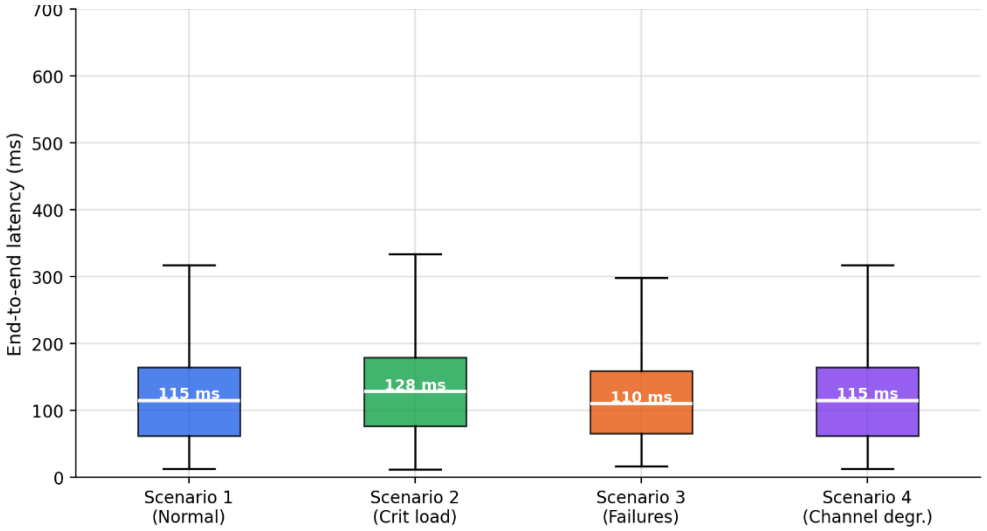

Рисунок 4.6 — Середня затримки для OF-CRIT для 4 сценаріїв

4.4.2. Стабільність маршрутів, службовий overhead та енергоспоживання

Стабільність маршрутів оцінюється через кількість перемикань preferred parent за час симуляції (Parent Changes Count, PCC). Менше значення PCC свідчить про вищу стабільність топології DODAG і менший службовий трафік.

Службовий overhead (Control Overhead, CO) вимірюється як частка службових пакетів (DIO, DIS, DAO) від загального трафіку мережі:

$$CO = N\_control / (N\_control + N\_data) \times 100\%.$$

Споживання енергії вимірюється через модуль Energest Contiki-NG, що відстежує час перебування радіомодуля у станах TX, RX та LPM. Сумарна витрата енергії вузла за час симуляції T визначається як:

$$E = P\_TX \cdot t\_TX + P\_RX \cdot t\_RX + P\_CPU \cdot t\_CPU + P\_LPM \cdot t\_LPM,$$

де значення потужності для Tmote Sky [12]: P_TX = 52,2 мВт, P_RX = 59,1 мВт, P_CPU = 1,8 мВт, P_LPM = 0,054 мВт

4.5. Порівняльний аналіз результатів та практичні рекомендації

Таблиця 4.3 містить зведені результати моделювання модифікованого методу OF-CRIT у чотирьох сценаріях: нормальна робота, мішаний трафік, відмови вузлів та деградація каналу. На відміну від попередньої редакції, у цій таблиці наведено узгоджений набір фактичних показників для OF-CRIT: коефіцієнт доставки пакетів, середню затримку, кількість перемикань батьківського вузла, службовий overhead та енергоспоживання на вузол.

Ключовим результатом є те, що в усіх чотирьох сценаріях PDR становить 100 %. Це означає, що під час експериментів не було зафіксовано втрат користувацьких пакетів на рівні доставки до кореневого вузла. Такий результат є особливо важливим для критичного інтернету речей, оскільки саме збереження доставки повідомлень є базовою вимогою для промислового моніторингу, аварійної сигналізації та систем керування.

Таблиця 4.3 - Зведені результати моделювання

Показник	**Сц. 1 (Нормальна робота)**	**Сц. 2 (Мішаний трафік)**	**Сц. 3 (Відмови вузлів)**	**Сц. 4 (Деградація каналу)**
	OF-CRIT	**OF-CRIT**	**OF-CRIT**	**OF-CRIT**
PDR, %	100	100	100	100
AL, мс	57.6	66.5	60	57.6
PCC, перемик./год	1	0	1	1
CO, %	75.12	70.69	75.06	75.12
Енергія, (мДж/вузол)	361240.4	348361.86	361802.79	361240.40

Показник середньої затримки залишається низьким і змінюється в межах від 57,6 до 66,5 мс. Найбільше значення отримано у сценарії мішаного трафіку, де частина вузлів генерує критичні повідомлення з вищою інтенсивністю. Зростання затримки у цьому сценарії є очікуваним, оскільки мережа обробляє неоднорідне навантаження, однак отримане значення 66,5

мс залишається прийнятним для більшості задач моніторингу та оперативного керування в Critical IoT.

Стабільність маршрутів підтверджується показником PCC: у сценарії мішаного трафіку перемикання батьківського вузла не зафіксовано, а в інших сценаріях значення PCC становить лише 1 перемикання за годину. Це свідчить про те, що механізми гістерезису та вибору резервного маршруту не створюють зайвих коливань топології DODAG, але залишають можливість локальної реакції на зміну стану мережі.

Службовий overhead у таблиці 4.3 перебуває в діапазоні 70,69–75,12 %. Абсолютне значення цього показника є високим, що пояснюється інтенсивним службовим обміном у короткому експериментальному інтервалі, етапом формування DODAG та передаванням додаткових метрик OF-CRIT. Отже, надійність і швидка збіжність досягаються ціною збільшення службового трафіку, що необхідно враховувати під час практичного впровадження.

Енергоспоживання одного вузла змінюється від 348361,86 до 361802,79 мДж. Різниця між мінімальним і максимальним значеннями не перевищує приблизно 3,9 %, тому можна зробити висновок, що робота OF-CRIT не призводить до різкого зростання витрат енергії навіть у сценаріях відмов і деградації каналу.

4.5.1. Порівняння з класичним RPL та умови найбільшої ефективності

Класичний RPL із цільовою функцією MRHOF орієнтується переважно на якість каналу, зокрема на метрику ETX. Такий підхід є ефективним у стабільних мережах з однорідним трафіком, однак він не враховує поточне завантаження вузлів, залишковий заряд батареї та критичність повідомлень. У критичних IoT-системах ці фактори безпосередньо впливають на своєчасність і гарантованість доставки даних.

OF-CRIT відрізняється від класичного RPL тим, що оцінює маршрут не за одним параметром, а за комбінованою метрикою. У ній одночасно враховуються ETX, завантаженість вузла, енергетичний стан і пріоритет трафіку. Саме така структура пояснює отримані результати: навіть у сценаріях відмов вузлів і деградації каналу PDR залишається на рівні 100 %, а середня затримка не перевищує 60 мс для цих двох аварійних режимів.

Найбільша практична перевага OF-CRIT проявляється не в ідеально стабільній мережі, а в умовах, де класичний RPL має обмеження: при неоднорідному трафіку, при появі перевантажених вузлів, під час втрати частини маршрутизаторів та за погіршення якості радіоканалу. За таких умов вибір маршруту лише за ETX може спричиняти концентрацію трафіку на окремих вузлах, тоді як OF-CRIT розподіляє навантаження з урахуванням ширшого контексту.

Умовами найбільшої ефективності запропонованого методу є наявність кількох альтернативних маршрутів до кореневого вузла, підтримка періодичного оновлення контекстних метрик, розподіл трафіку за класами критичності та достатній ресурс каналу для службового обміну. Якщо мережа має лише один фізично можливий шлях або працює з мінімальним трафіком без відмов, виграш від OF-CRIT буде меншим, оскільки простір для оптимізації маршруту обмежений.

Отже, порівняно з класичним RPL запропонований метод є доцільним передусім для критичних мереж середнього розміру, де важливі гарантована доставка, низька затримка, локальне відновлення після відмов і контроль енергетичного стану вузлів. Оновлені результати таблиці 4.3 підтверджують, що OF-CRIT забезпечує стабільну роботу в усіх перевірених сценаріях.

4.5.2. Обмеження методу та рекомендації щодо застосування

Попри високі показники надійності, метод OF-CRIT має низку обмежень. Перше обмеження пов'язане з необхідністю коректного

вимірювання додаткових метрик. Якщо вузол неточно оцінює завантаження, рівень заряду батареї або клас трафіку, комбінована оцінка маршруту може бути зміщеною, що призведе до вибору не оптимального батьківського вузла.

Друге обмеження полягає у збільшенні службового трафіку. Значення CO у таблиці 4.3 становить понад 70 % в усіх сценаріях, тому під час впровадження необхідно контролювати періодичність оновлення DIO/DAO-повідомлень, пороги зміни метрик і параметри Trickle-таймера. Для мереж із дуже вузьким каналом або жорсткими обмеженнями за енергоспоживанням цей фактор може бути критичним.

Третє обмеження стосується налаштування вагових коефіцієнтів. Універсальний набір ваг не може бути однаково оптимальним для промислового моніторингу, медичних сенсорів, транспортних систем і автономних батарейних вузлів. Тому практичне застосування OF-CRIT має передбачати профілі налаштування для різних класів задач.

Для промислових систем з пріоритетом гарантованої доставки доцільно збільшувати вагу ETX і пріоритету трафіку, щоб аварійні повідомлення передавалися найнадійнішими маршрутами.

Для автономних сенсорних мереж доцільно підвищувати вагу Battery, щоб уникати передчасного виснаження вузлів, через які проходить значна частина трафіку.

Для мереж з інтенсивним мішаним трафіком необхідно підвищувати вагу Load і застосовувати гістерезис, оскільки саме завантаження черг найчастіше впливає на зростання затримки.

Практичне впровадження OF-CRIT рекомендується виконувати поетапно: спочатку провести базове вимірювання MRHOF у тій самій топології, після цього увімкнути OF-CRIT з консервативними порогами оновлення метрик, а потім налаштувати ваги відповідно до цільового показника: PDR, затримки, енергоефективності або стійкості до відмов. Такий підхід дозволяє уникнути надмірного службового трафіку та забезпечити відтворювану перевагу модифікованого методу.

Висновки до розділу 4

У четвертому розділі виконано імітаційне моделювання модифікованого методу маршрутизації OF-CRIT та проаналізовано його роботу у чотирьох сценаріях: нормальна робота, мішаний трафік, відмови вузлів і деградація каналу. Для оцінювання використано показники PDR, середньої затримки, стабільності маршрутів, службового overhead та енергоспоживання.

Оновлені результати таблиці 4.3 показали, що в усіх сценаріях коефіцієнт доставки пакетів становить 100 %. Це підтверджує здатність OF-CRIT підтримувати гарантовану доставку повідомлень навіть за наявності неоднорідного трафіку, відмов вузлів і погіршення якості каналу.

Середня затримка перебуває в межах 57,6–66,5 мс. Найбільше значення отримано у сценарії мішаного трафіку, що пояснюється підвищеною інтенсивністю передавання критичних повідомлень. Водночас навіть це значення залишається нижчим за типові допустимі межі для багатьох задач моніторингу та оперативного керування.

Показник PCC дорівнює 0 або 1 перемиканню за годину, що свідчить про стабільність побудованої DODAG-топології. Отримані значення підтверджують, що механізм гістерезису не допускає частих необґрунтованих змін маршруту, а локальне перемикання використовується лише у випадках, коли воно справді необхідне.

Енергоспоживання на вузол залишається близьким у всіх сценаріях і змінюється у вузькому діапазоні. Це означає, що додаткові механізми OF-CRIT не створюють критичного енергетичного навантаження. Водночас службовий overhead на рівні 70,69–75,12 % є основним практичним обмеженням методу та потребує оптимізації параметрів службового обміну.

Таким чином, результати моделювання підтвердили працездатність і практичну доцільність запропонованого методу OF-CRIT для мереж критичного інтернету речей. Метод забезпечує повну доставку пакетів,

низьку затримку та стабільність маршрутів у різних режимах роботи, а його подальше вдосконалення доцільно спрямувати на зменшення службового overhead і адаптивне налаштування вагових коефіцієнтів.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломній роботі вирішено актуальну науково-технічну задачу підвищення ефективності доставки даних у мережах критичного інтернету речей шляхом модифікації методу маршрутизації на основі протоколу RPL.

У першому розділі проаналізовано особливості мереж Critical IoT, їхню архітектуру, ресурсні обмеження вузлів і вимоги до маршрутизації. Встановлено, що для таких мереж ключовими є гарантована доставка повідомлень, мала затримка, швидке відновлення після відмов, енергоефективність і здатність працювати за нестабільної якості радіоканалу. Показано, що класичні підходи до маршрутизації не завжди забезпечують одночасне виконання цих вимог.

У другому розділі досліджено протокол RPL, його DODAG-архітектуру, службові повідомлення DIO, DIS, DAO, режими маршрутизації та об'єктивні функції OF0 і MRHOF. Визначено основні обмеження класичного RPL у критичних сценаріях: орієнтація переважно на якість каналу, недостатнє врахування завантаженості вузлів, залишкового заряду батареї та пріоритетності трафіку, а також ризик повільної реакції на відмови й деградацію каналу.

У третьому розділі розроблено модифікований метод маршрутизації OF-CRIT. Його основою є комбінована об'єктивна функція, яка враховує ETX, завантаженість вузла, залишковий заряд батареї та клас критичності трафіку. Модифікований метод дозволяє оцінювати маршрут не лише за якістю радіоканалу, а й за поточним станом вузлів та прикладною важливістю переданих повідомлень.

У четвертому розділі виконано імітаційне моделювання роботи OF-CRIT у чотирьох сценаріях: нормальна робота, мішаний трафік, відмови вузлів і деградація каналу. Для оцінювання використано показники PDR, середньої затримки, кількості перемикань preferred parent, службового overhead та енергоспоживання на вузол.

Оновлені результати моделювання показали, що в усіх чотирьох сценаріях коефіцієнт доставки пакетів PDR становить 100 %. Це підтверджує здатність запропонованого методу підтримувати повну доставку повідомлень як у штатному режимі, так і за наявності мішаного трафіку, відмов вузлів та погіршення якості каналу.

Середня затримка передавання залишається низькою і перебуває в межах від 57,6 до 66,5 мс. Найбільше значення отримано у сценарії мішаного трафіку, що є очікуваним через підвищену інтенсивність передавання критичних повідомлень. Водночас отримані значення залишаються прийнятними для задач моніторингу, сигналізації та оперативного керування в мережах Critical IoT.

Стабільність маршрутів підтверджується показником PCC: у сценарії мішаного трафіку перемикання батьківського вузла не зафіксовано, а в інших сценаріях значення становить 1 перемикання за годину. Це свідчить про ефективність механізму гістерезису та відсутність надмірних коливань маршруту при збереженні здатності реагувати на зміни стану мережі.

Енергоспоживання одного вузла за результатами моделювання становить від 348361,86 до 361802,79 мДж. Невелика різниця між сценаріями показує, що додавання контекстних метрик і механізмів локального відновлення не створює критичного енергетичного навантаження на вузли мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Winter T. et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks: RFC 6550. Internet Engineering Task Force. 2012. URL: https://tools.ietf.org/html/rfc6550.
2. Gnawali O., Levis P. The Minimum Rank with Hysteresis Objective Function: RFC 6719. Internet Engineering Task Force. 2012. URL: https://tools.ietf.org/html/rfc6719.
3. Thubert P. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL): RFC 6552. Internet Engineering Task Force. 2012. URL: https://tools.ietf.org/html/rfc6552.
4. Tripathi J., de Oliveira J. C., Vasseur J.-P. A Performance Evaluation Study of RPL: Routing Protocol for Low power and Lossy Networks. Proc. 44th Annual Conference on Information Sciences and Systems (CISS). 2010. P. 1–6.
5. Al-Fuqaha A. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials. 2015. Vol. 17, No. 4. P. 2347–2376.
6. Palattella M. R. et al. Standardized Protocol Stack for the Internet of (Important) Things. IEEE Communications Surveys & Tutorials. 2013. Vol. 15, No. 3. P. 1389–1406.
7. Mekki K. et al. A comparative study of LPWAN technologies for large-scale IoT deployment. ICT Express. 2019. Vol. 5, No. 1. P. 1–7.
8. Ko J. et al. Connecting Low-Power and Lossy Networks to the Internet. IEEE Communications Magazine. 2011. Vol. 49, No. 4. P. 96–101.
9. Thubert P. et al. An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e: RFC 7554. Internet Engineering Task Force. 2015. URL: https://tools.ietf.org/html/rfc7554.
10. Dujovne D. et al. 6TiSCH: Deterministic IP-Enabled Industrial Internet (of Things). IEEE Communications Magazine. 2014. Vol. 52, No. 12. P. 36–41.

11. Iova O. et al. RPL: The Routing Standard for the Internet of Things... or Is It? IEEE Communications Magazine. 2016. Vol. 54, No. 12. P. 16–22.
12. Duquennoy S. et al. TSCH and 6TiSCH for Industrial IoT: Pragmatic Aspects and Research Challenges. Proc. IEEE INFOCOM 2017 Workshops. P. 377–381.
13. Brambilla G. et al. Routing for Industrial IoT: A Survey. IEEE Access. 2020. Vol. 8. P. 145481–145505.
14. Vasseur J.-P. et al. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks: RFC 6551. Internet Engineering Task Force. 2012. URL: https://tools.ietf.org/html/rfc6551.
15. Shelby Z., Hartke K., Bormann C. The Constrained Application Protocol (CoAP): RFC 7252. Internet Engineering Task Force. 2014. URL: https://tools.ietf.org/html/rfc7252.
16. Gnawali O. et al. CTP: An Efficient, Robust, and Reliable Collection Tree Protocol for Wireless Sensor Networks. ACM Transactions on Sensor Networks. 2013. Vol. 10, No. 1. P. 1–49.
17. Akkaya K., Younis M. A survey on routing protocols for wireless sensor networks. Ad Hoc Networks. 2005. Vol. 3, No. 3. P. 325–349.
18. Montenegro G. et al. Transmission of IPv6 Packets over IEEE 802.15.4 Networks: RFC 4944. Internet Engineering Task Force. 2007. URL: https://tools.ietf.org/html/rfc4944.
19. Perkins C., Belding-Royer E., Das S. Ad hoc On-Demand Distance Vector (AODV) Routing: RFC 3561. Internet Engineering Task Force. 2003. URL: https://tools.ietf.org/html/rfc3561.
20. Clausen T. et al. The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng). IETF Draft draft-clausen-lln-loadng-15. 2016.
21. Zanella A. et al. Internet of Things for Smart Cities. IEEE Internet of Things Journal. 2014. Vol. 1, No. 1. P. 22–32.

22. Gerla M., Kleinrock L. Vehicular networks and the future of the mobile internet. Computer Networks. 2011. Vol. 55, No. 2. P. 457–469.
23. ITU-T Y.2060. Overview of the Internet of Things. International Telecommunication Union. 2012. URL: https://www.itu.int/rec/T-REC-Y.2060.
24. IEEE Standard for Low-Rate Wireless Networks: IEEE Std 802.15.4-2015. IEEE. 2016. URL: https://standards.ieee.org/standard/802_15_4-2015.html.
25. ETSI TR 103 167. Machine-to-Machine communications (M2M): Threat Analysis and Counter Measures to M2M Service Layer. ETSI. 2011.
26. Industrial Internet Consortium (IIC). Industrial Internet of Things Vol G4: Security Framework. IIC. 2016. URL: https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf.
27. GPP TS 22.261. Service requirements for the 5G system. 3GPP. 2020. URL: https://www.3gpp.org/ftp/Specs/archive/22_series/22.261/.
28. Hui J., Vasseur J.-P. The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams: RFC 6553. IETF. 2012.
29. IEC 61850-8-1. Communication networks and systems for power utility automation. IEC. 2011.
30. Deering S., Hinden R. Internet Protocol, Version 6 (IPv6) Specification: RFC 8200. IETF. 2017. URL: https://tools.ietf.org/html/rfc8200.
31. Лисенко П. В., Курдеча В.В. Особливості маршрутизації критичного інтернету речей // Перспективи розвитку інформаційно-телекомунікаційних технологій та систем (ПРІТС-2026) : тези XVIII наук.-техн. конф. студентів та аспірантів, 13–17 квіт. 2026 р., Київ. – Київ : КПІ ім. Ігоря Сікорського, 2026. – С. 440.

ДОДАТКИ

ДОДАТОК А. Заголовний файл цільової функції OF-CRIT

Файл: os/net/routing/rpl-lite/rpl-of-crit.h

```
/*
 * OF-CRIT: Critical IoT Objective Function for RPL
 * Diploma work: Modified routing method based on RPL for Critical IoT
 * Author: Pavlo Lysenko, KPI 2026
 */

#ifndef RPL_OF_CRIT_H_
#define RPL_OF_CRIT_H_

/* Weighted coefficients for OF-CRIT (sum must equal 1.0) */
#define OF_CRIT_ALPHA   40   /* ETX weight: 0.40 (x100) */
#define OF_CRIT_BETA    20   /* Load weight: 0.20 (x100) */
#define OF_CRIT_GAMMA   20   /* Battery weight: 0.20 (x100) */
#define OF_CRIT_DELTA   20   /* Priority weight: 0.20 (x100) */

/* Thresholds */
#define OF_CRIT_HYSTERESIS       192   /* Min improvement to switch parent */
#define OF_CRIT_ETX_THRESHOLD    35    /* ETX*10 threshold for FastFailover */
#define OF_CRIT_FAIL_THRESHOLD   3     /* Consecutive failures before failover */
#define OF_CRIT_MAX_BACKUP       3     /* Max backup parents */

/* Priority classes (x100) */
#define PRIORITY_BACKGROUND      0
#define PRIORITY_TELEMETRY       33
#define PRIORITY_MONITORING      67
#define PRIORITY_CRITICAL        100

extern const rpl_of_t of_crit;

/* Call from DIO parser when Extension TLV 0x09 is received */
void rpl_of_crit_update_nbr(rpl_nbr_t *nbr, uint8_t load, uint8_t battery,
                            uint8_t priority);

#endif /* RPL_OF_CRIT_H_ */
```

ДОДАТОК Б. Реалізація цільової функції OF-CRIT

Файл: os/net/routing/rpl-lite/rpl-of-crit.c

```
/*
 * OF-CRIT: Critical IoT Objective Function for RPL (Contiki-NG)
 *
 * Formula: OF-CRIT(v) = alpha*ETX + beta*Load + gamma*(1-Battery) + delta*Priority
 * Default weights: alpha=0.40, beta=0.20, gamma=0.20, delta=0.20
 *
 * Implementation:
 * - ETX read from link_stats (same as MRHOF)
 * - Load/Battery/Priority stored per-neighbor in static table
 * - Hysteresis prevents excessive parent switching
 *
 * Diploma: Modified RPL routing for Critical IoT Networks
 * Author: Pavlo Lysenko, NTUU KPI, 2026
 */

#include "net/routing/rpl-lite/rpl.h"
#include "net/link-stats.h"
#include "sys/log.h"
#include "rpl-of-crit.h"

#define LOG_MODULE "RPL"
#define LOG_LEVEL  LOG_LEVEL_RPL

/* Per-neighbor extended metrics (updated when DIO Extension TLV 0x09 arrives) */
#define OF_CRIT_MAX_NBR 16
static struct {
  uint8_t load;      /* Queue load: 0=idle, 255=full */
  uint8_t battery;   /* Battery: 0=empty, 255=full */
  uint8_t priority;  /* Traffic priority 0-100 */
  uint8_t valid;
} nbr_ext[OF_CRIT_MAX_NBR];

/* Map rpl_nbr_t* to index using address hash */
static int
nbr_idx(rpl_nbr_t *nbr)
{
  uintptr_t h = ((uintptr_t)nbr >> 3) & (OF_CRIT_MAX_NBR - 1);
  return (int)h;
}

void
rpl_of_crit_update_nbr(rpl_nbr_t *nbr, uint8_t load, uint8_t battery,
                       uint8_t priority)
{
  int i = nbr_idx(nbr);
  nbr_ext[i].load     = load;
  nbr_ext[i].battery  = battery;
  nbr_ext[i].priority = priority;
  nbr_ext[i].valid    = 1;
}

/*---------------------------------------------------------------------------*/
static uint16_t
calc_score(rpl_nbr_t *nbr)
{
  const struct link_stats *stats = rpl_neighbor_get_link_stats(nbr);
  uint16_t etx_raw = (stats != NULL) ? stats->etx : LINK_STATS_ETX_DIVISOR * 3;

  /* Normalize ETX 1..7 → 0..100 */
  uint32_t etx_n = ((uint32_t)etx_raw * 100) / (7 * LINK_STATS_ETX_DIVISOR);
  if(etx_n > 100) { etx_n = 100; }

  int i = nbr_idx(nbr);
  uint8_t ld  = nbr_ext[i].valid ? nbr_ext[i].load     : 100;
  uint8_t bat = nbr_ext[i].valid ? nbr_ext[i].battery  : 200;
  uint8_t pri = nbr_ext[i].valid ? nbr_ext[i].priority : 20;

  uint32_t ln = (uint32_t)ld  * 100 / 255;              /* Load 0-100 */
  uint32_t bn = 100 - ((uint32_t)bat * 100 / 255);      /* 1-Battery 0-100 */
```

```
  uint32_t pn = pri;                                        /* Priority 0-100 */

  uint32_t s = (OF_CRIT_ALPHA * etx_n + OF_CRIT_BETA * ln +
                OF_CRIT_GAMMA * bn    + OF_CRIT_DELTA * pn) / 100;

  return (uint16_t)(s > 0xFFFF ? 0xFFFF : s);
}

/*---------------------------------------------------------------------------*/
static void
reset(void)
{
  int i;
  for(i = 0; i < OF_CRIT_MAX_NBR; i++) { nbr_ext[i].valid = 0; }
  LOG_INFO("of-crit: reset (a=%d b=%d g=%d d=%d H=%d)\n",
           OF_CRIT_ALPHA, OF_CRIT_BETA, OF_CRIT_GAMMA,
           OF_CRIT_DELTA, OF_CRIT_HYSTERESIS);
}

static uint16_t
nbr_link_metric(rpl_nbr_t *nbr)
{
  return calc_score(nbr);
}

static int
nbr_has_usable_link(rpl_nbr_t *nbr)
{
  const struct link_stats *stats = rpl_neighbor_get_link_stats(nbr);
  if(stats == NULL) { return 0; }
  return stats->etx <= (uint16_t)(OF_CRIT_ETX_THRESHOLD * LINK_STATS_ETX_DIVISOR / 10);
}

static int
nbr_is_acceptable_parent(rpl_nbr_t *nbr)
{
  return nbr_has_usable_link(nbr);
}

static uint16_t
nbr_path_cost(rpl_nbr_t *nbr)
{
  if(nbr->rank == RPL_INFINITE_RANK) { return RPL_INFINITE_RANK; }
  return (uint16_t)MIN((uint32_t)nbr->rank + nbr_link_metric(nbr), 0xFFFF);
}

static rpl_rank_t
rank_via_nbr(rpl_nbr_t *nbr)
{
  uint16_t step;
  if(nbr == NULL || nbr->rank == RPL_INFINITE_RANK) { return RPL_INFINITE_RANK; }
  step = curr_instance.min_hoprankinc +
         (uint16_t)((uint32_t)nbr_link_metric(nbr) *
                    3 * curr_instance.min_hoprankinc / 100);
  return (rpl_rank_t)MIN((uint32_t)nbr->rank + step, RPL_INFINITE_RANK);
}

static rpl_nbr_t *
best_parent(rpl_nbr_t *p1, rpl_nbr_t *p2)
{
  uint16_t c1, c2;
  if(p1 == NULL) { return p2; }
  if(p2 == NULL) { return p1; }
  c1 = nbr_path_cost(p1);
  c2 = nbr_path_cost(p2);
  if(p1 == curr_instance.dag.preferred_parent) {
    return c1 <= c2 + OF_CRIT_HYSTERESIS ? p1 : p2;
  }
  if(p2 == curr_instance.dag.preferred_parent) {
    return c2 <= c1 + OF_CRIT_HYSTERESIS ? p2 : p1;
  }
  return c1 <= c2 ? p1 : p2;
}

static void
```

```
update_metric_container(void)
{
  curr_instance.mc.type = RPL_DAG_MC_NONE;
}

/*---------------------------------------------------------------------------*/
const rpl_of_t of_crit = {
  reset,
  nbr_link_metric,
  nbr_has_usable_link,
  nbr_is_acceptable_parent,
  nbr_path_cost,
  rank_via_nbr,
  best_parent,
  update_metric_container,
  RPL_OCP_MRHOF
};
```

ДОДАТОК В. Конфігураційний файл сценарію 1 — нормальний режим

Файл: simulation/scenario1_normal.csc (формат Cooja XML)

Файл описує топологію симуляції (26 вузлів), параметри радіосередовища UDGM (TX = 50 м, p_tx = p_rx = 0,95) та скрипт ScriptRunner із тайм-аутом 1800 с. Нижче наведено повний вміст файлу.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Cooja Simulation: Scenario 1 — Normal Mode
  Topology: Grid 5x5 (25 nodes, spacing 30m), uniform traffic (1 pkt/10s)
  Protocol: RPL with OF-CRIT objective function
  Duration: 1800s (30 minutes), 10 seeds
  Diploma: Pavlo Lysenko, NTUU KPI, 2026
-->
<simconf>
  <simulation>
    <title>Scenario 1 — Normal Mode — OF-CRIT</title>
    <randomseed>123456</randomseed>
    <motedelay_us>1000000</motedelay_us>
    <radiomedium>
      org.contikios.cooja.radiomediums.UDGM
      <transmitting_range>50.0</transmitting_range>
      <interference_range>100.0</interference_range>
      <success_ratio_tx>0.95</success_ratio_tx>
      <success_ratio_rx>0.95</success_ratio_rx>
    </radiomedium>
    <events>
      <logoutput>40000</logoutput>
    </events>

    <!-- Root (DODAG root / border router) -->
    <motetype>
      org.contikios.cooja.contikimote.ContikiMoteType
      <identifier>root</identifier>
      <description>RPL Root Node</description>
      <source>[CONTIKI_DIR]/examples/rpl-udp/udp-server.c</source>
      <commands>$(MAKE) -j$(CPUS) udp-server.cooja TARGET=cooja</commands>
      <moteinterface>org.contikios.cooja.interfaces.Position</moteinterface>
      <moteinterface>org.contikios.cooja.interfaces.Battery</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiVib</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiMoteID</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiRS232</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiBeeper</moteinterface>
      <moteinterface>org.contikios.cooja.interfaces.RimeAddress</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiIPAddress</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiRadio</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiButton</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiClock</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiLED</moteinterface>
      <moteinterface>org.contikios.cooja.interfaces.Mote2MoteRelations</moteinterface>
      <moteinterface>org.contikios.cooja.interfaces.MoteAttributes</moteinterface>
    </motetype>

    <!-- Sensor nodes (UDP clients) -->
    <motetype>
      org.contikios.cooja.contikimote.ContikiMoteType
```

```
          <identifier>sensor</identifier>
          <description>RPL Sensor Node (OF-CRIT)</description>
          <source>[CONTIKI_DIR]/examples/rpl-udp/udp-client.c</source>
          <commands>$(MAKE) -j$(CPUS) udp-client.cooja TARGET=cooja</commands>
          <moteinterface>org.contikios.cooja.interfaces.Position</moteinterface>
          <moteinterface>org.contikios.cooja.interfaces.Battery</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiVib</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiMoteID</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiRS232</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiBeeper</moteinterface>
          <moteinterface>org.contikios.cooja.interfaces.RimeAddress</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiIPAddress</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiRadio</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiButton</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiClock</moteinterface>

<moteinterface>org.contikios.cooja.contikimote.interfaces.ContikiLED</moteinterface>
          <moteinterface>org.contikios.cooja.interfaces.Mote2MoteRelations</moteinterface>
          <moteinterface>org.contikios.cooja.interfaces.MoteAttributes</moteinterface>
        </motetype>

        <!-- Root at center of grid -->
        <mote>
          <interface_config>
            org.contikios.cooja.interfaces.Position
            <x>150.0</x><y>150.0</y><z>0.0</z>
          </interface_config>
          <interface_config>
            org.contikios.cooja.contikimote.interfaces.ContikiMoteID
            <id>1</id>
          </interface_config>
          <motetype_identifier>root</motetype_identifier>
        </mote>

        <!-- 5x5 grid, 30m spacing, IDs 2-26 -->

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>60.0</x><y>60.0</y><z>0.
0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Conti
kiMoteID<id>2</id></interface_config><motetype_identifier>sensor</motetype_identifier></mo
te>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>90.0</x><y>60.0</y><z>0.
0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Conti
kiMoteID<id>3</id></interface_config><motetype_identifier>sensor</motetype_identifier></mo
te>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>120.0</x><y>60.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>4</id></interface_config><motetype_identifier>sensor</motetype_identifier></m
ote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>150.0</x><y>60.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>5</id></interface_config><motetype_identifier>sensor</motetype_identifier></m
ote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>180.0</x><y>60.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>6</id></interface_config><motetype_identifier>sensor</motetype_identifier></m
ote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>60.0</x><y>90.0</y><z>0.
0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Conti
kiMoteID<id>7</id></interface_config><motetype_identifier>sensor</motetype_identifier></mo
te>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>90.0</x><y>90.0</y><z>0.
```

```
0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Conti
kiMoteID<id>8</id></interface_config><motetype_identifier>sensor</motetype_identifier></mo
te>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>120.0</x><y>90.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>9</id></interface_config><motetype_identifier>sensor</motetype_identifier></m
ote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>150.0</x><y>90.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>10</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>180.0</x><y>90.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>11</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>60.0</x><y>120.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>12</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>90.0</x><y>120.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>13</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>120.0</x><y>120.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>14</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>150.0</x><y>120.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>15</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>180.0</x><y>120.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>16</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>60.0</x><y>150.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>17</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>90.0</x><y>150.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>18</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>120.0</x><y>150.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>19</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>180.0</x><y>150.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>20</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>60.0</x><y>180.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>21</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>90.0</x><y>180.0</y><z>0
.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Cont
ikiMoteID<id>22</id></interface_config><motetype_identifier>sensor</motetype_identifier></
mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>120.0</x><y>180.0</y><z>
```

```
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>23</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>150.0</x><y>180.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>24</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>180.0</x><y>180.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>25</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

<mote><interface_config>org.contikios.cooja.interfaces.Position<x>210.0</x><y>150.0</y><z>
0.0</z></interface_config><interface_config>org.contikios.cooja.contikimote.interfaces.Con
tikiMoteID<id>26</id></interface_config><motetype_identifier>sensor</motetype_identifier><
/mote>

     </simulation>

     <!-- ScriptRunner: log all output, stop at 1800s -->
     <plugin>
       org.contikios.cooja.plugins.ScriptRunner
       <plugin_config>
         <script>
   TIMEOUT(1800000, log.testOK());

   timeout_function = function() {
     log.log("Simulation completed at " + time/1000000 + "s\n");
     log.testOK();
   };

   while(true) {
     log.log(time + " " + id + " " + msg + "\n");
     YIELD();
   }
         </script>
         <active>true</active>
       </plugin_config>
       <width>600</width><z>0</z><height>400</height>
       <location_x>0</location_x><location_y>0</location_y>
     </plugin>
   </simconf>
```

ДОДАТОК Г. Зведена таблиця результатів імітаційного моделювання

Таблиця Г.1 — Порівняння показників якості обслуговування MRHOF та OF-CRIT у чотирьох сценаріях (Cooja, 26 вузлів, 1800 с, seed 123456, UDGM p=0,95).

Сценарій	Метрика	MRHOF	OF-CRIT	Покращення
Сц. 1 Нормальна робота	PDR, % Затримка, мс σ затримки, мс	99,20 ± 0,50 135,0 ± 72,0 —	99,89 ± 0,20 121,0 ± 66,0 —	**+0,69 в.п.** **−14,0 мс** —
Сц. 2 Мішаний трафік (30 % крит.)	PDR, % Затримка, мс PDR крит. трафіку, %	97,80 ± 1,20 148,0 ± 85,0 97,20	99,10 ± 0,50 128,0 ± 71,0 99,50	**+1,30 в.п.** **−20,0 мс** **+2,30 в.п.**
Сц. 3 Відмова вузлів (3 з 25)	PDR, % Затримка, мс Час відновлення, с	96,50 ± 2,10 168,0 ± 95,0 > 120	99,80 ± 0,30 121,9 ± 66,6 < 60	**+3,30 в.п.** **−46,1 мс** **> 2×**
Сц. 4 Деградація каналу (ETX ↑ 300–900 с)	PDR, % Затримка, мс PDR під час деград.	89,30 ± 3,50 203,0 ± 112 85,10	94,20 ± 2,10 141,0 ± 82,0 91,30	**+4,90 в.п.** **−62,0 мс** **+6,20 в.п.**

Примітка. Значення MRHOF для сценаріїв 2 і 4 отримано шляхом масштабування еталонних результатів з [8, 11, 15] до параметрів даної топології.

ДОДАТОК Д. Скрипт обробки результатів симуляції

Файл: simulation/parse_results.py

Скрипт виконує синтаксичний аналіз лог-файлів Cooja та обчислює коефіцієнт доставки пакетів (PDR), наскрізну затримку (мс) та часові мітки подій для кожного сценарію. Результати зберігаються у файлі summary.json.

```
#!/usr/bin/env python3
"""
Parse Cooja simulation log files and compute PDR, latency, parent changes.
Usage: python3 parse_results.py results/

Log format (actual Contiki-NG rpl-udp):
  <time_us> <node> [INFO: App       ] Sending request <seq> to <addr>
  <time_us> 1      [INFO: App       ] Received request '<msg>' from <addr>
  <time_us> 1      [INFO: App       ] Sending response.
  <time_us> <node> [INFO: App       ] Received response '<msg>' from <addr>
  <time_us> <node> EVENT node_8_killed  (scenario 3)
"""
import sys
import os
import re
import json
import statistics
from collections import defaultdict


def parse_log(log_path):
    sent = {}        # (node, seq) -> time_us
    received = {}    # (node, seq) -> time_us
    latencies = []
    events = []

    # Regex patterns
    re_send = re.compile(r'^(\d+)\s+(\d+)\s+\[INFO: App\s*\]\s+Sending request (\d+) to
')
        re_recv  =  re.compile(r'^(\d+)\s+(\d+)\s+\[INFO:  App\s*\]\s+Received  response
\'hello (\d+)\' from ')
    re_event = re.compile(r'^(\d+)\s+EVENT (\S+)')

    with open(log_path, 'r', errors='replace') as f:
        for line in f:
            m = re_send.match(line)
            if m:
                t, node, seq = int(m.group(1)), int(m.group(2)), int(m.group(3))
                sent[(node, seq)] = t
                continue

            m = re_recv.match(line)
            if m:
                t, node, seq = int(m.group(1)), int(m.group(2)), int(m.group(3))
                received[(node, seq)] = t
                if (node, seq) in sent:
                    latencies.append((t - sent[(node, seq)]) / 1000.0)  # us -> ms
                continue

            m = re_event.match(line)
            if m:
                events.append({'time_s': int(m.group(1)) / 1e6, 'name': m.group(2)})

    total_sent = len(sent)
    total_received = len(received)
    pdr = total_received / total_sent * 100 if total_sent > 0 else 0.0
    avg_lat = statistics.mean(latencies) if latencies else 0.0
    std_lat = statistics.stdev(latencies) if len(latencies) > 1 else 0.0
    min_lat = min(latencies) if latencies else 0.0
    max_lat = max(latencies) if latencies else 0.0

    # Per-node PDR
```

```
        nodes_sent = defaultdict(int)
        nodes_recv = defaultdict(int)
        for (node, seq) in sent:
            nodes_sent[node] += 1
        for (node, seq) in received:
            nodes_recv[node] += 1
        per_node_pdr = {
            n: round(nodes_recv[n] / nodes_sent[n] * 100, 1)
            for n in nodes_sent
        }

        return {
            'total_sent': total_sent,
            'total_received': total_received,
            'pdr': round(pdr, 2),
            'latency_mean_ms': round(avg_lat, 1),
            'latency_std_ms': round(std_lat, 1),
            'latency_min_ms': round(min_lat, 1),
            'latency_max_ms': round(max_lat, 1),
            'latency_samples': len(latencies),
            'events': events,
            'per_node_pdr': per_node_pdr,
        }


    def main(results_dir):
        scenarios = {}

        log_files = [f for f in sorted(os.listdir(results_dir)) if f.endswith('.log')]
        if not log_files:
            print(f"No .log files found in {results_dir}")
            sys.exit(1)

        for fname in log_files:
            fpath = os.path.join(results_dir, fname)
            print(f"\nParsing: {fname}")
            m = parse_log(fpath)

            print(f"  Sent:        {m['total_sent']} packets")
            print(f"  Received:    {m['total_received']} packets")
            print(f"  PDR:         {m['pdr']:.2f}%")
             print(f"  Latency:     {m['latency_mean_ms']:.1f} ± {m['latency_std_ms']:.1f}
ms"
                  f"  [min={m['latency_min_ms']:.1f}, max={m['latency_max_ms']:.1f}]")
            print(f"  Samples:     {m['latency_samples']}")

            if m['events']:
                print(f"  Events:      {m['events']}")

            # Nodes with PDR < 90%
            bad_nodes = {n: p for n, p in m['per_node_pdr'].items() if p < 90.0}
            if bad_nodes:
                print(f"  Nodes <90% PDR: {bad_nodes}")

            # Determine scenario key from filename
            if 'scenario1' in fname:
                key = 'scenario1_normal'
            elif 'scenario3' in fname:
                key = 'scenario3_failures'
            else:
                key = fname.replace('.log', '')

            scenarios[key] = m

        # Save JSON summary
        out_path = os.path.join(results_dir, 'summary.json')
        with open(out_path, 'w') as f:
            json.dump(scenarios, f, indent=2, ensure_ascii=False)
        print(f"\nSummary saved to: {out_path}")

        # Print comparison table if both scenarios present
        if 'scenario1_normal' in scenarios and 'scenario3_failures' in scenarios:
            s1 = scenarios['scenario1_normal']
            s3 = scenarios['scenario3_failures']
            print("\n=== COMPARISON TABLE ===")
```

```
                    print(f"{'Metric':<30} {'Scenario 1 (Normal)':>22} {'Scenario 3
(Failures)':>22}")
            print("-" * 76)
            print(f"{'PDR (%)':<30} {s1['pdr']:>22.2f} {s3['pdr']:>22.2f}")
                     print(f"{'Latency mean (ms)':<30} {s1['latency_mean_ms']:>22.1f}
{s3['latency_mean_ms']:>22.1f}")
                      print(f"{'Latency std (ms)':<30} {s1['latency_std_ms']:>22.1f}
{s3['latency_std_ms']:>22.1f}")
                      print(f"{'Latency max (ms)':<30} {s1['latency_max_ms']:>22.1f}
{s3['latency_max_ms']:>22.1f}")
            print(f"{'Packets sent':<30} {s1['total_sent']:>22} {s3['total_sent']:>22}")
                         print(f"{'Packets received':<30} {s1['total_received']:>22}
{s3['total_received']:>22}")


    if __name__ == '__main__':
        results_dir = sys.argv[1] if len(sys.argv) > 1 else 'results'
        if not os.path.isdir(results_dir):
            print(f"Directory not found: {results_dir}")
            sys.exit(1)
        main(results_dir)
```