

Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова
Національна академія наук України
Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова
Національна академія наук України

Кваліфікаційна наукова праця
на правах рукопису

Пучко Іван Вікторович

УДК 621.311.161:(004.72 +004.75):004.22.


ДИСЕРТАЦІЯ

**Агентне моделювання високоінтелектуальних енергетичних мереж із
використанням неконфліктних реплікованих структур даних для опису
станів системи**

122 – Комп'ютерні науки
12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії.

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


І.В. Пучко

Науковий керівник:

Мохор Володимир Володимирович, член-кореспондент НАН України, доктор
технічних наук, професор

Київ — 2025

АНОТАЦІЯ

Пучко І.В. Агентне моделювання високоінтелектуальних енергетичних мереж із використанням неконфліктних реплікованих структур даних для опису станів системи. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 "Комп'ютерні науки". – Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, Київ, 2025.

Дисертація є комплексним та завершеним дослідженням, спрямованим на розробку, формалізацію, аналіз, оцінку та практичне застосування агентної моделі високоінтелектуальних енергетичних мереж на основі подання станів системи за допомогою неконфліктних реплікованих структур даних.

Інформаційна база дослідження ґрунтувалася на міжнародних і національних стандартах, наукових працях науковців України та світу, матеріалах конференцій і презентаціях провідних спеціалістів.

Результати дисертаційної роботи є підґрунтям для подальших теоретичних і практичних наукових розробок у дослідженні агентного моделювання високоінтелектуальних енергетичних мереж та розв'язанні оптимізаційних задач.

Дисертаційна робота складається зі вступу, чотирьох розділів, поділених на підрозділи, висновків, списку використаних джерел і додатків.

Об'єктом дослідження є процес забезпечення ефективної синхронізації станів у високоінтелектуальних електричних мережах на основі неконфліктних реплікованих структур даних.

Предметом дослідження є агентне моделювання високоінтелектуальних енергетичних мереж із використанням неконфліктних реплікованих структур даних для опису станів системи.

У вступі обґрунтовано актуальність теми дослідження, сформульовано мету, поставлено завдання, визначено предмет, об'єкт та методи дослідження, вказано наукову новизну та практичне значення отриманих результатів, а також

наведено відомості про апробацію та публікації основних результатів дисертації. Описано структуру та обсяг дисертаційної роботи.

В першому розділі дисертації наведено аналітичний огляд актуальних наукових досліджень у галузі високоінтелектуальних енергетичних мереж (далі - ВІЕМ) та підходів до їх моделювання і управління. Розглянуто принципи агентного підходу як інструменту для розподіленого прийняття рішень та координації у складних кіберфізичних середовищах, визначено типи агентів, їх внутрішню організацію та архітектурні засади побудови мультиагентних систем. Аналізуються методи забезпечення узгодженості станів у розподілених системах, окреслюються їхні обмеження та обґрунтовується необхідність розробки підходу синхронізації станів агентів мультиагентної системи на основі неконфліктних реплікованих структур даних (англ.: Conflict-Free Replicated Data Types, далі CRDT). Розглянуто теоретичні засади моделі акторів та функціональної парадигми програмування як інструментів побудови високонавантажених розподілених систем. Запропоновано поєднання акторної моделі, функціональної парадигми програмування та представлення станів вузлів системи на основі CRDT при створенні агентної моделі ВІЕМ.

В другому розділі зроблено формалізацію та спроектовано агентну модель розподіленої енергетичної мережі. Розроблено підхід синхронізації станів агентів мультиагентної системи на основі CRDT. Описано структуру агентів системи з визначеними статичними та динамічними характеристиками, виокремлено обмежений простір станів, що відображає ключові режими функціонування. Побудовано множину станів та правила переходів між ними. Розроблено узагальнене представлення ВІЕМ за допомогою CRDT.

У третьому розділі представлено розробку симуляційної моделі розподіленої енергетичної мережі. Обґрунтовано доцільність використання Scala та Akka як інструментів для поєднання акторної моделі та функціональної парадигми програмування для забезпечення необхідного рівня абстракції при побудові симуляційної моделі. Подано опис архітектури симулятора, що відображає принципи взаємодії агентів і механізми поширення станів у

мультіагентному середовищі. Деталізовано імплементацію симуляційної моделі, включно з процесами моделювання поведінки агентів, узгодження їхніх станів на основі безконфліктних структур даних та забезпечення масштабованості системи.

В четвертому розділі здійснено експериментальну перевірку запропонованої моделі синхронізації станів у розподілених енергетичних системах. Сформульовано постановку експериментальних досліджень та обґрунтовано методику їх проведення, що включає вибір сценаріїв моделювання, визначення параметрів симуляцій та підходи до збору й аналізу результатів. Аналіз результатів експериментів підтвердив ефективність запропонованої моделі щодо забезпечення узгодженості станів, її масштабованість та стійкість до мережевих відмов у різних сценаріях функціонування ВІЕМ. Експерименти підтвердили гнучкість запропонованої моделі та доцільність її використання при агентному моделюванні ВІЕМ.

У висновках наведено отримані наукові та практичні результати дослідження.

Практична цінність отриманих результатів підтверджена експериментами, виконаними в межах дисертації із застосуванням розробленого програмне забезпечення для моделювання ВІЕМ на основі CRDT, яке забезпечує підтримку рішень щодо балансування, верифікацію децентралізованих алгоритмів керування та відтворюване оцінювання стійкості до мережевих затримок і відмов.

Ключові слова: високоінтелектуальні енергетичні мережі, агентне моделювання, розподілені системи, децентралізовані системи, електроенергетична мережа, інтелектуальні системи, кіберфізичні системи, математична модель, високонавантажені системи, безконфліктні репліковані типи даних, реплікація, консенсус, стан, простір станів.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

1. **I.V. Пучко**, А.М. Примушко, Г.О. Кравцов, Розробка методичних рекомендацій використання функціональної парадигми програмування в мові Scala. *Електронне моделювання*, Том 43, № 6 (2021) с. 95-106, [doi: 10.15407/emodel.43.06.095](#). Фахове видання категорії Б. (Особистий внесок – структурування та написанні методичних рекомендацій).
2. А.М. Примушко, **I.V. Пучко**, М.С. Ярошинський, Д.П. Сінько, Програмний дизайн розподіленої високонавантаженої системи електроенергетичної мережі на базі моделі акторів із застосуванням смарт-контрактів; *Електронне моделювання*, Том 46, № 3 (2024) с. 57-72 [doi: 10.15407/emodel.46.03.057](#). Фахове видання категорії Б. (Особистий внесок – брав участь в розробці високорівневого дизайну розподіленої електроенергетичної системи на основі моделі акторів та кластерної топології).
3. A. Prymushko, **I. Puchko**, M. Yaroshynskyi, D. Sinko, H. Kravtsov, and V. Artemchuk, ‘Efficient State Synchronization in Distributed Electrical Grid Systems Using Conflict-Free Replicated Data Types’, *IoT*, vol. 6, no. 1, p. 6, Jan. 2025, [doi: 10.3390/iot6010006](#). **Indexed in Scopus Q1**. (Особистий внесок – брав участь в розробці моделі розподілених електроенергетичних систем, що базується на неконфліктних реплікованих типах даних. Спроектував і виконав основну частину розроблення програмного забезпечення “Vigilant Hawk” для симуляції розподілених електроенергетичних систем на базі акторної моделі та інструментарію Akka. Провів експериментальні дослідження, отримав та обробив результати, виконав їхню візуалізацію у вигляді графіків).
4. O. Sirotkin, A. Prymushko, **I. Puchko**, H. Kravtsov, M. Yaroshynskyi, V. Artemchuk. Parallel Simulation Using Reactive Streams: Graph-Based Approach for Dynamic Modeling and Optimization. *Computation*, vol. 13, no. 5, p. 103, Apr. 2025, [doi: 10.3390/computation13050103](#). **Indexed in Scopus**

- Q2.** (Особистий внесок – брав участь в оптимізації симуляції на основі графів та реактивних потоків за допомогою визначених функцій переходу).
5. M. Yaroshynskyi, A. Prymushko , **I. Puchko**, O. Sirotkin, and D. Sinko, ‘Akka as a tool for modelling and managing a smart grid system’, *Journal of Edge Computing*, vol. 4, no. 1, pp. 105–115, May 2025, doi: 10.55056/jec.822. **Indexed in Scopus.** (Особистий внесок – брав участь у проектуванні моделі управління інтелектуальною мережею, побудованої на ієрархічній структурі акторів. Провів експериментальні дослідження, обробив та візуалізував результати).
 6. M. Yaroshynskyi, **I. Puchko**, A. Prymushko , H. Kravtsov, and V . Artemchuk, ‘Investigating the Evolution of Resilient Microservice Architectures: A Compatibility-Driven Version Orchestration Approach’, *Digital*, vol. 5, no. 3, p. 27, July 2025, doi: 10.3390/digital5030027. **Indexed in Scopus Q2.** (Особистий внесок – брав участь у дослідженні та порівняльному аналізі існуючих стратегій управління еволюцією API. Розробив експериментальні сервіси на базі Spring Boot з використанням gRPC та для перевірки моделі).
 7. **I.B. Пучко**, А.М. Примушко, М.С. Ярошинський, Г.О. Кравцов, ‘Підвищення резильєнтності динамічних систем при синхронізації станів за допомогою CRDT’, Матеріали науково-практичної конференції ‘Резильєнтність динамічних систем, с. 50–52, Київ, Україна, 2024 URL: <https://ipme.kiev.ua/konferencii/naukovo-praktichna-konferenciya-rds-2024/>.
 8. A. Prymushko , M. Yaroshynskyi, and **I. Puchko**, ‘Representation and synchronization of states of distributed electrical grid systems based on conflict free replicated data types’, in 2024 14th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece: IEEE, Oct. 2024, pp. 1–5. doi: 10.1109/DESSERT65323.2024.11122143. **Indexed in Scopus.**

ANNOTATION

Puchko I.V. Agent-based modeling of smart grids using conflict-free replicated data types for system state representation – Qualification scientific work with the manuscript copyright.

The thesis on receipt of doctor of philosophy (PhD) scientific degree of the specialty 122 “Computer science” . – G.E. Pukhov Institute for Modelling in Energy Engineering of the NAS of Ukraine, Kyiv, 2025.

This dissertation is a comprehensive and complete study aimed at the development, formalisation, analysis, evaluation and practical application of an agent-based model of smartgrid systems, grounded in the representation of system states via Conflict-Free Replicated Data Types.

The research draws on international and national standards, scholarly works by Ukrainian and international researchers, conference proceedings, and presentations by leading experts.

The results provide a foundation for further theoretical and applied research in agent-based modelling of smart grids and for solving optimisation problems.

The dissertation comprises an introduction, four chapters subdivided into sections, conclusions, a list of references, and an appendix.

The object of the research is the process of ensuring efficient state synchronisation in smart grids using CRDTs.

The subject of the research is agent-based modelling of smart grids employing CRDTs to represent system states.

The introduction substantiates the relevance of the topic, states the aim, sets out the objectives, defines the subject, object, and research methods, highlights the scientific novelty and practical significance of the results obtained, and provides information on the validation and publications of the main findings. The structure and scope of the dissertation are also outlined.

The first chapter presents an analytical review of pertinent research on smart grids and approaches to their modelling and management. It considers the principles of the agent-based approach as a tool for distributed decision-making and

coordination in complex cyber-physical environments, identifies types of agents, their internal organisation, and architectural principles for building multi-agent systems. Methods for ensuring state consistency in distributed systems are analysed, their limitations outlined, and the need to develop an approach to multi-agent state synchronisation based on CRDTs is substantiated. The theoretical foundations of the actor model and the functional programming paradigm are examined as tools for constructing large-scale distributed systems. A combined approach integrating the actor model, functional programming, and CRDT-based state representation is proposed for building an agent-based model of smartgrids.

The second chapter formalises and designs the agent-based model of a distributed energy network. A state-synchronisation approach for a multi-agent system based on CRDTs is developed. The structure of the agents is described with defined static and dynamic characteristics, and a bounded state space is distinguished to reflect key operating modes. The set of states and transition rules is constructed. A generalised representation of the smartgrid is developed using CRDTs.

The third chapter presents the development of a simulation model of the distributed energy network. It substantiates the suitability of Scala and Akka as tools that combine the actor model with the functional programming paradigm to provide the required level of abstraction for building the simulation model. It provides a description of the simulator architecture that reflects the principles of agent interaction and the mechanisms of state propagation in a multi-agent environment. It details the implementation of the simulation model, including processes for modelling agent behavior, reconciling their states based on conflict-free replicated data types, and ensuring system scalability.

The fourth chapter provides the experimental verification of the proposed state-synchronisation model in distributed energy systems. The formulation of the experiments and the methodology of their implementation are substantiated, including the choice of scenarios, the definition of simulation parameters, and approaches to data collection and analysis. The results confirm the effectiveness of the proposed model in maintaining state consistency, its scalability, and its resilience

to network failures under various smartgrid operating scenarios. The experiments validate the model's flexibility and the feasibility of its application in agent-based modelling of such systems.

The conclusions summarise the scientific and practical outcomes of the research.

The practical value of the results is confirmed by experiments conducted using the software developed within the dissertation for modelling smartgrids based on CRDTs. The software supports decision-making for power balancing, verification of decentralised control algorithms, and reproducible assessment of resilience to network delays and failures.

Keywords: smartgrids, agent-based modelling, distributed systems, decentralised systems, power grid, intelligent systems, cyber-physical systems, mathematical model, high-load systems, conflict-free replicated data types (CRDTs), replication, consensus, state, state space.

ЗМІСТ

| | |
|---|----|
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ ТА СКОРОЧЕНЬ..... | 13 |
| ВСТУП..... | 14 |
| РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО АГЕНТНОГО МОДЕЛЮВАННЯ..... | 20 |
| 1.1. Агентний підхід до моделювання високоінтелектуальних енергетичних мереж..... | 21 |
| 1.1.1. Основні принципи агентного підходу..... | 23 |
| 1.1.2. Агентне моделювання у контексті високоінтелектуальних енергетичних мереж..... | 25 |
| 1.1.3. Типи агентів та принципи їх внутрішньої організації..... | 26 |
| 1.1.4. Архітектурні підходи до побудови мультиагентних систем..... | 27 |
| 1.1.5. Опис станів агентів у мультиагентних системах..... | 29 |
| 1.2. Методи забезпечення узгодженості станів у розподілених системах..... | 30 |
| 1.2.1. Поняття узгодженості в розподілених обчисленнях..... | 31 |
| 1.2.2. Традиційні підходи до підтримання узгодженості..... | 32 |
| 1.2.3. Розподілені алгоритми узгодження..... | 34 |
| 1.2.4. Моделі остаточної узгодженості..... | 37 |
| 1.2.5. Обмеження існуючих підходів у контексті високоінтелектуальних енергетичних мереж..... | 39 |
| 1.3. Неконфліктні репліковані структури даних у моделюванні енергетичних процесів..... | 41 |
| 1.3.1. Поняття неконфліктних реплікованих структур даних..... | 41 |
| 1.3.3. Класифікація неконфліктних реплікованих структур даних..... | 45 |
| 1.3.4. Дослідження практичного використання неконфліктних реплікованих структур даних..... | 48 |
| 1.3.5. Обмеження використання неконфліктних реплікованих структур | |

| | |
|--|----|
| | 11 |
| даних..... | 50 |
| 1.4. Модель акторів та функціональна парадигма програмування у високонавантажених системах..... | 51 |
| 1.4.1. Концепція моделі акторів..... | 52 |
| 1.4.2. Функціональна парадигма в контексті розподілених систем..... | 54 |
| Висновки розділу 1..... | 56 |
| РОЗДІЛ 2. ФОРМАЛІЗАЦІЯ ТА ПРОЄКТУВАННЯ АГЕНТНОЇ МОДЕЛІ РОЗПОДІЛЕНОЇ ЕНЕРГЕТИЧНОЇ МЕРЕЖІ..... | 58 |
| 2.1. Формалізація агентної моделі..... | 59 |
| 2.3. Підхід синхронізації станів агентів на основі неконфліктних реплікованих структур даних..... | 61 |
| Висновки розділу 2..... | 69 |
| РОЗДІЛ 3. РОЗРОБКА СИМУЛЯЦІЙНОЇ МОДЕЛІ РОЗПОДІЛЕНОЇ ЕНЕРГЕТИЧНОЇ МЕРЕЖІ..... | 70 |
| 3.1. Вибір інструментів реалізації..... | 70 |
| 3.1.1. Мова програмування Scala..... | 71 |
| 3.1.2. Фреймворк Akka..... | 72 |
| 3.2. Архітектура симулятора: взаємодія агентів та поширення станів..... | 75 |
| 3.3. Імплементация симуляційної моделі..... | 82 |
| Висновки розділу 3..... | 85 |
| РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА МОДЕЛІ СИНХРОНІЗАЦІЇ СТАНІВ У РОЗПОДІЛЕНИХ ЕНЕРГЕТИЧНИХ СИСТЕМАХ..... | 86 |
| 4.1. Постановка експериментальних досліджень..... | 86 |
| 4.2. Методика проведення експериментів..... | 87 |
| 4.2.1. Вибір сценаріїв моделювання..... | 87 |
| 4.2.2. Параметри симуляцій: кількість агентів, затримки мережі..... | 89 |
| 4.2.3. Методи збору та аналізу результатів..... | 91 |

| | |
|---|-----|
| | 12 |
| 4.3. Результати експериментальних досліджень..... | 92 |
| Висновки розділу 4..... | 95 |
| ВИСНОВКИ..... | 97 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... | 99 |
| ДОДАТОК А..... | 112 |
| ПРОДОВЖ. ДОД. А..... | 114 |
| ДОДАТОК Б..... | 115 |

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ ТА СКОРОЧЕНЬ

| | |
|-------------|---|
| БАС | багатоагентна система |
| ВДЕ | відновлювані джерела енергії |
| BIEM | високоінтелектуальні енергетичні мережі |
| CmRDT | Commutative Replicated Data Type |
| CRDT | Conflict-Free Replicated Data Types |
| CvRDT | Convergent Replicated Data Type |
| LWWMap | Last-Writer-Wins Map |
| LWWRegister | Last-Writer-Wins Register |
| ORSet | Observed-Remove Set |

ВСТУП

Обґрунтування вибору теми дослідження. Електроенергетична галузь переживає радикальну трансформацію від небагатьох централізованих теплових та атомних електростанцій до розподіленої генерації у вигляді багатьох децентралізованих малих екологічно чистих виробників. Завдяки розвитку альтернативних джерел енергії, таких як фотовольтаїчні сонячні панелі, вітрові турбіни та мікротурбіни на природному газі, окремі домогосподарства й бізнеси встановлюють нове обладнання для виробництва та зберігання електроенергії, щоб частково або повністю покривати власні потреби й навіть продавати надлишки іншим споживачам у мереж [1]. Такий перехід до децентралізованої генерації без докорінної зміни існуючої системи спричинив еволюцію всієї системи в напрямку концепції «високоінтелектуальних енергетичних систем» [2].

Дана концепція передбачає розподілену систему виробництва електроенергії з багатьма інтелектуальними виробниками та споживачами, тобто такими енергетичними об'єктами, що оснащені вбудованими системами вимірювання, прогнозування та автоматизованого управління, здатними аналізувати власний стан і стан мережі, а також приймати рішення щодо виробництва, споживання чи зберігання енергії [1], [2], [3].

Експлуатація таких систем потребує програмно-обчислювальної інфраструктури, здатної забезпечити збір і обмін даними, узгодження дій та реалізацію функцій апаратно-програмних компонентів мережі. Вимоги до побудови таких рішень узгоджуються з ключовими положеннями концепції багатоагентних систем (далі - БАС): розподіленістю, автономністю прийняття рішень, асинхронністю взаємодії та масштабованістю [1], [2], [4].

Під багатоагентною системою розуміємо сукупність автономних інтелектуальних агентів, що взаємодіють у спільному середовищі для досягнення індивідуальних і/або спільних цілей. Агенти можуть бути реалізовані як програмно-апаратні або кіберфізичні об'єкти. Під

інтелектуальними агентами розуміємо автономні сутності, що сприймають середовище, мають цілі та обирають дії для їх досягнення, взаємодіючи з іншими. З огляду на це, БАС доцільно використати як інструмент моделювання ВІЕМ [1].

Узгодженість дій агентів ґрунтується на спільному баченні стану системи, тобто узгодженому поданні множини станів агентів системи, що потребує механізмів синхронізації та підтримання узгодженості даних на всіх вузлах.

У розподіленій системі сильна узгодженість гарантує, що всі клієнти мають однакове впорядковане бачення атомарних оновлень [5]. Водночас, за теоремою CAP [6], у разі розділення мережі неможливо одночасно забезпечити сильну узгодженість і доступність. За таких умов системи, що надають пріоритет сильній узгодженості, під час мережових розділень обмежують доступність і блокують операції до відновлення узгодженого стану [7]. Натомість підходи на основі CRDT віддають перевагу доступності, тоді як узгодженість досягається завдяки детермінованій збіжності реплік у межах слабших моделей через формальні правила злиття оновлень. Такий підхід підтримує передбачувану поведінку систем навіть за високих навантажень і нестабільного мережевого оточення. Застосування CRDT ґрунтується на ідемпотентності, асоціативності та комутативності операцій, що дає змогу досягати узгодженого стану незалежно від порядку застосування оновлень. Ці властивості спрощують керування станами вузлів і роблять процес синхронізації прозорим та передбачуваним. Особливо це важливо для енергетичних систем як об'єктів критичної інфраструктури, де збої чи розбіжності можуть мати значні негативні наслідки.

Таким чином, розроблення підходу до агентного моделювання високоінтелектуальних енергетичних мереж із використанням неконфліктних реплікованих структур даних для опису станів системи є актуальним науковим завданням.

Мета і завдання дослідження. Забезпечення ефективної синхронізації станів агентів у високоінтелектуальних енергетичних мережах шляхом застосування

неконфліктних реплікованих структур даних для опису станів системи. **Можна виділити наступні завдання дослідження :**

1. проаналізувати існуючі підходи до синхронізації інформації про стан мультиагентної системи у кожного агента.
2. розробити формальну модель представлення станів вузлів високоінтелектуальних енергетичних мережі на основі неконфліктних структур даних.
3. розробити підхід синхронізації станів агентів мультиагентної системи на основі CRDT.
4. розробити комплекс програмного забезпечення для агентного моделювання високоінтелектуальних енергетичних мереж, що ухвалюють локальні рішення та координуються через обмін повідомленнями із використанням неконфліктних структур даних.
5. експериментально підтвердити ефективність запропонованого підходу за допомогою розробленого програмного забезпечення.
6. визначити оптимальні параметри роботи системи в умовах різних мережових затримок.

Об'єктом дослідження є процес забезпечення ефективної синхронізації станів у високоінтелектуальних електричних мережах на основі неконфліктних реплікованих структур даних. Під ефективністю розуміється масштабованість та стійкість до збоїв.

Предмет дослідження – агентне моделювання високоінтелектуальних енергетичних мереж із використанням неконфліктних реплікованих структур даних для опису станів системи.

Методи дослідження. У дисертаційній роботі при розв'язанні поставлених наукових завдань використовувалися методи системного і функціонального аналізу, математичного моделювання, об'єктно-орієнтованого та

функціонального програмування, планування наукового експерименту та обробки його результатів тощо.

Наукова новизна. В дисертаційній роботі отримано наступні наукові результати:

1. Вперше запропоновано підхід до синхронізації станів агентів мультиагентної системи на основі неконфліктних структур даних, який, на відміну від централізованих/консенсусних підходів, забезпечує детерміновану збіжність без глобальної координації.
2. Вперше імплементовано агентну модель високоінтелектуальних енергетичних мереж на основі підходу синхронізації станів за допомогою неконфліктних реплікованих структур даних, що, на відміну від класичних підходів синхронізації, підвищує стійкість до відмов та забезпечує масштабованість та передбачувану поведінку під високими навантаженнями й мережевою нестабільністю.
3. Вперше проаналізовано вплив зростання мережових затримок на час поширення станів, представлених за допомогою неконфліктних реплікованих структур даних, що дало змогу експериментально підтвердити лінійну масштабованість системи.
4. Запропоновано використовувати акторну модель при моделюванні високоінтелектуальних енергетичних мереж, яка, відрізняється від існуючих відсутністю спільних структур синхронізації та локальним ухваленням рішень, що забезпечує передбачувану продуктивність і спрощує масштабування системи.

Практичне значення отриманих результатів полягає у наступному:

1. Розроблено комплект програмного забезпечення для моделювання високоінтелектуальних енергетичних мереж на основі неконфліктних реплікованих структур даних, яке забезпечує автоматизацію прийняття рішень щодо балансування, верифікацію децентралізованих алгоритмів

керування та відтворюване оцінювання стійкості до мережових затримок і відмов.

2. Визначено оптимальні параметри роботи розробленого програмного забезпечення за різних профілів мережових затримок для коректної синхронізації станів і мінімізації часу їх поширення.
3. Розроблене програмне забезпечення придатне до практичного застосування як інженерний інструмент для аналізу, проєктування та оптимізації моделей високоінтелектуальних енергетичних мереж. Крім того, розроблене ПЗ може бути використано при розробленні стратегій керування такими мережами.

Зв'язок роботи з науковими програмами, планами і темами. Тематика дисертаційної роботи відповідає пріоритетним напрямкам розвитку науки і техніки в Україні. Робота виконувалась відповідно до плану наукових досліджень Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, зокрема в рамках НДР «Розвиток наукових засад алгебраїчної теорії сильного штучного інтелекту стосовно кібернетичної безпеки об'єктів критичної інфраструктури в галузі енергетики» (№ ДР 0123U100913, 2023-2027 рр.) та «Розвиток розподіленої енергетики в умовах ринку електричної енергії України з використанням технологій та систем цифровізації. Розділ 1. Організаційні та математичні моделі взаємодії учасників децентралізованого ринку електроенергії» (№ ДР 0125U000237, 2025-2026 рр.). Дослідження дисертаційної роботи проведено під керівництвом чл.-кор. НАН України, д-р техн.наук, професора Мохора Володимира Володимировича.

Особистий внесок здобувача. Наукові результати дослідження, які виносяться на захист, одержані автором самостійно.

Публікації. Наукові результати та висновки дисертаційного дослідження підтверджуються публікаціями у наукових виданнях та їх апробацією на

науково-практичних конференціях. Опубліковано 8 наукових праць. У тому числі 2 наукових статей опубліковано у фахових виданнях України, 4 статті індексовані в міжнародній науково-інформаційній базі Scopus, 2 тез доповідей на всеукраїнських і міжнародній науково-технічних конференціях.

Апробація результатів дослідження. Результати дисертаційного дослідження доповідались на науково-практичних конференціях:

- **І.В. Пучко**, А.М. Примушко, М.С. Ярошинський, Г.О. Кравцов, ‘Підвищення резильєнтності динамічних систем при синхронізації станів за допомогою CRDT’, Матеріали науково-практичної конференції ‘Резильєнтність динамічних систем, с. 50–52, Київ, Україна, 2024 URL: <https://ipme.kiev.ua/konferencii/naukovo-praktichna-konferenciya-rds-2024/>
- A. Prymushko, M. Yaroshynskyi, and **I. Puchko**, ‘Representation and synchronization of states of distributed electrical grid systems based on conflict free replicated data types’, in 2024 14th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece: IEEE, Oct. 2024, pp. 1–5. doi: 10.1109/DESSERT65323.2024.11122143.

Indexed in Scopus.

Структура та обсяг роботи. Дисертація складається із вступу, чотирьох розділів, висновків, списку використаних джерел, додатку. Робота містить 115 сторінок, із них 98 сторінок основного тексту, 12 рисунків і 5 таблиць, список використаних джерел зі 109 найменувань на 13 сторінках.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО АГЕНТНОГО МОДЕЛЮВАННЯ

Численні дослідження окреслюють ВІЕМ як кіберфізичні системи з високим рівнем децентралізації та різнотипними учасниками системи. Попри те, що стохастичність попиту та пропозиції притаманна енергосистемам, зростання частки погодозалежних ВДЕ та електрифікації споживання суттєво підвищує мінливість і невизначеність, що ускладнює балансування, ринкові процеси та оперативне керування елементами мережі й висуває підвищені вимоги до відновлюваності, резилієнтності та кібербезпеки. За умов просторової розподіленості учасників системи, мережевих затримок і втрат та підвищених вимог до відмовостійкості централізовані схеми прийняття рішень є обмежено ефективними, оскільки підвищують латентність, ускладнюють масштабування та створюють єдину точку відмови. У цьому контексті мультиагентні системи (анг: Multi-Agent Systems, далі - БАС) позиціонуються як інструмент для децентралізованого прийняття рішень та координації у ВІЕМ, підкреслюючи їхню здатність масштабувати міркування та дії в умовах розподіленості й ризиків відмов чи атак [1], [8], [9], [10], [11], [12], [13], [14], [15].

Огляд в *Energies* (2024) [1] систематизує застосування БАС у чотирьох ключових напрямках: керування попитом і пропозицією; відновлення та самовідновлення; захист і безпека; симуляція та імплементація. Автори не лише агрегують методи й результати, а й формулюють відкриті питання для кожного напрямку та подають мета-аналіз [16] загальних викликів. Додатково деталізується спектр архітектур БАС — централізованих, децентралізованих та ієрархічних — і відповідні парадигми прийняття рішень та оптимізації, що добираються з огляду на специфіку завдання й операційний контекст. Серед наскрізних викликів виокремлюють вимоги до масштабованості та резилієнтності в умовах високої частки децентралізованої генерації і стохастичності ВДЕ. Додатково наявні виклики, пов'язані з різномірністю

інтерфейсів і протоколів, а також обмеження й ненадійність комунікацій (затримки, втрати, розділення мережі) за жорстких вимог роботи в реальному часі. Окремо наголошується на потребі в інтегрованих підходах до кібербезпеки та резилієнтності. Водночас відзначається брак еталонних сценаріїв, відкритих наборів даних і узгоджених метрик, що ускладнює відтворюваність і порівнюваність результатів. Відповідно окреслюються пріоритетні напрями подальших досліджень, зокрема розроблення методів із формальними гарантіями збіжності та стабільності, інтеграція агентних рішень із механізмами узгодженого подання стану, а також створення масштабованих відтворюваних симуляційних платформ і еталонних сценаріїв оцінювання.

У моделях ВІЕМ на основі БАС агенти виконують закріплені за ними завдання та функції де вибір правила прийняття рішень або методу оптимізації залежить від архітектури БАС та типу застосування.

Зокрема, централізований підхід має переваги, коли для всієї мережі потрібен єдиний критерій оптимізації й прийнятними є потенційні “вузькі місця” координації. Децентралізовані протоколи консенсусу, що спираються на інтенсивний обмін повідомленнями між сусідніми агентами, відповідають сценаріям із жорсткими вимогами до сильної узгодженості, однак мають притаманні обмеження: зростання латентності й комунікаційних накладних витрат, чутливість до розділення мережі та проблеми масштабованості. У сценаріях, що допускають слабшу або подієву узгодженість, застосовуються децентралізовані схеми координації без глобального консенсусу з локальною взаємодією агентів. У будь-якому випадку необхідною передумовою ефективності лишається узгоджене уявлення про релевантний стан системи.

1.1. Агентний підхід до моделювання високоінтелектуальних енергетичних мереж

Агенти у межах мультиагентного підходу розглядаються як інтелектуальні сутності, здатні діяти автономно та гнучко, приймати обґрунтовані рішення на основі вбудованих алгоритмів і накопиченого досвіду взаємодії із середовищем

[11]. Агенти характеризуються можливістю адаптації до змін середовища та здатністю до кооперації з іншими агентами для досягнення індивідуальних чи колективних цілей. На відміну від традиційних елементів керування, агенти наділені властивістю «раціональності», що передбачає вибір оптимальних дій з множини можливих відповідно до заданих умов.

Мультиагентна система визначається як сукупність агентів, що взаємодіють між собою та зовнішнім середовищем задля розв'язання завдань, що перевищують функціональні можливості окремого агента [11]. Таким чином, мультиагентна система надає можливість дослідити ефективні методи організації децентралізованого управління складними технічними об'єктами, до яких належать і ВІЕМ.

ВІЕМ у сучасному розумінні є інтегративною архітектурою, що поєднує традиційну енергетичну інфраструктуру з передовими інформаційними технологіями, методами керування та сенсорними засобами моніторингу. Можна виокремити наступні ключові особливості ВІЕМ:

- висока динамічність внаслідок значної частки ВДЕ з непередбачуваною генерацією [17] ;
- необхідність роботи в реальному часі з урахуванням змін попиту та пропозиції [18];
- підвищені вимоги до надійності та резилієнтності системи в умовах аварій, збоїв чи кібератак [19].

Традиційним системам керування, заснованим переважно на централізованих підходах, притаманні ряд обмежень [20]. Зокрема, до завдань, які ще очікують свого вирішення належать: значні часові витрати на виконання аналітичних процедур; складність масштабування релейного захисту; обмеженість механізмів перемикання у мережі; жорстка прив'язка до конкретних протоколів комунікації; а також недостатня гнучкість у керуванні технологічними процесами на рівні електростанцій.

Використання мультиагентних систем у контексті ВІЕМ надає змогу долати зазначені обмеження [1], [8], [11]. БАС забезпечують децентралізацію

процесів прийняття рішень, зменшення часу реакції системи, підвищення стійкості до збоїв комунікацій, а також можливість інтеграції різномірних компонентів у єдине інтелектуальне середовище. Експлуатаційні демонстрації БАС підтверджують практичну життєздатність децентралізованої координації як альтернативи суто централізованим підходам, забезпечуючи оперативне балансування та керування в режимі, наближеному до реального часу [21], [22].

1.1.1. Основні принципи агентного підходу

Агентний підхід виник як методологія розробки та дослідження складних систем, що характеризуються динамічністю, розподіленістю та наявністю численних гетерогенних компонентів. Центральним поняттям цієї парадигми є агент, що визначається як інтелектуальна сутність, здатна сприймати стан середовища, приймати рішення та здійснювати дії відповідно до поставлених цілей [23].

Ключовим принципом є автономність, тобто здатність агента функціонувати без постійного зовнішнього контролю. Автономні агенти приймають рішення локально, що дозволяє системі уникати “вузьких місць” (англ.: *bottleneck*), характерних для централізованих архітектур, і зменшує залежність від єдиного керувального центру.

Другий принцип — соціальність, яка полягає у здатності агентів до взаємодії між собою та з оточенням. За допомогою стандартизованих протоколів обміну даними агенти координують дії, формують коаліції та досягають колективних цілей. Саме взаємодія перетворює набір ізольованих компонентів на узгоджену мультиагентну систему.

Важливою властивістю є реактивність: агенти здатні швидко реагувати на зміни в середовищі, що особливо критично для систем реального часу, зокрема в енергетичних мережах. У поєднанні з реактивністю реалізується проактивність — здатність агентів до цілеспрямованої поведінки, прогнозування майбутніх станів системи та ініціювання власних дій. Це

дозволяє не лише відповідати на зовнішні події, а й формувати довгострокові стратегії.

Ще одним принципом є адаптивність, що забезпечує зміну поведінки агентів залежно від попереднього досвіду, інформації від інших агентів чи умов середовища. Адаптивність є передумовою стійкості системи до стохастичних впливів та непередбачуваних подій.

Гетерогенність агентів полягає у можливості співіснування різномірних компонентів — програмних, апаратних та кіберфізичних. Така властивість дає змогу інтегрувати у спільне середовище як традиційні енергетичні ресурси, так і нові технології, зокрема ВДЕ чи розподілені накопичувачі.

Фундаментальним принципом мультиагентних систем є децентралізація: немає єдиного центру управління, натомість прийняття рішень здійснюється на локальному рівні кожним агентом. Це сприяє масштабованості системи, її стійкості до відмов і збоїв, а також підвищує ефективність при розширенні мережі.

Завершальним принципом є цілеспрямованість, а саме, поведінка агентів визначається як локальними завданнями, так і глобальними цілями системи. Баланс між індивідуальною раціональністю агента та колективною узгодженістю забезпечує досягнення системного ефекту, який перевищує можливості окремих компонентів.

Сукупність перелічених принципів становить основу агентного підходу. Вони визначають мультиагентні системи як ефективний інструмент для моделювання та керування складними динамічними об'єктами. Застосування цих принципів у галузі енергетики створює передумови для побудови гнучких і стійких ВІЕМ, здатних до самовідновлення, інтеграції розподілених ресурсів та роботи в умовах невизначеності [23], [24].

1.1.2. Агентне моделювання у контексті високоінтелектуальних енергетичних мереж

Розвиток енергетики на сучасному етапі характеризується переходом від централізованих систем електропостачання до розподілених архітектур, де важливу роль відіграють ВДЕ, розподілені накопичувачі, електротранспорт та інтелектуальні системи керування [11]. В умовах зростання динамічності, стохастичності генерації та багатоваріантності сценаріїв споживання традиційні централізовані методи управління виявляють низку обмежень: значні часові затримки, обмежену масштабованість, недостатню стійкість до відмов та кібератак [20].

Агентне моделювання у цьому контексті розглядається як підхід до дослідження і побудови ВІЕМ. Воно базується на поданні системи у вигляді множини взаємодіючих агентів, кожен з яких моделює певний компонент або групу компонентів енергомережі: виробники енергії, споживачі, накопичувачі, оператори систем розподілу чи ринкові агенти [24].

Модель дозволяє описати поведінку кожного агента через локальні правила прийняття рішень та механізми взаємодії, що відображає реальну складність і різноманітність сучасних енергосистем. У такий спосіб досягається відповідність між архітектурою ВІЕМ та агентним підходом: кожен учасник енергетичної системи функціонує як автономний агент, а глобальна динаміка формується у результаті їхньої кооперації [1].

До основних переваг агентного моделювання у контексті ВІЕМ слід віднести [1], [23]:

- розподіленість управління, що усуває “вузькі місця” централізованих схем;
- масштабованість, яка дозволяє інтегрувати нові ресурси без радикальної зміни архітектури;
- адаптивність і гнучкість, що забезпечують роботу в умовах невизначеності попиту та стохастичності генерації;

- стійкість і резилієнтність, досягнуті завдяки можливості швидкої локальної реакції агентів на збої чи аварії;
- можливість моделювання ринкових і технічних механізмів одночасно, що важливо для аналізу економіко-технічних аспектів функціонування енергосистем.

Таким чином, агентне моделювання виступає інструментом не лише для наукового аналізу, а й для практичного проектування механізмів керування ВІЕМ, що поєднують технічні, економічні та інформаційні компоненти. Воно дозволяє досліджувати ефективність стратегій балансування попиту й пропозиції, оцінювати стійкість до збурень, аналізувати сценарії впровадження нових технологій та політик [1].

1.1.3. Типи агентів та принципи їх внутрішньої організації

Внутрішня організація агента визначає його поведінку, тобто дії як функцію від сприйняття середовища та змін у ньому [25]. Залежно від рівня автономності, складності моделей сприйняття та механізмів прийняття рішень виділяють кілька типів агентів, кожен з яких має власні архітектурні особливості та сфери застосування [26]:

- Рефлексивні агенти (англ.: Reflexive agents). Функціонують на основі простих правил типу “умова → дія”. Не мають внутрішнього уявлення про середовище, але забезпечують швидку реакцію. Застосовуються у випадках, коли критичним є час відгуку, наприклад, у системах релейного захисту.
- Агенти, орієнтовані на цілі (англ.: Goal-based agents). Мають внутрішню модель середовища та здатність зберігати історію сприйняття. Вони спрямовують свої дії на досягнення заданих цілей і можуть реалізовувати складніші стратегії поведінки. Прикладом застосування є управління мікрогрідом чи оптимізація графіків навантаження.

- Агенти, орієнтовані на корисність (англ.: Utility-based agents). Використовують функцію корисності для оцінювання можливих дій і вибору оптимальної. Вони забезпечують раціональну поведінку з урахуванням кількох критеріїв ефективності, таких як мінімізація витрат чи підвищення надійності. У сфері ВІЕМ подібні агенти застосовують для динамічного ціноутворення та управління розподіленими енергоресурсами.
- Агенти, що навчаються (англ.: Learning agents). Поєднують властивості попередніх типів, але додатково здатні змінювати власну поведінку на основі досвіду. Використовують алгоритми навчання з учителем, без учителя або з підкріпленням. Типовим прикладом є агенти для прогнозування попиту на електроенергію чи оптимізації процесів заряджання електромобілів.

Таким чином, різні типи агентів відрізняються рівнем складності внутрішньої архітектури й функціональними можливостями. Вибір конкретного типу визначається завданням, яке необхідно вирішити в рамках енергетичної системи: від швидкого реагування у разі відмов до довгострокового прогнозування та оптимізації.

1.1.4. Архітектурні підходи до побудови мультиагентних систем

Архітектура мультиагентної системи визначає спосіб організації агентів, розподіл функцій між ними та механізми координації їхньої діяльності. Вибір архітектури безпосередньо впливає на такі властивості системи, як масштабованість, стійкість до відмов, гнучкість та ефективність взаємодії [23]. У контексті енергетичних мереж, де спостерігається висока динамічність і неоднорідність компонентів, архітектурні рішення суттєво впливають на ефективність системи.

Виокремлюють декілька базових архітектурних підходів, що застосовуються для моделювання та управління ВІЕМ [1], [27], [28]:

1. Централізована архітектура (англ.: centralized). Усі агенти підпорядковуються єдиному центральному контролеру, який акумулює інформацію з елементів мережі, виконує обчислення та формує управлінські рішення. Такий підхід забезпечує глобальну узгодженість і спрощує координацію, однак має низку суттєвих недоліків: високу обчислювальну складність, ризик виникнення єдиної точки відмови та обмежену масштабованість при зростанні кількості учасників.
2. Децентралізована архітектура (англ.: decentralized). У цій моделі кожен агент володіє достатнім рівнем автономії для прийняття рішень, базуючись на локальній інформації та обміні даними з іншими агентами. Такий підхід дозволяє досягти високої масштабованості та підвищує стійкість до відмов окремих вузлів, проте потребує складних механізмів узгодження рішень і може призводити до конфліктів між агентами.
3. Ієрархічна архітектура (англ.: hierarchical). Агенти організовані за рівнями управління: від локальних (на рівні мікрогрід чи окремих підстанцій) до регіональних та глобальних координаторів. Ієрархічний підхід поєднує переваги централізації та децентралізації, забезпечуючи локальну автономність разом із можливістю глобальної координації. Це особливо корисно для багаторівневих енергетичних систем.
4. Голонічна архітектура (англ.: holon). Базується на концепції «голона», тобто об'єкта, який є одночасно автономною цілісною одиницею та частиною більшої структури. У контексті ВІЕМ це означає, що кожна мікрогрід або підсистема може виступати як незалежний агент, але водночас залишатися інтегрованою у більшу енергомережу. Голонічні системи ефективні для моделювання гнучких ієрархій керування та координації в умовах змінної топології.
5. Коаліційна архітектура (англ.: coalition). У цій моделі агенти мають можливість утворювати коаліції для розв'язання конкретних завдань — наприклад, управління балансом навантаження, реагування на аварійні ситуації чи спільну оптимізацію ресурсів. Коаліції можуть формуватися

динамічно залежно від поточних умов, що забезпечує високу адаптивність і гнучкість системи.

Вибір архітектурного стилю визначає ефективність мультиагентного моделювання у сфері енергетики. Кожна з наведених архітектур має власні переваги та обмеження, а на практиці часто застосовуються гібридні рішення, що поєднують різні принципи організації з метою досягнення оптимального балансу між надійністю, швидкодією та масштабованістю.

1.1.5. Опис станів агентів у мультиагентних системах

У мультиагентних системах ключовим аспектом є опис та підтримка актуальних станів агентів і системи загалом. Під станом агента розуміють сукупність внутрішніх і зовнішніх параметрів, які визначають його поведінку та можливості взаємодії з іншими агентами й середовищем [24], [25]. Опис стану включає інформацію про навколишнє середовище, цілі, історію взаємодій, поточні ресурси та обмеження [29].

Оскільки агенти у ВІЕМ можуть бути як фізичними (генератори, накопичувачі, споживачі), так і віртуальними сутностями (системи прогнозування, оптимізації чи управління), потрібна уніфікована та розширювана модель стану, здатна узгоджено описувати їхні атрибути та підтримувати синхронізацію в розподіленому середовищі [26], [30]. Для скоординованої роботи агентів необхідне уніфіковане представлення ключових параметрів, серед яких виділяють:

- Енергетичний стан – рівень виробництва, споживання, збереження та передавання енергії;
- Комунікаційний стан – доступність каналів зв'язку, затримки, втрати пакетів, рівень мережевої надійності;
- Операційний стан – режим роботи обладнання, наявність чи відсутність відмов, технічні обмеження;

- Інформаційний стан – накопичені знання, результати прогнозування та обчислень, локальні й глобальні правила прийняття рішень;
- Соціальний стан – відносини агента з іншими агентами, належність до груп, ступінь довіри.

Важливою вимогою є узгодженість станів, під якою розуміємо відсутність конфліктів між реплікованими поданнями даних на різних вузлах згідно з обраною моделлю узгодженості. У багатьох випадках агенти працюють асинхронно, а інформація надходить із затримками чи в умовах часткових відмов. Це може призводити до розбіжностей у баченні системи різними агентами.

Усунення таких розбіжностей досягається шляхом використання механізмів реплікації даних та алгоритмів консенсусу, які забезпечують узгоджене відображення станів у розподіленій системі навіть за відсутності повністю централізованого контролю [29], [31].

Відповідно, опис станів у мультиагентних системах виступає не лише технічним інструментом збереження даних, а й основою для організації взаємодії між агентами, їхньої автономії та колективного прийняття рішень [24].

1.2. Методи забезпечення узгодженості станів у розподілених системах

Опис станів агентів і системи загалом є визначальною умовою їхньої взаємодії та колективного прийняття рішень. Проте сам факт наявності структурованого опису стану ще не гарантує його коректності в умовах розподіленого середовища. Агенти можуть отримувати інформацію з різними затримками, у різні моменти часу, а також стикатися з неповними чи суперечливими даними. У таких умовах постає завдання забезпечення узгодженості (англ.: consistency) між локальними уявленнями про стан системи.

Забезпечення узгодженості станів є одним з ключових викликів у розподілених обчисленнях. На відміну від централізованих систем, де існує єдине джерело істини, у розподілених середовищах необхідно досягати певного

рівня погодженості даних між численними вузлами без постійної централізованої координації. Вибір методів і моделей узгодженості залежить від цілей системи, вимог до продуктивності, відмовостійкості та часу відгуку. Для ВІЕМ цей аспект є критичним, адже неконсистентність даних може призводити до критичних наслідків - від неефективного використання ресурсів до аварій у мережі [32].

1.2.1. Поняття узгодженості в розподілених обчисленнях

Узгодженість у розподілених обчисленнях визначає ступінь відповідності локальних копій даних, що зберігаються на різних вузлах системи. У розподіленому середовищі дані реплікуються між багатьма вузлами, що працюють асинхронно, можуть відмовляти або мати різну затримку у зв'язку. Це створює ризик розбіжностей у баченні стану системи [33].

Класичне визначення узгодженості у рамках моделі CAP (англ.: Consistency, Availability, Partition tolerance) передбачає, що при наявності розділення мережі система повинна обирати між узгодженістю та доступністю. Тобто абсолютна узгодженість (англ.: strong consistency) у глобально розподілених середовищах часто недосяжна без втрат продуктивності чи доступності [34].

Залежно від гарантій, які надаються щодо узгодженості, у сучасній літературі виділяють кілька основних моделей [35]:

- Сильна узгодженість (англ.: strong consistency) гарантує, що всі клієнти бачать однакові дані після завершення оновлення, тобто система поводить себе так, ніби існує єдина глобальна копія стану.
- Причинна узгодженість (англ.: causal consistency) враховує причинно-наслідкові залежності між операціями; усі вузли погоджуються щодо порядку подій, які пов'язані причинно, але не обов'язково щодо незалежних подій.

- Остаточна узгодженість (англ.: eventual consistency) передбачає, що у відсутності нових оновлень усі копії даних зрештою збіжаться, хоча на проміжних етапах можливі розбіжності.
- Слабка узгодженість (англ.: weak consistency) надає мінімальні гарантії: система може тимчасово повертати застарілі або неповні дані, якщо це виправдано вимогами до продуктивності чи масштабованості.

Такий поділ дозволяє системним архітекторам обирати компроміс між строгістю гарантій, продуктивністю й відмовостійкістю залежно від конкретного застосування. Іншими словами, вибір моделі узгодженості не є лише технічним аспектом, а стратегічним рішенням, що визначає баланс між безпекою даних і ефективністю роботи системи. Наприклад, сильна узгодженість забезпечує найвищий рівень надійності, але часто знижує масштабованість та призводить до затримок у виконанні операцій. Навпаки, остаточна чи слабка узгодженість дозволяє підвищити пропускну здатність і забезпечити роботу в умовах високої динаміки, проте може тимчасово створювати розбіжності у станах вузлів.

1.2.2. Традиційні підходи до підтримання узгодженості

Традиційні підходи до забезпечення узгодженості в розподілених системах зазвичай базуються на використанні централізованих або децентралізованих механізмів, які забезпечують, що всі вузли системи бачать однакові дані в певний момент часу. Ці підходи мають на меті запобігти конфліктам, що можуть виникнути в результаті паралельних запитів на зміну даних, і гарантувати, що система функціонує як єдине ціле.

Такі підходи до забезпечення узгодженості станів у розподілених системах базуються на принципах сильної узгодженості, що вимагає, щоб усі вузли системи в кожен момент часу бачили однаковий стан даних. Ці методи є історичною основою для розробки надійних розподілених баз даних та файлових систем. Однією з найпростіших, але ефективних стратегій є

централізоване блокування (англ.: centralized locking), де єдиний, виділений сервер керує доступом до спільних ресурсів. Коли вузол-ініціатор хоче виконати операцію запису, він надсилає запит на блокування і лише після отримання дозволу може продовжити. Це ефективно запобігає конфліктам, але створює значне “вузьке місце”, що обмежує пропускну здатність системи, а також робить її вразливою до єдиної точки відмови (англ.: single point of failure) [36], [37].

На противагу цьому, децентралізовані протоколи, такі як двофазовий фіксаційний протокол (англ.: Two-Phase Commit, далі - 2PC), розподіляють відповідальність за атомарність транзакцій між усіма учасниками. У першій фазі (назва англ.: “prepare”), координатор запитує кожного учасника, чи може він успішно виконати транзакцію. Якщо всі учасники відповідають "vote commit", у другій фазі (назва англ.: commit) координатор надсилає команду фіксації. Якщо хоча б один вузол відповідає "vote abort", вся транзакція відкочується. Ключове обмеження 2PC полягає в тому, що якщо координатор вийде з ладу після першої фази, учасники можуть застрягти в невизначеному стані очікування, що називається блокуванням (англ.: blocking). Щоб усунути цей недолік, був розроблений трифазовий фіксаційний протокол (далі - 3PC), який додає проміжну фазу "pre-commit", що дозволяє учасникам виявити відмову координатора і прийняти рішення самостійно, зменшуючи ймовірність блокування [38].

Крім протоколів, що базуються на блокуванні та фіксації, існують інші методи, що забезпечують узгодженість, зокрема:

- Протоколи на основі часових міток (англ.: Timestamp-based protocols). Цей підхід використовує глобальні часові мітки для впорядкування транзакцій. Кожна транзакція отримує унікальну часову мітку, і всі операції виконуються в порядку зростання цих міток. Якщо система виявляє, що транзакція намагається отримати доступ до даних, які були змінені більш новішою транзакцією, що вже зафіксувалася, то поточна транзакція відкочується. Цей метод дозволяє уникнути взаємних

блокувань, але може призводити до частих відкочувань транзакцій, що знижує загальну ефективність [39].

- Оптимістичний контроль конкурентності (англ.: Optimistic Concurrency Control, далі - OCC). На відміну від песимістичних підходів, які запобігають конфліктам, OCC припускає, що вони трапляються рідко. Кожна транзакція виконується без блокувань, а перед фіксацією проходить фазу перевірки (англ.: validation). На цій фазі система перевіряє, чи не були її дані змінені іншими транзакціями. Якщо конфліктів немає, транзакція фіксується. В іншому випадку вона відкочується і може бути перезапущена. Цей підхід забезпечує високий рівень паралелізму, але є менш ефективним у системах з високою інтенсивністю конфліктів [40].

Ці традиційні методи, попри їх надійність, несуть високі накладні витрати на обмін повідомленнями та синхронізацію, що робить їх неоптимальними для сучасних, високомасштабованих систем, де доступність та стійкість до розділення мережі мають пріоритет.

1.2.3. Розподілені алгоритми узгодження

Обмеження традиційних методів забезпечення узгодженості, зокрема централізованих координаторів і блокуючих протоколів на кшталт 2PC, призвели до розвитку розподілених алгоритмів узгодження (англ.: distributed consensus algorithms). Їхня мета — досягти єдиного погодженого рішення між множиною вузлів у системі, що може функціонувати в умовах часткових відмов, асинхронності та можливих розділень мережі.

Один з найвідоміших і найвпливовіших алгоритмів консенсусу — Paxos, запропонований [Lamport, 1998] [41]. Його ключовою ідеєю є забезпечення узгодженості між множиною вузлів у системі, яка може працювати в умовах асинхронності, часткових збоїв і непередбачуваних затримок. Алгоритм формулюється через взаємодію трьох основних ролей: пропонентів, акцепторів

і виконавців. Пропоненти ініціюють процес, формуючи пропозиції щодо значення, яке має бути прийняте системою. Акцептори виступають арбітрами цього процесу, голосуючи за запропоновані значення. Рішення вважається прийнятим лише тоді, коли його підтримує більшість акцепторів, тобто досягнуто кворуму. Виконавці, у свою чергу, отримують остаточне погоджене значення та поширюють його серед решти вузлів системи.

Гарантії, які надає Paxos, охоплюють два фундаментальні аспекти. По-перше, алгоритм забезпечує безпомилковість (англ.: safety), тобто неможливість того, що два різні вузли приймуть різні рішення. По-друге, за умови відсутності тривалих збоїв у комунікаціях та роботи достатньої кількості вузлів, Paxos гарантує живучість (англ.: liveness), тобто прогрес системи неминучий і погоджене значення буде визначене. Важливим механізмом, що лежить в основі Paxos, є використання унікальних номерів пропозицій. Це дає змогу впорядковувати паралельні ініціативи та уникати суперечностей між конкурентними пропозиціями. Попри високий рівень формальної обґрунтованості, Paxos здобув репутацію важкого для практичної імплементації. Це зумовлено його багатоступеневою процедурою, необхідністю точного дотримання ролей учасників і складністю координації повідомлень у розподіленому середовищі. Саме ця складність імплементації стала причиною пошуку більш практичних альтернатив.

Raft було запропоновано як більш зрозумілий і практично орієнтований алгоритм консенсусу, що зберігає ті самі гарантії безпеки та живучості, які забезпечує Paxos [42]. Основна ідея Raft полягає у спрощенні механізмів узгодження завдяки чіткій структуризації процесу реплікації логів та впровадженню ролі лідера, що координує діяльність решти вузлів.

Алгоритм працює у рамках трьох ролей: лідера, кандидатів і послідовників. Лідер обирається на підставі голосування серед усіх вузлів, після чого він бере на себе відповідальність за прийом клієнтських запитів і їх упорядкування у журналі операцій. Послідовники пасивно приймають записи, надіслані лідером, а кандидати з'являються під час виборів у випадку, коли

система не має активного лідера. Цей механізм забезпечує єдину точку координації та значно знижує складність узгодження, порівняно з класичним Paxos.

Ключовим компонентом Raft є механізм реплікації логів. Лідер отримує команди від клієнтів, додає їх у свій журнал і поширює серед послідовників. Після того, як більшість вузлів підтвердить отримання запису, команда вважається закоміченою і може бути застосована до стану системи. Таким чином забезпечується як безпомилковість (усі вузли застосовують однакові команди в однаковому порядку), так і живучість (система продовжує роботу, доки кворум вузлів залишається доступним).

На відміну від Paxos, який відзначається складністю реалізації та розуміння, Raft було спроектовано з акцентом на зрозумілість. Чітке розмежування механізмів — вибори лідера, реплікація журналу, гарантії безпечності — спрощує впровадження протоколу. Саме ця властивість зумовила його широке поширення у промислових системах, де простота обслуговування та перевірюваність мають критичне значення. Raft сьогодні застосовується як базовий механізм узгодження у численних розподілених інфраструктурних рішеннях, включно з системами керування конфігураціями та кластеризацією.

ZooKeeper Atomic Broadcast (англ.: Zookeeper Atomic Broadcast, далі - ZAB) є спеціалізованим протоколом, розробленим для забезпечення узгодженої реплікації у сервісі Apache ZooKeeper. Його основна мета полягає у реалізації атомарного широкомовлення, що гарантує доставку всіх повідомлень у єдиному порядку для всіх учасників або ж їх повне відхилення. Завдяки цьому репліки кластера підтримують узгоджений стан навіть у разі відмови окремих вузлів.

Архітектура ZAB базується на механізмі лідера, який приймає оновлення та поширює їх серед послідовників. Зміни вважаються затвердженими лише після підтвердження більшістю учасників, що забезпечує безпечність і відмовостійкість. У випадку втрати лідера протокол підтримує гарантований перехід керування, не порушуючи узгодженості системи. Завдяки простоті та ефективності ZAB добре підходить для сценаріїв із високою інтенсивністю

читання та меншою кількістю записів, характерних для ZooKeeper. Це робить його прикладом спеціалізованого протоколу, оптимізованого для конкретних потреб координаційних сервісів у великих розподілених інфраструктурах.

У середовищах, де можливі зловмисні або довільні відмови вузлів, застосовуються алгоритми типу Practical Byzantine Fault Tolerance (далі - PBFT) [43]. На відміну від Paxos чи Raft, які передбачають лише аварійні відмови (англ.: crash failures), PBFT здатний забезпечувати узгодженість навіть тоді, коли деякі вузли поведуться довільно або навмисно некоректно. Це досягається завдяки використанню багатоетапного протоколу повідомлень між вузлами, що дозволяє системі функціонувати коректно за умови, що не більше третини учасників є зловмисними. PBFT став основою для низки критичних систем, де потрібна підвищена надійність і безпека, зокрема у фінансових додатках та блокчейн-платформах.

HotStuff є сучасною еволюцією протоколів BFT, що поєднує властивості масштабованості та спрощеної реалізації [44]. Його архітектура базується на використанні послідовних раундів голосування, які зменшують кількість комунікаційних кроків, необхідних для досягнення консенсусу. Завдяки цьому HotStuff досягає лінійної комунікаційної складності й зберігає властивості безпечності та живучості навіть у великих мережах.

1.2.4. Моделі остаточної узгодженості

Остаточна узгодженість (англ. eventual consistency) — є однією з найпоширеніших слабких моделей консистентності, що застосовуються у масштабованих розподілених системах, особливо в середовищах, де першочерговими вимогами є висока доступність та стійкість до відмов. Основна ідея цієї моделі полягає в тому, що за відсутності нових оновлень усі репліки системи зрештою збігаються до одного узгодженого стану, навіть якщо протягом певного часу клієнти можуть спостерігати застарілі або різні значення даних [45].

Згідно з формулюванням Vogels [45], *«остаточна узгодженість є гарантією живучості: якщо нові оновлення не відбуваються, то зрештою всі звернення повертатимуть останнє оновлене значення»*. Водночас ця модель не забезпечує властивості безпеки, оскільки проміжні стани не регламентуються. Як зазначає Bailis [46]: *«за цим визначенням, остаточна узгодженість є виключно гарантією живучості — щось добре зрештою станеться — але вона не забезпечує жодних властивостей безпеки»*.

Основні характеристики остаточної узгодженості:

- Висока доступність (англ.: high availability): дозволяє обробляти операції під час мережових розділень або відмов, надаючи доступ навіть під час асинхронної реплікації.
- Живучість без гарантій безпеки (англ.: a liveness property, not a safety property): гарантується зрештою збіжність, але не безпека — проміжні значення можуть бути неконсистентними.
- Збіжність (англ.: convergence) — у відсутності нових оновлень усі копії даних зрештою приходять до однакового значення.
- Відсутність негайної консистентності — читання в проміжку можуть повертати застарілі або суперечливі дані.

Щоб підвищити передбачуваність поведінки з точки зору користувачів, у межах остаточної узгодженості розроблено низку клієнт-орієнтованих гарантій (англ.: session guarantees) [47]:

- Читання власних записів (англ.: read-your-writes) — клієнт завжди бачить власні оновлення;
- Монотонні читання (англ.: monotonic reads) — клієнт не отримає старішу версію після того, як вже прочитав новішу;
- Монотонні записи (англ.: monotonic writes) — записи одного клієнта відбуваються у правильному порядку;

- Записи після читань (англ.: writes-follow-reads) — операції запису відображають спостережувані раніше читання.

Більш строгим варіантом є причинно-наслідкова узгодженість (англ.: causal consistency), яка гарантує, що всі причинно-пов'язані операції спостерігаються у правильному порядку [48]. Подальші модифікації включають Causal+ Consistency, що додає гарантію збіжності через механізми розв'язання конфліктів, та Tunable Causal Consistency (TCC), яка дозволяє клієнтам обирати рівень гарантій залежно від вимог до продуктивності [49].

Сучасні дослідження також розглядають питання забезпечення консистентності у хмарних та P2P-системах. Наприклад, Campêlo et al. [50] систематизують підходи до реплікаційної консистентності у хмарних середовищах, тоді як Kleppmann і Howard [51] пропонують модель Byzantine Eventual Consistency для захисту від ворожих або зловмисних вузлів.

Таким чином, моделі остаточної узгодженості задають баланс між масштабованістю, доступністю та узгодженістю, поступаючись строгістю гарантій заради продуктивності та відмовостійкості.

1.2.5. Обмеження існуючих підходів у контексті високоінтелектуальних енергетичних мереж.

Попри значні досягнення у розробці алгоритмів забезпечення узгодженості в розподілених системах, їх пряме застосування у ВІЕМ стикається з низкою суттєвих обмежень. Специфічні для предметної області чинники, такі як стохастичність відновлюваної генерації, динамічність попиту, гетерогенність учасників та вимоги до роботи у реальному часі, часто охоплюються лише частково у класичних моделях, що зумовлює потребу адаптації моделей та механізмів узгодженості [1], [52].

По-перше, затримки та ненадійність комунікацій у широкомасштабних мережах ускладнюють досягнення сильної консистентності, що потребує глобальної синхронізації. Mashal та ін. [53] підкреслюють, що ефективність

ВІЕМ значною мірою залежить від надійності та оперативності інформаційних потоків, а також від сумісності та конфіденційності даних.

По-друге, гетерогенність учасників — від виробників, накопичувачів та мікромереж до торгових платформ і споживачів — створює середовище, в якому універсальні підходи до узгодженості (наприклад, Paxos чи Raft) демонструють суттєві обмеження. Зокрема, Paxos хоч і забезпечує високу відмовостійкість, є складним у реалізації, тоді як Raft, хоч і простіший, виявляється менш придатним для великих децентралізованих мереж, включно з блокчейном, де представники агенти суттєво різняться за ресурсами та ролями [54].

По-третє, обмеженням залишається масштабованість та резилієнтність. Традиційні протоколи консенсусу розроблялися для обмежених середовищ і характеризуються високими комунікаційними витратами при збільшенні кількості вузлів. Наприклад, Paxos, вимагають інтенсивного обміну повідомленнями між усіма або більшістю вузлів для досягнення згоди. У міру збільшення кількості вузлів (N), кількість необхідних повідомлень зростає експоненційно або поліноміально ($O(N^2)$ або вище), що обмежує масштабованість протоколу. Для ВІЕМ, де кількість агентів може сягати мільйонів, накладні витрати на комунікацію роблять традиційні протоколи консенсусу, зокрема Paxos та Raft, практично непридатними для великомасштабних розгортань [55], [56]. Щодо резилієнтності (англ.: resilience) слід зазначити, що хоча традиційні протоколи консенсусу і забезпечують високу стійкість до відмов збоєм (англ.: crash failures), вони не завжди ефективно справляються з візантійськими відмовами (англ.: Byzantine failures), які є критичними для безпеки в енергетичних мережах [57]. Крім того, ці протоколи розроблялися для відносно стабільних мереж, тоді як топологія ВІЕМ є надзвичайно динамічною, що вимагає постійного переформатування груп вузлів, що є дорогим і повільним процесом.

Таким чином, розглянуті підходи до забезпечення узгодженості, що добре зарекомендували себе у класичних розподілених системах, виявляють суттєві

обмеження у контексті ВІЕМ. Це обумовлює необхідність пошуку нових рішень, здатних поєднати масштабованість, стійкість до відмов та інтегровану безпеку.

1.3. Неконфліктні репліковані структури даних у моделюванні енергетичних процесів

Потреба в оперативному, надійному та безпечному обміні даними між тисячами, а подекуди й мільйонами гетерогенних пристроїв, вимагає переосмислення традиційних підходів до узгодженості даних. Класичні протоколи консенсусу, що базуються на сильному, глобальному узгодженні, часто виявляються неефективними через високі комунікаційні витрати, затримки та низьку відмовостійкість у динамічних мережах.

Центральним аспектом є активна синхронізація станів [58], [59] у розподіленій мережі для забезпечення її структурної та функціональної узгодженості. Така синхронізація є вирішальною для критичної інфраструктури та є однією з необхідних умов для її безперебійного функціонування та стабільності [60], [61]. CRDT [62] можуть забезпечувати узгодженість детерміновано та без конфліктів. Вони пропонують принципово іншу модель для роботи з розподіленими даними, де узгодженість досягається без потреби у центральному координаторі або глобальному консенсусі. CRDT використовують математично обґрунтовані підходи, що гарантують сильну остаточну узгодженість (англ.: strong eventual consistency): навіть якщо різні вузли працюють автономно та отримують оновлення в різному порядку, їх кінцевий стан завжди буде ідентичним після злиття. Така властивість дозволяє створити високоефективні системи, які стійкі до відмов, швидко масштабуються і оптимально використовують обмежені ресурси пристроїв.

1.3.1. Поняття неконфліктних реплікованих структур даних

Неконфліктні репліковані структури даних - це клас математично визначених абстракцій даних, призначених для забезпечення сильної

узгодженості в кінцевому підсумку. Їхня ключова особливість полягає в тому, що вони дозволяють незалежне та одночасне оновлення на різних репліках без необхідності координації. Конфлікти, які неминуче виникають у таких сценаріях, вирішуються автоматично, гарантуючи, що всі репліки зрештою зійдуться до одного і того ж стану, незалежно від порядку застосування операцій [63].

Концепція CRDT була вперше систематизована Марком Шапіро та співавторами у 2011 році [63], які показали, що завдяки властивостям комутативності, асоціативності та ідемпотентності можна побудувати структури даних, що не потребують централізованого контролю, але залишаються коректними.

Подальші дослідження [64] розширили початкову концепцію CRDT, систематизувавши її для різних рівнів розробки — від застосунків до системної інженерії та конструювання структур даних. У цій роботі CRDT були представлені як інструмент, що здатний гарантувати детерміновану збіжність реплік у середовищах із високим рівнем конкуренції оновлень, асинхронною доставкою та масштабною геореплікацією. Особливий акцент зроблено на промисловій практиці, де CRDT застосовуються у великих розподілених базах даних і колективних редакторах.

CAP-теорема [34] формально довела, що в умовах розподіленої системи неможливо одночасно гарантувати тріаду властивостей: узгодженість (англ.: consistency), доступність (англ.: availability) та стійкість до розділення мережі (англ.: partition tolerance). У реальних умовах, де розділення мережі є неминучими, системи мають обирати компроміс між узгодженістю та доступністю. CRDT пропонують формалізоване рішення цієї дилеми: вони відмовляються від сильної узгодженості на користь остаточної, що дозволяє підтримувати високу доступність і масштабованість без критичної залежності від центрального координатора. Таким чином, CRDT можна розглядати як практичну реалізацію підходу AP-систем у термінах CAP-теореми, де конфлікти

усуваються за рахунок математичних властивостей операцій, а не глобального контролю.

1.3.2. Теоретичні основи неконфліктних реплікованих структур даних

Основою коректності CRDT є низка алгебраїчних властивостей операцій, які гарантують що операції можуть застосовуватися в будь-якому порядку, і кінцевий результат буде однаковим на всіх репліках [63]:

- Ідемпотентність (англ.: idempotence): Ця властивість означає, що повторне застосування однієї й тієї ж операції не змінює результат. Якщо операція x застосовується до стану S , то $\text{apply}(S, x)$ дорівнює $\text{apply}(\text{apply}(S, x), x)$. Це критично важливо для надійного обміну повідомленнями в мережі, де повідомлення можуть дублюватися.
- Комутативність (англ.: commutativity): Ця властивість гарантує, що порядок застосування двох різних операцій не має значення. Якщо є дві операції x і y , то $\text{apply}(\text{apply}(S, x), y)$ дорівнює $\text{apply}(\text{apply}(S, y), x)$. Це дозволяє реплікам незалежно обробляти оновлення, що надходять з різних джерел у різному порядку.
- Асоціативність (англ.: associativity): Ця властивість дозволяє агрегувати оновлення в довільному порядку, що є основою для масштабованої реплікації у великих системах. Якщо є три операції x , y і z , то $\text{apply}(S, x, (y, z))$ дорівнює $\text{apply}(S, (x, y), z)$.

Завдяки поєднанню цих властивостей CRDT забезпечують коректність та збіжність системи без глобального контролю.

У традиційних системах узгодженість досягається через централізоване управління — координатор вирішує, у якому порядку застосовувати оновлення. Натомість логіка CRDT дозволяє обійтися без централізованого координатора або механізмів блокування. Кожен вузол системи працює зі своєю локальною копією даних. Реплікація відбувається асинхронно, коли вузли обмінюються або повним станом, або ж тільки операціями. Кожна операція виконується миттєво

на локальній репліці, а потім розсилається іншим вузлам. Під час об'єднання оновлень використовується математично визначена функція злиття (англ.: merge function), яка гарантує конфліктівільність завдяки згаданим вище властивостям. Таким чином, система не потребує централізованого координатора й залишається функціональною навіть за умов розділення мережі (англ.: network partition) чи затримок синхронізації [64].

CRDT значно відрізняються від традиційних підходів до узгодження, таких як консенсусні протоколи, блокування та транзакції:

- Консенсус (напр., Paxos, Raft): Ці протоколи гарантують, що всі вузли погоджуються на одному порядку операцій (сильна узгодженість). Це забезпечує максимальну коректність, але є дорогим з точки зору комунікаційних витрат і нестійким до розділення мережі. Натомість, CRDT гарантують узгодженість у кінцевому підсумку, що робить їх набагато доступнішими і стійкими до розділення, але вони не можуть гарантувати, що всі репліки будуть ідентичними в кожен момент часу.
- Блокування (напр., 2PL): Протоколи блокування, такі як двофазне блокування, забезпечують узгодженість, запобігаючи одночасному доступу до даних. Це створює вузькі місця та може призводити до взаємних блокувань (англ.: deadlocks). CRDT повністю уникають блокувань, оскільки дозволяють одночасні зміни та вирішують конфлікти постфактум.
- Транзакції (напр., 2PC): Розподілені транзакції гарантують атомарність (ACID-властивості), але вимагають інтенсивної координації між вузлами, що робить їх повільними та вразливими до відмов координатора. CRDT, навпаки, не вимагають централізованого управління транзакціями, що робить їх ідеальними для систем, де критичними є висока доступність та стійкість до відмов мережі.

На противагу до цих підходів, CRDT належать до класу рішень, що реалізують оптимістичну реплакацію (англ.: optimistic replication): вони

дозволяють системам залишатися доступними у випадку часткових відмов і синхронізуватися поступово, досягаючи остаточної узгодженості без втрати даних [64]. Це дозволяє використання CRDT для масштабованих, децентралізованих і резилієнтних архітектур, зокрема у сфері ВІЕМ, де необхідна безперервна робота навіть у разі нестабільних комунікацій.

1.3.3. Класифікація неконфліктних реплікованих структур даних

CRDT поділяються на дві основні категорії, залежно від того, як вони забезпечують узгодженість: за допомогою обміну станом або обміну операціями [63].

Перша, CRDT на основі стану (англ.: State-based CRDT, далі - CvRDT), працює шляхом обміну повним станом даних між репліками. Кожна репліка, отримуючи стан від іншої, зливає його з власним, використовуючи функцію злиття, яка відповідає властивостям напівґратки (англ.: semilattice). Цей підхід є простим у реалізації, оскільки не вимагає гарантованого порядку доставки повідомлень.

Друга категорія, CRDT на основі операцій (англ.: Operation-based CRDT, далі - CmRDT) забезпечує узгодженість, обмінюючись лише самими операціями. Ці операції мають бути ідемпотентними та комутативними, що дозволяє застосовувати їх у будь-якому порядку. Цей підхід є більш ефективним з точки зору використання мережевих ресурсів, оскільки передаються лише невеликі повідомлення-операції, але він вимагає надійного механізму доставки, що гарантує, що кожна операція буде доставлена та оброблена.

Таким чином, обидва підходи пропонують сильні гарантії узгодженості в кінцевому підсумку, обходячи потребу у складних протоколах консенсусу чи блокування, але роблять це з різними компромісами щодо ефективності та складності. Порівняння між CvRDT та CmRDT наведені в таблиці 1.1.

Таблиця 1.1. Порівняння характеристик CvRDT та CmRDT

| Характеристика | CvRDT | CmRDT |
|-----------------------|---|---|
| Обмін даними | Обмін повним станом | Обмін операціями |
| Складність | Низька - реалізується через функцію злиття станів. | Висока - гарантована причинно-узгоджена доставка операцій. |
| Комунікаційні витрати | Вищі - потрібно передавати весь стан | Нижчі - передаються лише невеликі операції |
| Узгодженість | Ідемпотентність та асоціативність функцій злиття гарантують остаточну узгодженість. | Комутативність операцій забезпечує збереження узгодженості між репліками. |
| Втрата повідомлень | Стійкі до втрат - повторне поширення повного стану компенсує втрачені повідомлення. | Уразливі до втрат, якщо не використовуються надійні механізми доставки. |

На основі зазначених підходів були сформовані спеціалізовані типи даних, серед яких лічильники, реєстри, множини та графи. Водночас одним із обмежень CRDT є фіксованість їхньої структури, яка визначається на етапі проектування і не підлягає зміні в процесі експлуатації. У таблиці 1.2 наведено класифікацію CRDT із зазначенням ключових властивостей та прикладів застосування [63].

Таблиця 1.2. Класифікація CRDT

| Тип CRDT | Підтримувані операції | Ключові властивості | Приклади застосування у розподілених системах |
|-----------------------|--|--|---|
| Лічильники (Counters) | Інкремент, декремент (G-Counter, PN-Counter) | Асоціативність, комутативність, ідемпотентність | Підрахунок запитів, кількість унікальних подій, енергооблік |
| Множини (Sets) | Додавання, видалення елементів (G-Set, 2P-Set, OR-Set) | Конфліктівільне додавання/видалення; уникнення дублювання | Списки учасників у P2P, контроль підключених споживачів у BIEM |
| Регістри (Registers) | Запис значення, збереження останнього оновлення (LWW-Register) | Розв'язання конфліктів за мітками часу; останній запис має пріоритет | Синхронізація стану датчиків (температура, напруга), конфігурації пристроїв |
| Карти (Maps) | Операції над ключами та вкладеними CRDT | Підтримка складних структур; кожне значення є CRDT | Конфігураційні бази, розподілені каталоги, моделі станів вузлів BIEM |
| Графи (Graphs) | Додавання/видалення вузлів і ребер (Add-Wins Graph, Remove-Wins Graph) | Неконфліктне управління топологією; підтримка збіжності | Соціальні мережі, топологія енергосистеми, маршрутизація у P2P |

Наведена класифікація типів CRDT показує, що вони охоплюють широкий спектр структур — від простих лічильників до складних графів. Це надає можливість використовувати CRDT як інструмент для побудови розподілених систем, які потребують масштабованої синхронізації та відмовостійкості. У випадку BIEM, застосування різних типів CRDT надає можливість детерміновано моделювати та узгоджувати як локальні стани окремих вузлів, так і глобальну топологію мережі.

1.3.4. Дослідження практичного використання неконфліктних реплікованих структур даних

В дослідженнях [65] було представлено протокол VCuve-Sync, який підтримує стан розподіленого застосунку за допомогою CRDT, дозволяючи виконувати операції одночасно на різних вузлах і при цьому забезпечуючи детерміновану збіжність кінцевого стану. Розробники підкреслюють, що ключова перевага підходу полягає у зниженні накладних витрат на координацію: система демонструє відмінні показники затримки доставки повідомлень (англ.: latency) і високу ефективність використання пропускну здатності навіть у сценаріях із великою кількістю публікаторів та підписників, що особливо важливо для pub/sub-архітектур та систем інтернету речей.

Подальші дослідження цього напрямку були розвинуті у [66], які показали, що CRDT-орієнтована синхронізація досягає надійної збіжності навіть у висококонкурентних середовищах, зокрема в багатопоточних і мультиагентних системах, де традиційні механізми блокувань або координації втрачають ефективність

У цьому ж контексті в дослідженні [67] було запропоновано концепцію розподіленої системи, у якій стан машини репрезентується CRDT-структурами, що дало змогу реалізувати консенсус у режимі реального часу. Ця робота показала, що представлення стану через CRDT не лише спрощує механізми узгодження, але й дозволяє уникнути використання “важких” протоколів

консенсусу на кшталт Paxos чи Raft, залишаючи при цьому властивість узгодженої синхронізації між вузлами.

У свою чергу, в [68] зосередилися на дослідженні різних протоколів синхронізації станів у розподілених системах та запропонували блокчейн-систему CChain, яка вперше інтегрувала методи CRDT-орієнтованої остаточної узгодженості у платформу Hyperledger Fabric. Це рішення продемонструвало, що навіть у таких критично чутливих до узгодженості середовищах, як *permissioned blockchain*, можна застосовувати CRDT для зменшення затримок і підвищення масштабованості, не жертвуючи при цьому конвергентністю станів.

В дослідження [62] був наданий огляд використання CRDT для синхронізації в швидкозмінних розподілених системах. Автори запропонували модель PS-CRDT (видавець/передплатник) для забезпечення просторового та часового розв'язання поширення оновлень. Таке рішення гарантує сумісність CRDT з динамічним входом та виходом вузлів у нестабільних середовищах.

В сучасних дослідженнях простежується значний інтерес до використання різних типів CRDT для представлення стану розподілених систем з метою усунення збурень і порушень, спричинених конфліктами між репліками. Ряд праць [69], [70] демонструють, що підходи на основі CRDT забезпечують високу ефективність, продуктивність, низьку затримку та добру масштабованість у контексті розповсюдження повідомлень, особливо в системах великого масштабу, побудованих за моделлю Pub/Sub. Отримані результати свідчать, що такі рішення є перспективним напрямом для досягнення сильної остаточної узгодженості [71], [72] у розподілених сховищах даних. Це робить їх особливо цінними для систем, що мають відповідати вимогам підвищеної відмовостійкості та резилієнтності.

1.3.5. Обмеження використання неконфліктних реплікованих структур даних.

Попри значні переваги CRDT у забезпеченні сильної остаточної узгодженості без потреби в глобальній координації, їх використання пов'язане з низкою обмежень, які визначають як сферу доцільності, так і виклики для подальших досліджень.

Насамперед, критикою часто піддається накладне зростання метаданих. У поданні CRDT обов'язково міститься інформація про історію змін, що з часом призводить до експоненціального накопичення метаданих. У дослідженні [73] показано, що це істотно ускладнює збереженням оперативної здатності до реагування та масштабованості, навіть якщо з'являються оптимізації для видалення зайвої інформації на основі стабільності операцій.

Другим викликом є складність побудови CRDT для складних структур даних, зокрема графів або схем, що мають інваріанти цілісності, наприклад, *property graph*. У роботі [74] наголошується, що наявні CRDT не підтримують схеми графів, а коректна обробка таких випадків вимагає окремих дизайнів і складних алгоритмічних рішень.

Окремо варто відзначити складність програмного використання. У дослідженні [75] виділено, що розробники часто стикаються з труднощами у застосуванні CRDT-бібліотек, зокрема з реалізацією таких функцій, як операція *undo* — що надто складно додається вручну у існуючі бібліотеки і спричиняє помилки та труднощі у підтримці коду.

Таким чином, хоча CRDT представляють рішення для масштабних, асинхронних систем, їх практичне застосування обмежене через витрати на метадані, складність реалізацій для складних даних, ризики на рівні моделей поведінки та складнощі інтеграції.

1.4. Модель акторів та функціональна парадигма програмування у високонавантажених системах

Розвиток високонавантажених розподілених систем стикається з фундаментальними викликами: насамперед потребою в масштабованості, стійкості до збоїв та мінімізації комунікаційних витрат. Класичні синхронні механізми — зокрема блокування, монітори або імперативні багатопоточні підходи — часто породжують ускладнення конкурентності, включаючи взаємне блокування (англ.: deadlock) чи гонки даних (англ.: race conditions), що негативно впливає на продуктивність і підтримку системи в умовах великих масштабів [37].

Модель акторів [76] пропонує альтернативний підхід до організації обчислень: кожен актор має власний локальний стан і взаємодіє з іншими виключно через асинхронні повідомлення, що усуває потребу у блокуваннях та підвищує адаптивність системи до збоїв та змін топології. Ці властивості безпосередньо узгоджуються з функціональною парадигмою: незмінні (immutable) повідомлення та дані зменшують ризики конкурентного доступу, а обробники подій, реалізовані як наближено чисті функції з чітко локалізованими побічними ефектами, спрощують формальне міркування, тестування й налагодження. У підсумку поєднання моделі акторів та функціонального програмування забезпечує ізоляцію станів, передбачувану композицію компонентів і масштабовану паралельність — критично важливі властивості для паралельних і розподілених систем.

У контексті роботи вибір моделі акторів та функціональної парадигми програмування обумовлений необхідністю розробки симуляційної моделі розподіленої енергетичної мережі, де агенти повинні взаємодіяти асинхронно обмінюючись повідомленнями. Актори забезпечують пряму відповідність такій агентній постановці завдання, оскільки кожен агент може бути відображений у вигляді незалежного актора з власним станом і правилами поведінки. Функціональний підхід, у свою чергу, гарантує коректність і передбачуваність логіки цих агентів завдяки імутабельності та відсутності прихованих побічних

ефектів. Таким чином, поєднання цих підходів формує методологічну основу для розробки відмовостійкої та масштабованої моделі BIEM.

1.4.1. Концепція моделі акторів

Модель акторів (англ.: Actor Model) була запропонована як формальний підхід до організації паралельних та розподілених обчислень. Її ключова ідея полягає у поданні обчислювальних процесів у вигляді множини незалежних об'єктів — акторів, які функціонують у спільному середовищі та взаємодіють між собою виключно шляхом асинхронного обміну повідомленнями [76], [77].

Кожен актор характеризується трьома основними складовими:

- поштова скринька (англ.: mailbox) — черга для приймання вхідних повідомлень;
- поведінка (англ.: behaviour) — функція або набір правил, що визначають реакцію актора на повідомлення;
- стан (англ.: state) — внутрішні дані актора, які відображають його поточний контекст у певний момент часу.

Обробка повідомлень виконується послідовно, одне за одним, що дозволяє уникати класичних ускладнень конкурентності, таких як взаємне блокування чи гонки даних. Водночас актори існують у межах так званої акторної системи (англ.: actor system), яка визначає ієрархію та правила взаємодії між ними, забезпечуючи масштабованість та можливість організації багаторівневих структур [78].

Використання акторної моделі забезпечує низку переваг, особливо у випадках розробки розподілених, асинхронних та високонавантажених систем. Основні переваги застосування акторної моделі [7], [78], [79] можна узагальнити таким чином:

- Природне моделювання асинхронних процесів. Модель акторів є відповідною для асинхронного програмування, де завдання виконуються паралельно й незалежно одне від одного. Обмін повідомленнями між

акторами відбувається без блокування, що спрощує створення паралельних процесів без необхідності синхронізації потоків.

- Покращена масштабованість. Модель дозволяє легко додавати нові актори або розподіляти їх між вузлами з мінімальними змінами у коді. Це робить можливим масштабування як вертикальне (на одному сервері), так і горизонтальне (між кількома серверами чи вузлами).
- Ізоляція стану. Кожен актор має власний стан, який є недоступним безпосередньо для інших акторів. Це усуває потребу у блокуваннях спільних змінних і знижує ризик помилок, пов'язаних із синхронізацією. Актори можуть незалежно оновлювати свій стан, що підвищує надійність системи.
- Висока резилієнтність і самовідновлення. Модель акторів передбачає вбудований механізм нагляду, у якому один актор може контролювати інший. У випадку відмови “наглядач” перезапускає підлеглого актора чи виконує коригувальні дії, що забезпечує автоматичне відновлення. Це особливо важливо для розподілених систем, які повинні гарантувати безперервність роботи.
- Спрощення розподілених обчислень. Оскільки актори обмінюються повідомленнями, вони можуть існувати на різних вузлах розподіленої системи. Взаємодія акторів можлива як локально, так і через мережу, що значно спрощує розробку розподілених застосунків, де фізичне розташування компонентів не має вирішального значення.
- Висока продуктивність завдяки асинхронній обробці. Оскільки актори не блокують потоки під час обробки повідомлень, система здатна одночасно опрацьовувати велику кількість запитів. Це забезпечує високу продуктивність навіть за умов значного навантаження, оскільки відсутні затримки через блокування ресурсів.
- Природне керування паралелізмом. В акторній системі кожен актор виступає як ізольована одиниця паралельних обчислень. Це спрощує розподіл робочих навантажень між різними ядрами процесора або навіть

між різними машинами, уникаючи складнощів, притаманних класичній багатопоточності.

- Зручність розробки складних систем. Модель акторів придатною для застосунків, у яких об'єкти мають складну поведінку чи часто взаємодіють між собою. Замість створення складних механізмів блокування та синхронізації, актори використовують обмін повідомленнями, що суттєво спрощує розробку та підтримку коду.

Таким чином, модель акторів поєднує у собі властивості, критично важливі для інтелектуальних енергетичних мереж: асинхронність обробки подій, масштабованість, стійкість до збоїв і можливість природного відображення гетерогенних учасників системи. Це робить її придатною основою для моделювання та реалізації інтелектуальних енергетичних мереж, де необхідно забезпечити надійну взаємодію великої кількості автономних агентів у розподіленому середовищі [79].

1.4.2. Функціональна парадигма в контексті розподілених систем

Функціональна парадигма програмування розглядає обчислення як застосування функцій до аргументів, уникаючи змінюваного стану та побічних ефектів. Вона базується на принципах лямбда-числення, чистоти функцій (англ.: pure functions) та декларативного стилю опису обчислень, що забезпечує прозорість, передбачуваність та математичну строгість [80].

Історично функціональні мови програмування (LISP, Haskell, Erlang, OCaml) формувалися як інструменти для математичного моделювання та роботи з формальними системами. Згодом їхні концепції були інтегровані в мейнстримні мови (Java, C#, Scala, Python), що підтвердило їхню придатність для вирішення інженерних завдань у високонавантажених розподілених середовищах [81]. Сучасні фреймворки (Apache Spark, Flink, Akka Streams) демонструють, що саме функціональний стиль забезпечує простоту вираження паралельних і потокових обчислень, які становлять основу Big Data застосунків.

У контексті розподілених систем функціональна парадигма надає такі переваги [82], [83], [84], [85], [86]:

- Нематеріальність стану та ідемпотентність обчислень. Чисті функції, що завжди повертають однаковий результат для однакових вхідних значень, спрощують забезпечення узгодженості й стійкості системи навіть за умов копіювання або втрати даних.
- Природна паралельність. Відсутність спільного змінюваного стану дає змогу безпечно розпаралелювати виконання функцій між потоками або вузлами, що є критично важливим у масштабованих, навантажених системах.
- Спрощене відтворення та налагодження. Відсутність побічних ефектів означає, що обчислення залежать лише від вхідних параметрів, що підвищує передбачуваність коду та полегшує тестування і відладку.
- Композиційність. Функціональна парадигма дозволяє створювати складні обчислювальні процеси шляхом поєднання простих функцій. У розподілених системах це забезпечує гнучкість та модульність у побудові обробних конвеєрів і адаптивної архітектури — без втрат стабільності.
- Застосування у сучасних розподілених платформах. Функціональні концепції, такі як “передача функцій” (англ.: *passing functions*) над розподіленими, незмінними даними (англ.: *immutable data*), широко застосовуються в платформах для обробки великих даних, наприклад, Apache Spark. Вони спрощують відновлення після несправностей, оскільки достатньо повторно застосувати функції до незмінних даних, а не працювати зі станом.
- Абстракція над складністю розподілених обчислень. Функціональні конструкції дають змогу абстрагуватися від деталей конкурентності, мережових затримок, втрат повідомлень і збоїв, при цьому зберігаючи композиційність і декларативність моделі обчислень.

Додатковою перевагою функціональних підходів є їхня придатність до формальної верифікації. У розподілених системах, де складність взаємодій між компонентами швидко зростає, можливість перевіряти властивості програм за допомогою математичних методів (наприклад, теореми про коректність або

моделі перевірки) дозволяє знижувати ризики системних збоїв [87]. Це особливо важливо у критичних доменах, таких як енергетика, де некоректна поведінка системи може призвести до масштабних відмов.

Поєднання функціональної парадигми з акторною моделлю пропонує ефективний підхід до проєктування та створення розподілених систем. Декларативність, композиційність та відсутність побічних ефектів, властиві функціональним методам, органічно поєднуються з ізоляцією станів та асинхронною взаємодією акторів, забезпечуючи побудову масштабованих і відмовостійких BIEM.

Висновки розділу 1

У розділі 1 дисертаційної роботи здійснено огляд актуальних наукових досліджень у галузі BIEM та підходів до їх моделювання і управління. Увагу приділено мультиагентним системам, які розглядаються як ефективний інструмент для розподіленого прийняття рішень та координації у складних кіберфізичних середовищах. Аналіз літератури показав, що мультиагентні системи дедалі ширше застосовуються для вирішення завдань балансування попиту та пропозиції, забезпечення резилієнтності, кібербезпеки, відновлення та симуляційного моделювання.

Застосування мультиагентних систем має низку обмежень. Це, зокрема, складність забезпечення узгодженості станів у розподіленому середовищі, висока вартість комунікацій та ризики затримок у реальному часі, гетерогенність агентів і протоколів взаємодії. Традиційні методи узгодженості, засновані на централізованих координаторах або блокуючих протоколах, демонструють обмежену ефективність у масштабних і динамічних мережах. Це зумовлює активний розвиток сучасних підходів, таких як алгоритми консенсусу нового покоління (Raft, ZAB, BFT-протоколи) та CRDT, здатні забезпечувати остаточну узгодженість у децентралізованих умовах.

Огляд досліджень показує, що агентний підхід до моделювання BIEM може реалізовуватися з використанням акторної моделі, яка забезпечує

асинхронну взаємодію та ізоляцію станів агентів. Функціональна парадигма програмування, застосована під час розробки відповідного програмного забезпечення, дозволяє формально описувати поведінку агентів і процесів, роблячи модель прозорою, строго визначеною та придатною до формальної верифікації. Використання CRDT своєю чергою дасть змогу забезпечити детерміновану конвергенцію локальних подань стану без глобальної координації, зменшить потребу в блокуваннях і підвищить стійкість до збоїв. Поєднання цих концепцій спрощує організацію паралельних процесів, знижує ризики системних збоїв та підвищує надійність результатів. *Таким чином, розроблення підходу до агентного моделювання високоінтелектуальних енергетичних мереж із використанням неконфліктних реплікованих структур даних для опису станів системи є актуальним науковим завданням.*

РОЗДІЛ 2. ФОРМАЛІЗАЦІЯ ТА ПРОЄКТУВАННЯ АГЕНТНОЇ МОДЕЛІ РОЗПОДІЛЕНОЇ ЕНЕРГЕТИЧНОЇ МЕРЕЖІ

Агентний підхід до моделювання ВІЕМ дозволяє розглядати кожен елемент системи як автономного учасника з власним станом та правилами поведінки. Проте для побудови коректної та відтворюваної моделі важливим є як формалізоване подання станів як окремих вузлів, так і системи загалом. Сукупність локальних станів усіх агентів визначає глобальний стан розподіленої електричної мережі, відображаючи як індивідуальні характеристики, так і взаємодії між компонентами [7], [88].

До ключових аспектів такого представлення належать [7], [89]: доступність вузлів (наявність у мережі та можливість обміну даними), інформація про ресурси (обсяг виробництва, накопичення чи споживання енергії), мережеве навантаження (ступінь використання каналів обміну та ризики перевантажень) та стабільність з'єднань (надійність каналів комунікації та їхня стійкість до збоїв). Сукупність цих параметрів визначає, наскільки ефективно мережа може виконувати свої функції в умовах динамічного середовища.

Моделювання подій, пов'язаних із передаванням електроенергії споживачам, ускладнюється їх дискретно-подійною природою та мережевими обмеженнями. Вони поділяються на безперервні процеси (відхилення від номінальних характеристик, спричинені змінами навантаження чи нелінійною природою споживачів) та випадкові події напруги (раптові та значні відхилення, обумовлені аваріями, погодними умовами чи зовнішніми впливами). До першої категорії відносяться коливання частоти та амплітуди струму, а до другої — перебої живлення, провали чи перенапруги. Ці явища носять стохастичний характер, нерівномірно розподілені в часі й залежать від топології та експлуатаційних умов енергосистеми [7].

2.1. Формалізація агентної моделі

У процесі побудови моделі розподіленої електричної мережі було закладено принцип, за яким зміна стану кожного агента розглядається як залежна не лише від його власних характеристик у поточний момент часу, але й від зовнішньої інформації про стани інших агентів системи. Це дає змогу відобразити взаємозалежність елементів мережі та забезпечити моделювання їхньої поведінки у динамічних умовах функціонування.

Загальна модель високоінтелектуальної енергетичної мережі подається у вигляді множини моделей рівноправних агентів, кожен з яких відображає енергетичного постачальника з певними статичними та динамічними характеристиками. До статичних характеристик належать унікальний ідентифікатор агента та його географічна належність. Динамічні характеристики включають параметри, зокрема номінальну та фактичну потужність. Сукупність цих характеристик визначає індивідуальний стан кожного агента в моделі.

Для коректного представлення стану агента необхідно виокремити характеристики, що є унікальними, стабільними та відносно рідко змінюваними, але водночас мають істотний вплив на прийняття рішень системою управління. До таких характеристик віднесено, зокрема, показники номінальної та фактичної потужності.

Оскільки модель відображає стан системи, зберігання фактичних значень зазначених показників у її структурі є недоцільним, адже це зумовило б формування нескінченної множини можливих станів, що безперервно змінюються та потребують постійної реплікації. Враховуючи, що незначні коливання показників є допустимими та, як зазначалося вище, зазвичай не призводять до зміни загального стану, використання нескінченної множини станів не є обґрунтованим. Натомість стан системи доцільно представити у вигляді обмеженої множини з трьох можливих станів (Рис. 2.1).

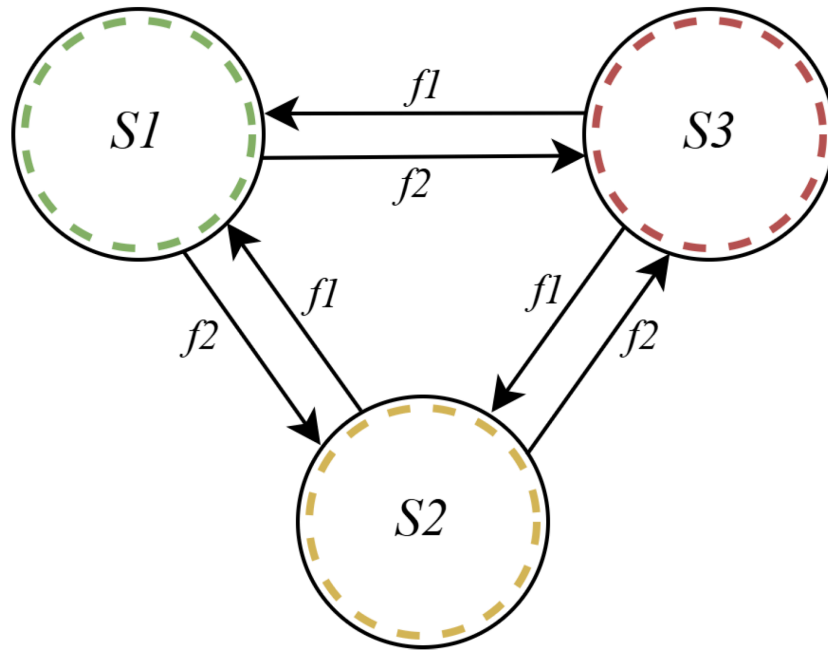


Рис. 2.1 Діаграма переходів станів

На цій фігурі $S1$ (зелений) позначає, що агент має надлишкову потужність; $S2$ (жовтий) вказує, що пропозиція відповідає попиту; а $S3$ (червоний) позначає, що попит перевищує пропозицію.

Бажаним станом будь-якого вузла є S_1 , тоді як S_3 репрезентує критичний стан із найвищим пріоритетом балансування серед сусідніх агентів системи. Перехід f_1 позначає зміну з більш бажаного стану до менш бажаного, що визначається умовою $f_1: P_{actual} < P_{nominal}$, тоді як f_2 описує протилежний перехід, зумовлений умовою $f_2: P_{actual} > P_{nominal}$. Простір станів, представлений як множина можливих станів відносно поточного, визначається наступним чином: $\{S1 : \{S2, S3\}, S2 : \{S1, S3\}, S3 : \{S1, S2\}\}$.

Простір станів, з переходами, пов'язаними з відповідними умовами, подано у таблиці 2.1.

Таблиця 2.1. Простір станів агента

| | S1 | S2 | S3 |
|----|----|----|----|
| S1 | 0 | f2 | f2 |
| S2 | f1 | 0 | f2 |
| S3 | f1 | f1 | 0 |

Синхронізація даних у розподілених системах визначається як процедура узгодження множини копій даних (станів) з метою підтримання їхньої цілісності. Ключовим механізмом цього процесу є консенсус — досягнення спільної згоди між вузлами розподіленої платформи. Забезпечення консенсусу є необхідною умовою надійності та відмовостійкості: у децентралізованому середовищі, де окремі учасники можуть діяти автономно чи навіть зловмисно, система повинна зберігати узгоджене уявлення про стан [90].

У процесі обміну даними в мережах, де кожен агент має власну копію даних, забезпечення узгодженості та вирівнювання станів є передумовою узгодженої роботи мережі. Коли значна кількість агентів намагається модифікувати дані одночасно, виникають конфлікти, що потребують безперервного й ефективного розв’язання з метою уникнення розбіжностей у даних. Відтак вирішення цього завдання полягає у розробці методів і підходів, які унеможливають виникнення конфліктів та гарантують узгодженість даних у розподілених мережах [7].

2.3. Підхід синхронізації станів агентів на основі неконфліктних реплікованих структур даних

У роботі запропоновано підхід синхронізації станів агентів БАС на основі CRDT, для забезпечення детермінованої збіжності реплік без глобальної координації.

CRDT [64] є абстрактним типом даних зі статично визначеним інтерфейсом, що забезпечує розподілену реплікацію на кількох вузлах. Він дозволяє здійснювати оптимістичну реплікацію [91] у формалізований спосіб. Кожна репліка може змінюватися за потреби й незалежно, без додаткової координації між вузлами. Якщо будь-які дві репліки отримали однакову множину оновлень, навіть у різному порядку, то завдяки застосуванню математично обґрунтованих правил вони детерміновано досягають одного й того ж стану, забезпечуючи збіжність (англ.: convergence) [92]. Більше того, якщо одночасні оновлення певного елемента даних є комутативними, і всі репліки виконують усі оновлення у причинно-узгодженому порядку, значення реплік збігаються. Такий тип визначається як комутативний реплікований тип даних [92]. Його призначенням є забезпечення комутативності одночасних операцій. Це усуває потребу у складних механізмах контролю конкурентності, дозволяючи виконувати операції у довільному порядку за умови збіжності результатів реплік. Таким чином, підхід гарантує відсутність конфліктів і знімає необхідність у механізмах узгодження на основі консенсусу [64], [93].

Використання CRDT для моделювання стану розподіленої системи має низку переваг [7], [63], [94], [95], [96], [97]:

- Комутативність та збіжність структур даних. CRDT проєктуються так, щоб бути комутативними та збіжними. Це означає, що кілька копій одних і тих самих даних можуть оновлюватися незалежно, а потім об'єднуватися без конфліктів. Така властивість забезпечує ефективний і точний аналіз даних навіть у випадках відключення мережі або за умов високого навантаження.
- Формалізовані правила розв'язання конфліктів. Підхід CRDT ґрунтується на попередньо визначених правилах усунення конфліктів, які задають їхню семантику. Зазвичай ці правила є специфічними для конкретної структури даних. CRDT забезпечують механізм виявлення й розв'язання конфліктів, підтримуючи узгодженість між усіма копіями даних. Це

особливо важливо в електроенергетичних системах, де інформація про стан може зберігатися у багатьох різних місцях.

- Використання стано-орієнтованих структур. При виборі CRDT, що зберігають поточний стан даних, а не історію змін (англ.: state-based CRDT), вони стають придатними для моніторингу та аналізу даних у режимі реального часу в енергетичних мережах.
- Масштабованість. CRDT дозволяють будувати розподілені системи, які здатні масштабуватися горизонтально, не потребуючи складних механізмів координації.
- Відмовостійкість. CRDT забезпечують відмовостійкість, оскільки допускають реплікацію даних між вузлами та їх відновлення у випадку відмови одного чи кількох вузлів.

Порівняно з методами, що базуються на блокуваннях/консенсусі, CRDT мають низку переваг, узагальнених у табл. 2.2. Ці переваги особливо релевантні в контексті синхронізації, притаманної розподіленим електроенергетичним системам, зокрема синхронізації розподілених генераторів із силовою електронікою [98] або керування синхронізацією каскадних відмов у енергомережах [99].

Таблиця 2.2. Порівняння синхронізації на основі CRDT та методів, що базуються на блокуваннях/консенсусі

| Критерій | CRDT-орієнтовані методи | Методи на основі блокувань/консенсусу |
|---------------------------|-----------------------------|---------------------------------------|
| Модель узгодженості | Остаточна узгодженість | Сильна узгодженість |
| Затримка (англ.: latency) | Низька (локальні оновлення) | Вища (потребує координації) |
| Доступність | Висока | Залежить від кворуму/лідера |

| Критерій | CRDT-орієнтовані методи | Методи на основі блокувань/консенсусу |
|-----------------------|--|--|
| Обробка конфліктів | Автоматична та детермінована | Явне блокування або рішення кворуму |
| Масштабованість | Висока | Помірна |
| Сценарії використання | системи спільної роботи в реальному часі | критично важливі, транзакційні системи |

Сукупність зазначених переваг забезпечує можливість формування систем з підвищеною відмовостійкістю та резилієнтністю.

У даній роботі використано CvRDT із можливістю поширення дельти змін між вузлами. Вибір стано-орієнтованої структури CRDT [63], яка зберігає поточний стан даних, а не історію змін, робить її придатною для моніторингу та аналізу даних в електроенергетичних мережах у режимі реального часу.

Для опису загального стану системи на основі CRDT, що репрезентує тип даних *Map* і може бути реплікований на всьому наборі агентів, відповідно до [100](с. 6), може бути використана модифікація *Observed Remove Set (ORSet)* [63](с. 26). Оскільки контейнерний тип *Map* базується на структурі *Set* [63](с. 21), його реалізація матиме ті самі семантичні властивості, що й ORSet, за винятком випадків конкурентних оновлень значень, які потребують специфічного механізму злиття. Для розв'язання цього завдання застосовано CRDT типу *LWWRegister* [63](с. 19). Використання цього типу даних гарантує збереження лише останнього (найновішого) записаного значення, тоді як попереднє (якщо воно існувало) відкидається. Оскільки у дослідженні нас цікавить актуальний (поточний) стан конкретного вузла в загальній системі, така стратегія є оптимальною.

Формально LWW-Register може бути визначений як множина, що є або порожньою, або містить одне значення:

$\{v \mid wr(v) \in O \wedge \nexists wr(u) \in O \cdot wr(v) < wr(u)\}$ [101](с. 4). Слід також зазначити, що як ORSet, так і LWWRegister належать до класу Delta State Replicated Data

Types [102], що зменшує потребу передавати повний стан під час оновлень. Інакше кажучи, при зміні стану об'єкта передається лише сама зміна, а не весь об'єкт із новим станом. Наприклад, додавання елементів c та d до множини $\{a, b\}$ призведе до надсилання дельти $\{c, d\}$ та об'єднання цього стану з наявним на стороні отримувача, у результаті чого буде сформовано множину $\{a, b, c, d\}$. Таким чином, можна сформулювати просту формулу зміни стану наступним чином:

$$S'_i = S_i \circ \Delta_i \quad (2.1)$$

де S'_i — новий стан вузла i , S_i — попередній стан вузла i ; Δ_i — зміна, застосована до вузла i ; символ \circ позначає операцію злиття змін.

Загальний стан системи буде представлений за допомогою типу даних *Map*, що ґрунтується на *ORSet* і має подібну семантику обробки ключів. Проте у випадку конкурентних оновлень значень будь-який потенційний конфлікт вирішуватиметься на користь останнього записаного значення. Концептуальне уявлення того, як зміни стану CRDT поширюються та збігаються під час конфліктів між різними вузлами для системи, описаної у цій роботі, наведене на рисунку 2.2. Процес поширення стану, показаний на рисунку, є типовим для будь-яких конкурентних змін у розподіленій енергетичній мережі, зокрема миттєвих оновлень станів комутаційних елементів, спричинених змінами на рівні низькорівневих сенсорів.

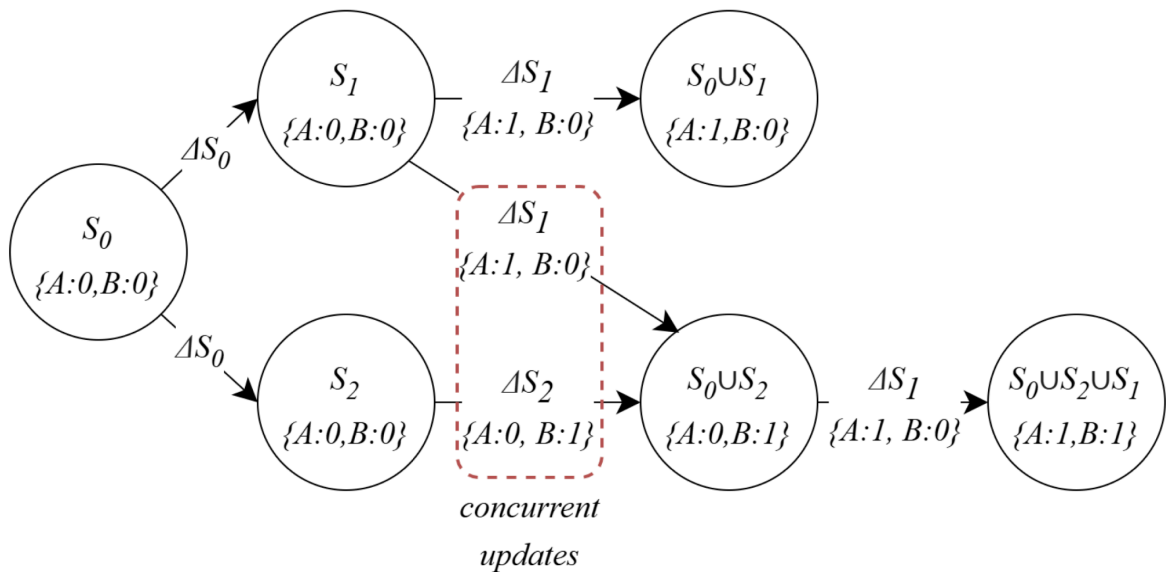


Рис. 2.2. Концептуальна схема розповсюдження станів CRDT між агентами, що оновлюються конкурентно.

На основі рівняння 2.1 можна визначити набір даних, який підлягає поширенню між вузлами, таким чином:

$$\Delta i = S'i - S_i \quad (2.2)$$

Семантично кожна дельта відображає набір змін певних параметрів стану агента, які поширюються до інших агентів та зливаються з їхнім внутрішнім станом. Наприклад, на основі рисунка 2.2 можна визначити різні дельти: $\Delta S1$ та $\Delta S2$. Використовуючи рівняння 2.2, ці дельти можна переписати у такій формі:

$$\Delta S1 = S'1 - S1 = \{A:1, B:0\} - \{A:0, B:0\} = \{A:1, B:0\},$$

$$\Delta S2 = S'2 - S2 = \{A:0, B:1\} - \{A:0, B:0\} = \{A:0, B:1\}$$

Можна простежити, як різні дельти, $\Delta S1$ та $\Delta S2$, збігаються під час конкурентних оновлень шляхом послідовного виконання операцій над станом $S2$. Результуючий стан становитиме об'єднання змін з $\Delta S1$ та $\Delta S2$, що може бути переписано у формі:

$$S0 \cup S1 \cup S2 = (S0 \circ \Delta S1) \circ \Delta S2 = (\{A:0, B:0\} \circ \{A:1, B:0\}) \circ \{A:0, B:1\} = \{A:1, B:1\}$$

Дельти формуються на основі операцій, визначених структурою даних CRDT, які повинні задовольняти властивості ідемпотентності, комутативності та асоціативності.

Представлений набір даних має дискретну природу, що ефективно відображає реальні сценарії, оскільки розподілені подання станів зазвичай оперують кількісними значеннями або категоріями, а не неперервними діапазонами. Дискретні стани спрощують моделювання, відображаючи ключові характеристики поведінки системи, переходи та взаємодії, такі як аварійні стани, режими роботи чи доступність ресурсів.

Загальне представлення високоінтелектуальної енергетичної мережі на основі CRDT може бути записане у такій формі:

$$Map[K, LWWRegister[V]] \rightarrow ORSet[(K \rightarrow LWWRegister)]$$

де *Map* позначає структуру даних типу *key* \rightarrow *value* з узагальненим типом *K* для ключа та визначеним станом вузла *V* для значення (Рисунок 2.3).

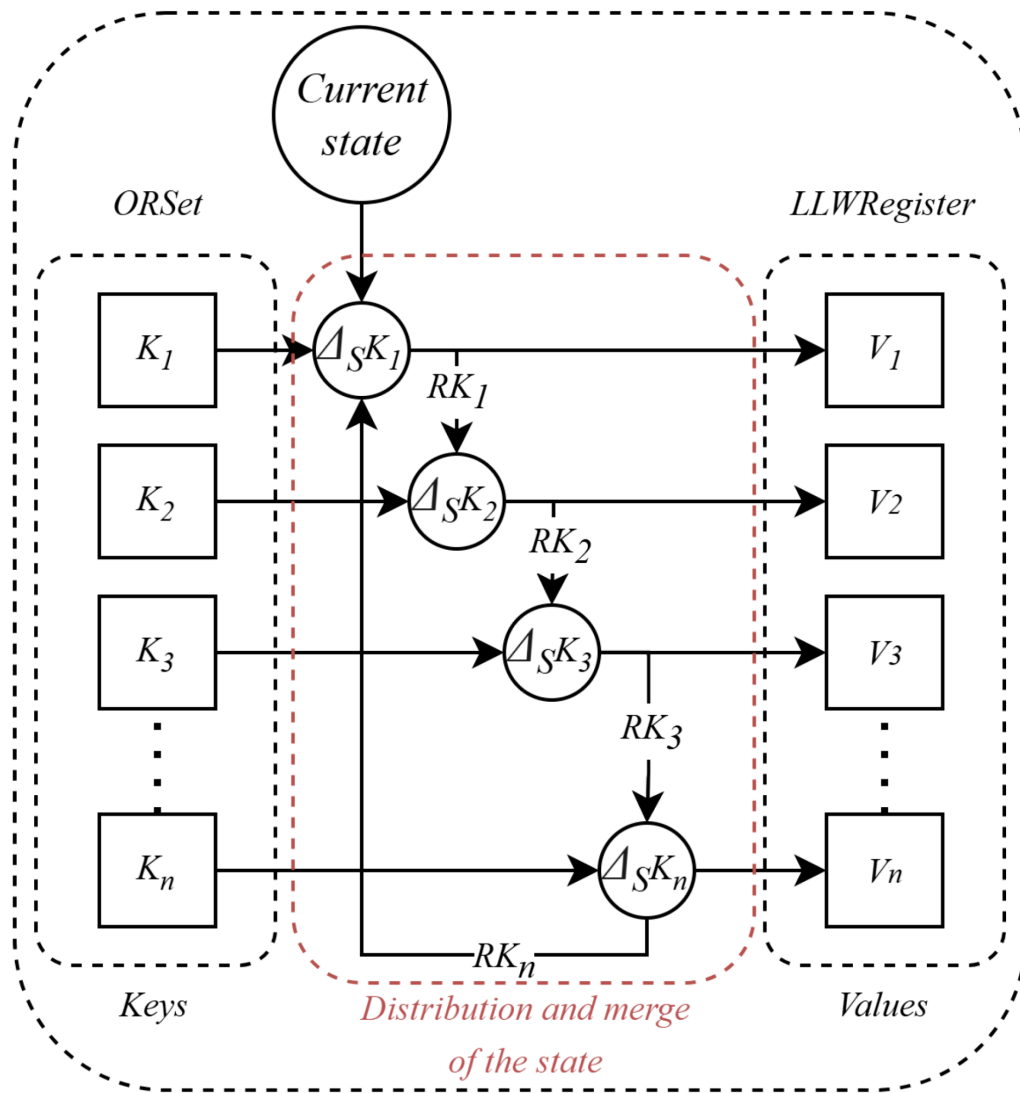


Рис. 2.3. Загальне подання високоінтелектуальної енергетичної мережі, побудоване на основі CRDT.

Тут *Keys* представляють ORSet усіх можливих ідентифікаторів вузлів, а *Values* — структури LLWRegister, що містять відповідні стани. Поняття *Current state* позначає стан S конкретного вузла після початкового збурення ΔS , яке використовується для порушення балансу системи шляхом безперервної синхронізації з іншими вузлами через розповсюдження дельт. Синхронізація певного вузла породжує унікальний результат RKi , який може бути переданий наступному вузлу $i+1$. Результат RKn відображає підсумок синхронізації для n -го вузла, що включає зворотний вплив початкових умов. Таким чином, досягнення балансу у всій системі означає відсутність передавання дельт або виконання умови $RKn = 0$. Таке подання підкреслює характер оновлень,

зумовлений специфікою реалізації структур ORSet та LWWRegister, що є критично важливим для інтерпретації експериментальних результатів.

Якщо певний тип даних задовольняє всі три властивості — ідемпотентність, комутативність та асоціативність, — то доведено, що він є конвергентним реплікованим типом даних.

Ідемпотентність. CRDT складається з множини всіх доданих елементів та множини всіх видалених елементів. Якщо здійснити повторну спробу виконати операцію *add* або *remove* над одним і тим самим елементом, структура CRDT залишиться незмінною. Така властивість запобігає дублюванню й забезпечує цілісність типу даних, дозволяючи CRDT поводитися узгоджено у розподілених системах, підтримуючи збіжність і остаточну узгодженість.

Додавання (операція *add*) або видалення (операція *remove*) елемента з CRDT може бути формалізовано за допомогою рівняння 2.3, де *add* та *remove* є ідемпотентними операціями:

$$\begin{aligned} add(a, Si) &= (S' \cup \{a\}) = (S' \cup \Delta i) = (S' \cup (S' \cup -Si)) = add(a, add(a, Si)) \\ remove(a, Si) &= (S' \cup \{a\}) = (S' \cup \Delta i) = (S' \cup (S' \cup -Si)) = remove(a, remove(a, Si)) \end{aligned} \quad (2.3)$$

Комутативність. Операції додавання (*add*) та видалення (*remove*) не залежать одна від одної, а отже, порядок їх виконання не впливає на кінцевий результат. Цю властивість можна описати, спираючись на рівняння 2.3, у такій формі:

$$\begin{aligned} add(a, Si) \rightarrow add(b, Si) &= add(b, Si) \rightarrow add(a, Si) \\ remove(a, Si) \rightarrow remove(b, Si) &= remove(b, Si) \rightarrow remove(a, Si) \end{aligned} \quad (2.4)$$

Асоціативність. Результат виконання операцій не залежить від порядку їх застосування, незалежно від того, як саме вони групуються. Цю властивість можна описати на основі рівняння 2.4 у такій формі:

$$add(a, remove(b, Si)) = remove(a, add(a, Si))$$

Ідемпотентність, комутативність та асоціативність встановлюють фундаментальні правила, які визначають множину можливих функцій та операторів, пов'язаних із описаними CRDT. Ці принципи також окреслюють діапазон допустимих дельт, що можуть поширюватися у межах конкретних топологій системи. Дотримання зазначених правил забезпечує повноцінне використання CRDT-структур даних, водночас підтримуючи передбачуваність

та впорядкованість обчислень, пов'язаних із безперервними та конкурентними оновленнями, навіть за умови, що кожен вузол оновлює лише власну частину стану. Будь-які CRDT-операції, які розглядатимуться надалі, слід розглядати як комбінацію описаних вище операцій *add* та *remove*.

Висновки розділу 2

У розділі 2 було здійснено формалізацію та проектування агентної моделі розподіленої енергетичної мережі, що поєднує засади агентного підходу з використанням CRDT для опису станів системи. Формалізовано структуру системи, у межах якої вузли представлені як рівноправні агенти з чітко визначеними статичними й динамічними атрибутами; визначено множину станів і правила переходів, що описують основні режими функціонування та взаємодії.

Проаналізовано аспекти синхронізації даних у розподілених середовищах і обґрунтовано доцільність використання CRDT як альтернативи блокуванням і консенсусу.

Запропоновано підхід синхронізації станів агентів на основі CvRDT з передаванням дельт-оновлень.

Показано, що використання запропонованого підходу на основі CvRDT із передаванням дельт змін забезпечує ефективність, відмовостійкість і горизонтальну масштабованість. Формально доведені властивості ідемпотентності, комутативності та асоціативності операцій гарантують детерміновану конвергенцію станів без необхідності складних механізмів керування конкурентністю.

Запропоноване узагальнене представлення BIEM у вигляді структури `Map[Key, LWWRegister[Value]]` продемонструвало здатність відображати динаміку змін, включно зі стохастичними збуреннями, із одночасним збереженням узгодженості у масштабованому середовищі.

РОЗДІЛ 3. РОЗРОБКА СИМУЛЯЦІЙНОЇ МОДЕЛІ РОЗПОДІЛЕНОЇ ЕНЕРГЕТИЧНОЇ МЕРЕЖІ

Формалізація структури та механізмів узгодження станів у BIEM формує методичні засади переходу до побудови симуляційної моделі, яка дозволяє перевірити працездатність теоретичних положень у динамічних сценаріях. Розробка такої моделі надає можливість відтворення поведінки BIEM в умовах стохастичних збурень, зміни навантаження, відмов вузлів та непередбачуваних зовнішніх впливів.

Симуляційна модель слугує інструментом для оцінки ефективності агентного підходу та CRDT-орієнтованих методів синхронізації, а також для аналізу стійкості й масштабованості системи.

Важливим аспектом є можливість моделювання у реальному часі процесів, що відбуваються у BIEM, де асинхронна взаємодія, децентралізоване прийняття рішень та висока інтенсивність інформаційних потоків формують складні сценарії. Враховуючи використання моделі акторів та CRDT, симуляційна модель покликана забезпечити формальне відтворення процесів синхронізації станів між агентами та оцінку їхньої ефективності в умовах динамічного середовища.

3.1. Вибір інструментів реалізації.

Розробка агентної моделі високоінтелектуальної енергетичної мережі потребує використання інструментів, здатних одночасно забезпечити гнучкість програмування, підтримку розподілених обчислень і роботу в умовах асинхронної взаємодії між вузлами. Оскільки модель ґрунтується на великій кількості автономних агентів, що обмінюються повідомленнями та координують власні дії у динамічному середовищі, це зумовлює необхідність обрання середовища, яке забезпечує масштабованість, стійкість до збоїв і структурну підтримку паралельності.

Традиційні імперативні підходи (наприклад, реалізації на Java з використанням потоків та механізмів синхронізації) забезпечують базовий

рівень паралелізму, проте ускладнюють управління станами та призводять до ускладнень конкурентності, таких як взаємне блокування чи гонки даних. Використання мов Python чи C++ дозволяє створювати високопродуктивні симуляційні середовища, але потребує додаткових інструментів для організації відмовостійких розподілених обчислень.

Мова програмування Scala [103] поєднує об'єктно-орієнтовану та функціональну парадигми, надаючи зручні засоби для створення абстракцій і роботи з асинхронними потоками даних. Завдяки інтеграції з екосистемою JVM забезпечується висока продуктивність і сумісність із перевіреними бібліотеками на Java. Scala є інструментом для реалізації розподілених систем і відрізняє її від чисто функціональних мов, зокрема Haskell, які обмежені в застосуванні для індустріальних рішень через слабшу підтримку промислових бібліотек та інструментів.

Фреймворк Akka [104] реалізує акторну модель обчислень і забезпечує високорівневий механізм управління акторними системами. На відміну від класичних бібліотек для багатопоточності, він підтримує прозору кластеризацію, маршрутизацію повідомлень та механізми автоматичного відновлення після збоїв.

3.1.1. Мова програмування Scala

Scala є сучасною мовою програмування, що поєднує в собі об'єктно-орієнтовану та функціональну парадигми. Такий підхід забезпечує високу виразність синтаксису та гнучкість у створенні моделей складних систем. У контексті моделювання розподілених енергетичних мереж це дозволяє одночасно використовувати інкапсуляцію об'єктно-орієнтованих структур і математичну строгість функціональних обчислень, що є важливим для опису агентів та їх взаємодій.

Однією з ключових характеристик Scala є тісна інтеграція з екосистемою JVM. Це забезпечує доступ до широкого спектра існуючих бібліотек і високий рівень сумісності з перевіреними технологічними рішеннями, що робить

можливим використання напрацювань, створених для Java, у нових проєктах без суттєвих витрат на адаптацію. Завдяки цьому Scala поєднує сучасні парадигмальні можливості з індустріальною зрілістю JVM-платформи.

Платформа надає засоби асинхронних обчислень і конкурентності, зокрема Future, Promise та інструменти керування потоками виконання і даних. Це дозволяє ефективно працювати в умовах високої динамічності системи та великої кількості паралельних взаємодій, що є типовим для BIEM.

У порівнянні з чисто функціональними мовами, такими як Haskell, Scala є більш орієнтованою на практичне застосування в індустрії завдяки сумісності з існуючими рішеннями та простішій інтеграції у розподілені системи. Водночас у порівнянні з класичними об'єктно-орієнтованими мовами (Java, C#) вона надає розширені можливості для декларативного опису поведінки системи та роботи з незмінюваними структурами даних, що сприяє підвищенню надійності й передбачуваності обчислень.

Використання Scala як основної мови програмування у даній роботі зумовлене її здатністю поєднувати переваги об'єктно-орієнтованого та функціонального підходів [77], підтримкою асинхронних і конкурентних обчислень, а також сумісністю з потужною інфраструктурою JVM.

3.1.2. Фреймворк Akka

Фреймворк Akka є інструментарієм для побудови розподілених, масштабованих та відмовостійких систем, що реалізує акторну модель обчислень. Його концептуальною основою є принципи реактивних систем [105] (англ.: Reactive Principles), які забезпечують асинхронність, еластичність, стійкість до збоїв і високу відтворюваність у динамічних середовищах. Використання акторів як універсальних структурних одиниць дозволяє моделювати вузли системи, їхній стан та взаємодію через обмін повідомленнями.

Основними перевагами використання акторної моделі [76], [77], [78] є її здатність відображати асинхронні процеси та забезпечувати управління

паралельними обчисленнями. Акторна модель відповідає вимогам програмування з конкурентним виконанням завдань: комунікація здійснюється асинхронними повідомленнями, що усуває потребу в блокуванні. Це спрощує створення паралельних процесів та усуває потребу у складних механізмах синхронізації.

Масштабованість забезпечується тим, що нових акторів можна додавати й розміщувати на різних вузлах із мінімальними змінами у кодї; це дозволяє вертикальне та горизонтальне масштабування застосунків. Кожен актор має власний ізольований стан, який не є безпосередньо доступним для інших, завдяки чому відпадає необхідність у блокуванні спільних ресурсів, а система стає більш надійною та стійкою до помилок синхронізації.

Додатковим механізмом забезпечення стійкості є вбудована система нагляду: у випадку збою наглядовий актор може перезапустити підлеглого або виконати інші коригувальні дії, що гарантує автоматичне відновлення та безперервність функціонування розподіленої системи. Оскільки актори комунікують виключно через повідомлення, вони можуть бути розподілені між різними вузлами й взаємодіяти як локально, так і через мережу, що істотно спрощує проектування розподілених обчислень. Завдяки асинхронній обробці повідомлень система здатна одночасно обробляти велику кількість запитів без блокувань ресурсів, демонструючи високу продуктивність навіть за умов інтенсивного навантаження.

Акторна модель також забезпечує природне управління паралелізмом, оскільки кожен актор функціонує як ізольована одиниця обчислення, що спрощує розподіл завдань між процесорними ядрами або фізично віддаленими машинами.

Нарешті, вона значно полегшує розробку складних систем: замість використання громіздких механізмів синхронізації актори регулюють власну поведінку через повідомлення, що робить код більш зрозумілим, підтримуваним і передбачуваним у динамічному середовищі [7].

Використання фреймворку Akka є доцільним для моделювання ВІЕМ, оскільки він забезпечує такі властивості [79]:

1. Akka ґрунтується на акторній моделі, яка є підходом до паралельних обчислень. Це забезпечує можливість репрезентації асинхронних взаємодій між розподіленими вузлами мережі через математичні примітиви та відповідні структурні залежності.
2. Akka працює з акторами, що виступають універсальними, незалежними та багаторівневими структурними одиницями в межах ієрархічної акторної системи. Кожен актор має власний стан і може використовуватися для моделювання будь-яких суб'єктно-об'єктних сутностей у енергетичній мережі та їхніх взаємодій і зв'язків. Актори здатні змінювати лише власний стан, впливаючи на систему опосередковано — через обмін повідомленнями, що не вимагає синхронізації на основі блокування стану.
3. Акторна система може інтегрувати зовнішні джерела інформації, такі як сенсори чи комутаційні пристрої, для оцінки та реагування на зміни в середовищі. Ієрархічна структура акторної системи та її здатність інкапсулювати доменні відносини незалежно від інших груп акторів створюють умови для фізичного розподілу або реплікації підсистем ВІЕМ на різних серверах.
4. Акторні повідомлення не мають прямого доступу до стану чи поведінки об'єкта-актора, що дозволяє паралелізувати процеси виконання внутрішньої логіки та мінімізує вплив внутрішнього стану одного актора на стан усієї акторної системи. Це дає можливість розробникам абстрагуватися від конкуренції потоків, водночас зберігаючи гнучкість у моделюванні об'єктів і сутностей енергетичної мережі.

Попри те, що нині існує значна кількість технологій та програмних фреймворків, які можуть бути використані для побудови розподілених систем, Akka вирізняється сукупністю властивостей, які роблять її придатною для

моделювання та управління BIEM. На відміну від мікросервісних архітектур [106] (наприклад, Spring Cloud [107]), основна увага яких зосереджена на оркестрації сервісів та реалізації RESTful-взаємодії, Akka пропонує вбудовану модель конкурентності, що базується на акторах. Актори інкапсулюють як стан, так і поведінку об'єкта, обмінюючись інформацією виключно за допомогою асинхронних повідомлень, що забезпечує високу масштабованість і природну підтримку паралельних процесів.

Інші розподілені платформи, зокрема Hazelcast, або комунікаційні рішення на кшталт JGroups, також надають розробникам засоби для організації спільного доступу до станів та реалізації групових комунікацій. Однак у їхньому функціоналі відсутня вбудована підтримка ієрархічного контролю за виконанням акторів, ізоляції збоїв та стратегій відновлення на рівні окремих виконавчих одиниць. У той же час застосування таких компонентів, як Akka Cluster та Akka Sharding, дозволяє не лише забезпечити фізичний розподіл обчислень, але й реалізувати еластичність масштабування, що є важливим для управління енергомережами, які географічно розподілені та характеризуються високим ступенем динамічності.

Крім того, у сфері багатоагентного моделювання існують спеціалізовані платформи, такі як JADE, Jason, Repast, проте вони зазвичай не забезпечують належного рівня інтеграції із сучасними моделями конкурентності. Також вони часто не мають прямої підтримки високорівневих абстракцій, необхідних для координації систем у режимі реального часу. Натомість акторна модель, реалізована у Akka Toolkit, виступає водночас концептуальною та практичною основою побудови стійких розподілених систем, у яких кожен компонент є незалежним, самодостатнім і відмовостійким за своєю архітектурою [79].

3.2. Архітектура симулятора: взаємодія агентів та поширення станів.

Симульований кластер складається з вузлів електроенергетичних одиниць. Кожен агент має власний унікальний ідентифікатор (*ID*), ідентифікатор регіону (*RI*), номінальну та фактичну потужність, дві множини з

ідентифікаторами інших агентів, а також значення запозиченої та поверненої електроенергії від/до цих агентів.

Структурне подання вузла наведено на рисунку 3.1, де *реплікатор* виступає абстрактною одиницею, відповідальною за поширення повідомлень, включно з операціями читання та запису будь-якої ΔS , між усіма вузлами. *Merger* є абстрактною одиницею, що відповідає за стратегію злиття, яка визначає процес оновлення поточного стану вузла No_0 до певного результату R з урахуванням конкурентних оновлень від різних вузлів. І реплікатор, і мерджер є уніфікованими складовими кожного вузла системи.

Формально агента системи можна подати у вигляді кортежу:

$$N = (ID, RI, Pnom, Pact, Sin, Sout, Rep, Mer)$$

де:

ID — унікальний ідентифікатор вузла;

RI — ідентифікатор регіону, до якого належить вузол;

Pnom — номінальна потужність;

Pact — фактична потужність;

Sin — множина ідентифікаторів інших вузлів, від яких вузол отримує (запозичує) потужності;

Sout — множина ідентифікаторів інших вузлів, яким вузол передає (повертає) потужності;

Rep (*replicator*) — абстрактний компонент, що відповідає за розповсюдження повідомлень (операції читання/запису для будь-якої ΔS) між вузлами;

Mer (*merger*) — абстрактний компонент, що відповідає за стратегію злиття, яка узгоджує поточний стан вузла No із результатом R під час конкурентних оновлень.

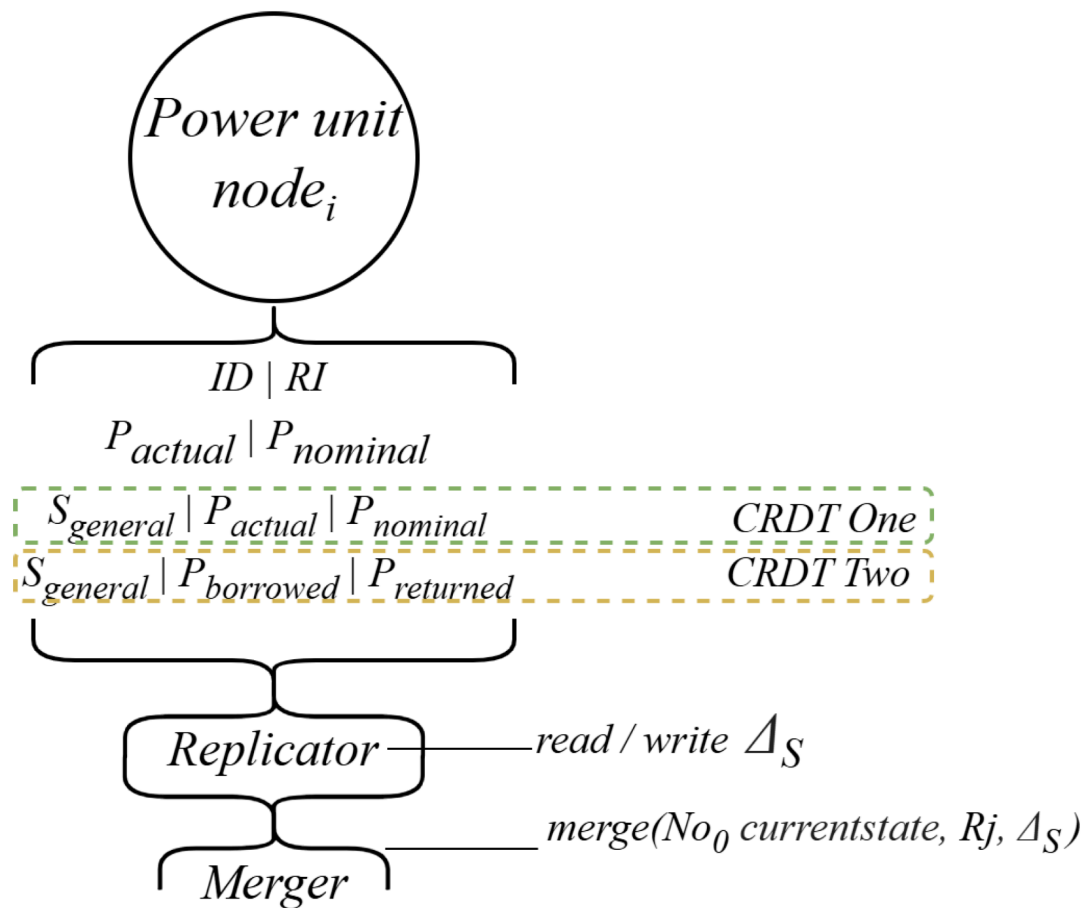


Рис 3.1. Структурна схема агента високоінтелектуальної енергетичної мережі.

Крім того, кожен агент має власний стан P_i , який відображає співвідношення між номінальною $P_{nominal}$ та фактичною P_{actual} потужністю:

$$P_i = \begin{cases} green \rightarrow if & P_{actual} \leq P_{nominal} \\ yellow \rightarrow if & P_{actual} = P_{nominal} \\ red \rightarrow if & P_{actual} > P_{nominal} \end{cases}$$

Кожен агент формує власне уявлення про стан мережі загалом на основі реплікованих даних $S_{i,t} \rightarrow ID \cup RI \cup P_{i,t} \cup S_{general,t} - S_{i,t-1}$, де $S_{i,t}$ — це стан агента в певний момент часу t , а вираз $S_{general,t} - S_{i,t-1}$ вказує на загальний стан системи відносно попереднього стану i -го агента у момент часу $t-1$, включно зі станами інших агентів. Кожен агент відповідає за публікацію змін власного стану в цій структурі даних та зчитування оновлень від інших агентів. У роботі реалізовано агентів, орієнтованих на цілі, з внутрішнім станом і локальною

моделлю середовища. Відповідно, перебуваючи у кластері та маючи уявлення про стан інших агентів, можуть бути розглянуті наступні сценарії роботи агента.

Сценарій дефіциту. Коли агент втрачає баланс і його стан переходить у *red*, він, орієнтуючись на своє уявлення про мережу, звертається до географічно найближчого агента зі станом *green* із запитом на запозичення потужності.

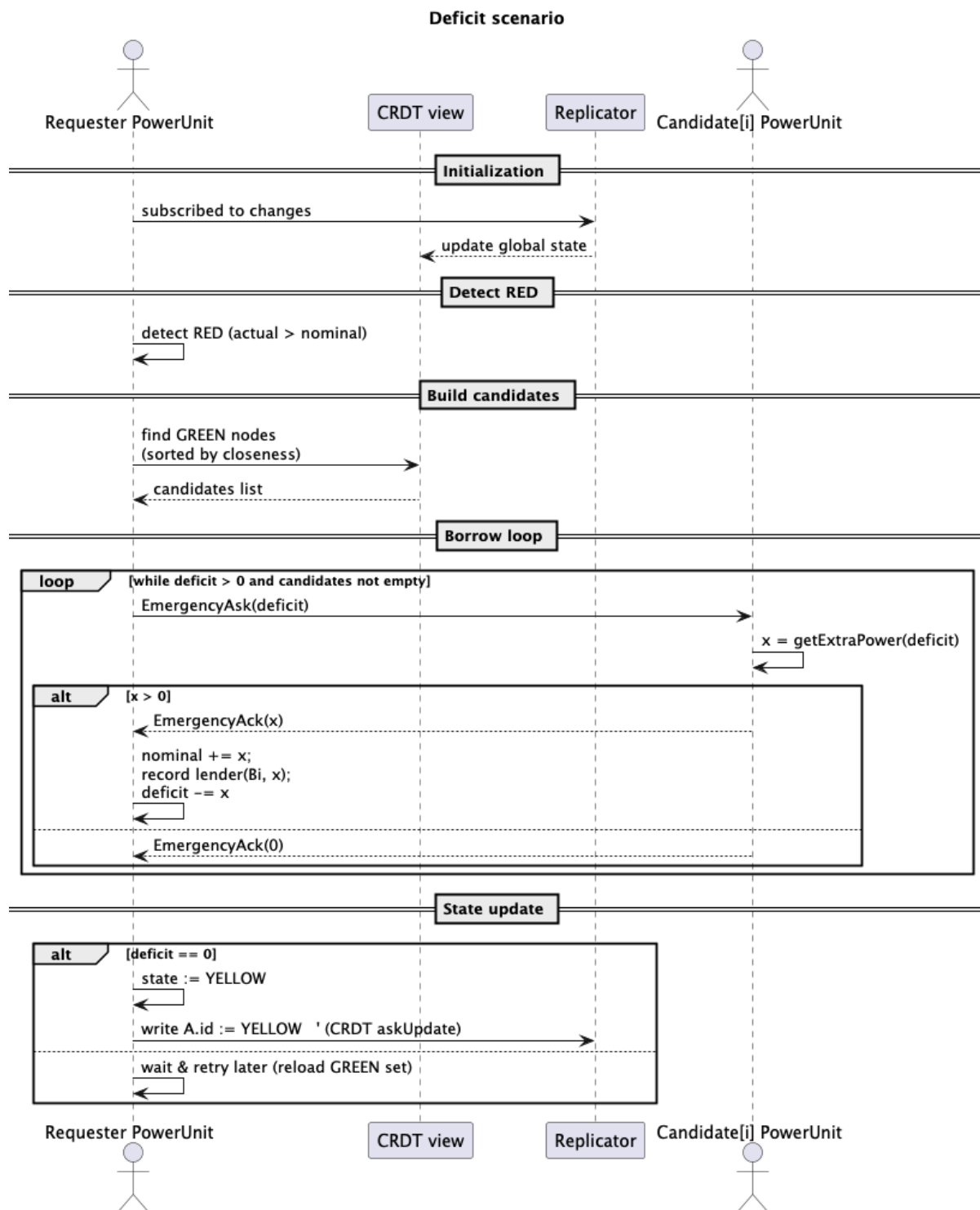


Рисунок 3.2. Діаграма послідовності сценарію дефіциту потужності

Відповідь на такий запит може мати три варіанти: надання всієї необхідної потужності, часткове задоволення потреби або відмова у передачі ресурсу (залежно від фактичного стану агента-донора). У випадку отримання потужності агент фіксує, у кого і в якому обсязі він здійснив запозичення, щоб у майбутньому повернути ресурс. Якщо запозиченої потужності виявляється недостатньо, агент повторює запити до наступних кандидатів, дотримуючись пріоритету географічної близькості. Після отримання необхідного обсягу ресурсу стан агента змінюється з *red* на *yellow*.

На Рисунку 3.2 зображена діаграма послідовності даного сценарію. На ній **Requester PowerUnit** — агент, який за сценарієм потребує додаткової потужності. **Candidate[i] PowerUnit** — і-й елемент списку агентів, до якого надсилається запит на запозичення потужності. Цей список формується на основі CRDT-подання, отриманого від **Replicator**, і відображає уявлення агента про глобальний стан системи на певний момент часу.

Сценарій надлишку. Коли у агента з'являється надлишкова потужність переході зі станів *red* або з *yellow* у стан *green*, він перевіряє наявність раніше отриманих запозичень. У разі їх існування агент повертає потужність, дотримуючись принципу географічного пріоритету, причому спершу здійснюється повернення найбільш віддаленим агентам. Це забезпечує справедливий розподіл ресурсів і підтримання балансу у мережі (Рисунок 3.3).

Сценарій виходу з ладу агента. У разі надання потужності агент-донор створює запис про запозичення (ідентифікатор позичальника та обсяг переданої потужності) і відстежує його життєздатність. Якщо позичальник виходить з ладу, ініціюється процедура повернення: усі раніше передані обсяги потужності повертаються первинним власникам відповідно до зафіксованих записів. Позичальник, у свою чергу, фіксує, у кого і скільки запозичив; тому в разі виходу з ладу донора його потужність зменшується на відповідну величину (Рисунок 3.4).

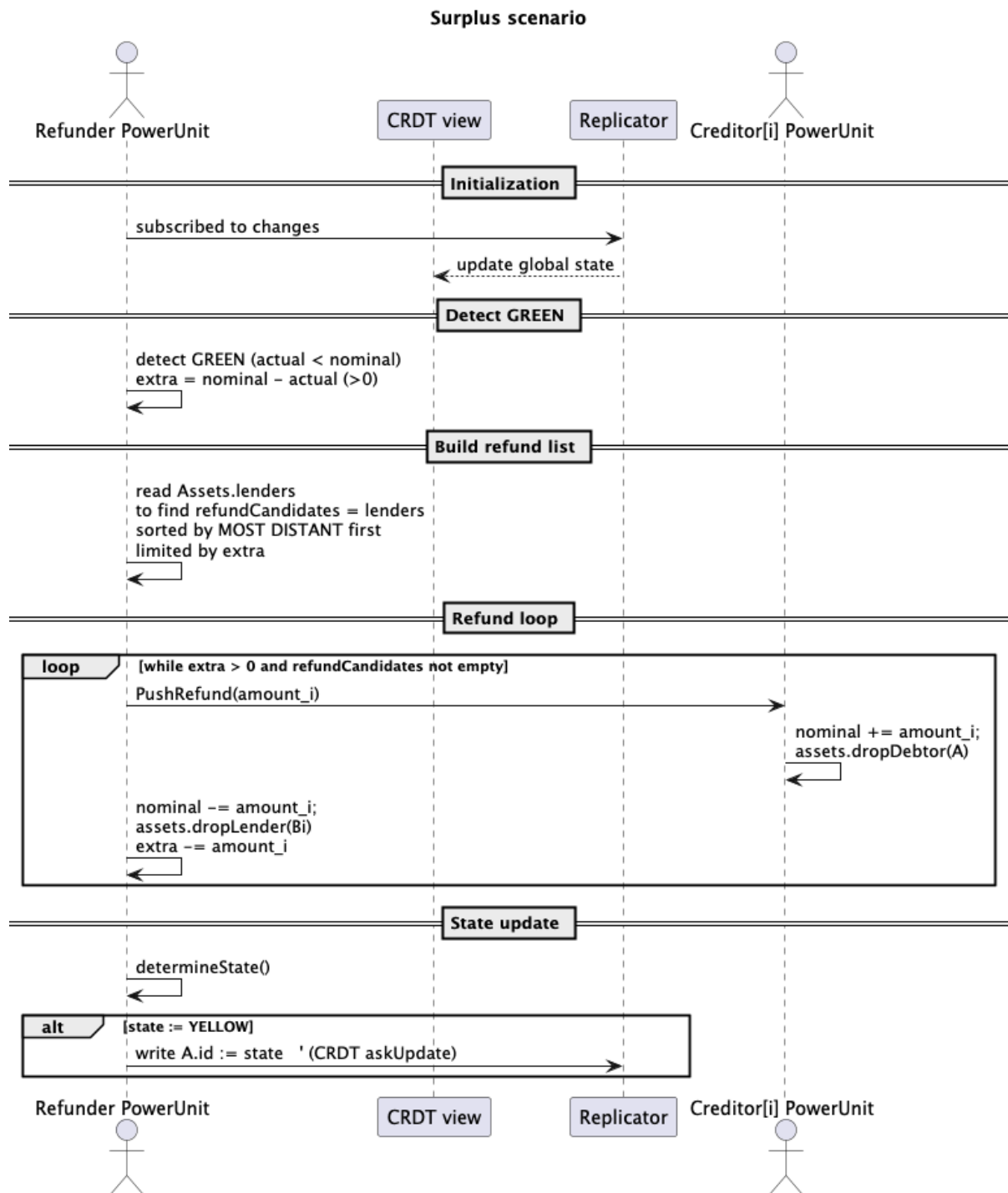


Рисунок 3.3. Діаграма послідовності сценарію надлишку потужності

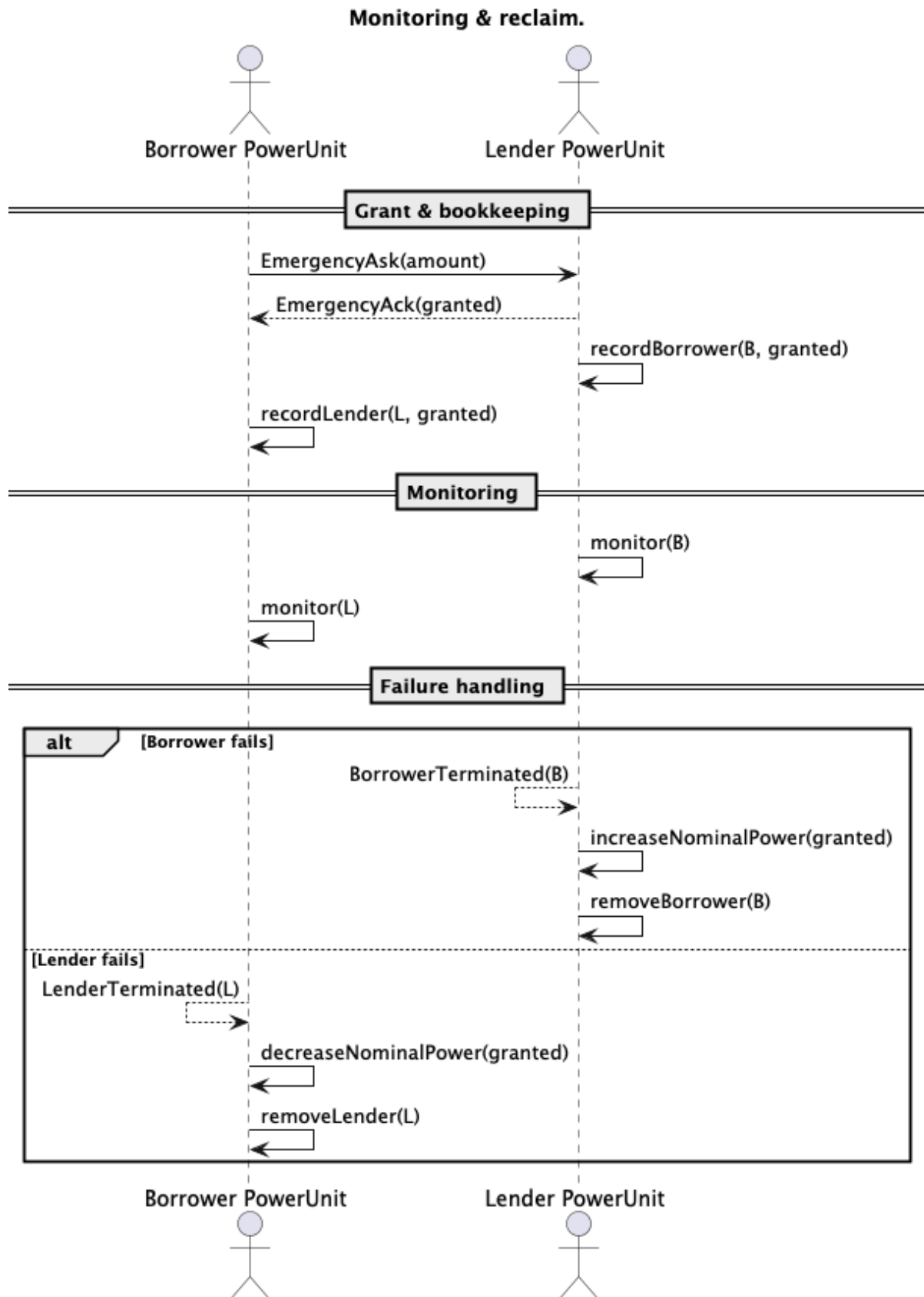


Рисунок 3.4. Діаграма послідовності сценарію виходу з ладу агента

3.3. Імплементація симуляційної моделі

Агент імплементований у вигляді актора **PowerUnit**, що включає в себе як статичну інформацію таку як унікальний ідентифікатор агента та його географічне розташування, так і динамічні характеристики про свій стан: значення фактичної та номінальної потужностей, локальне бачення станів інших агентів, а також інформацію про операції запозичення потужності (як наданої ним, так і отриманої від інших агентів). Всі динамічні характеристики агента були інкапсульовані в клас **PowerUnitState** згідно діаграми класів зображеної на Рисунок 3.5.

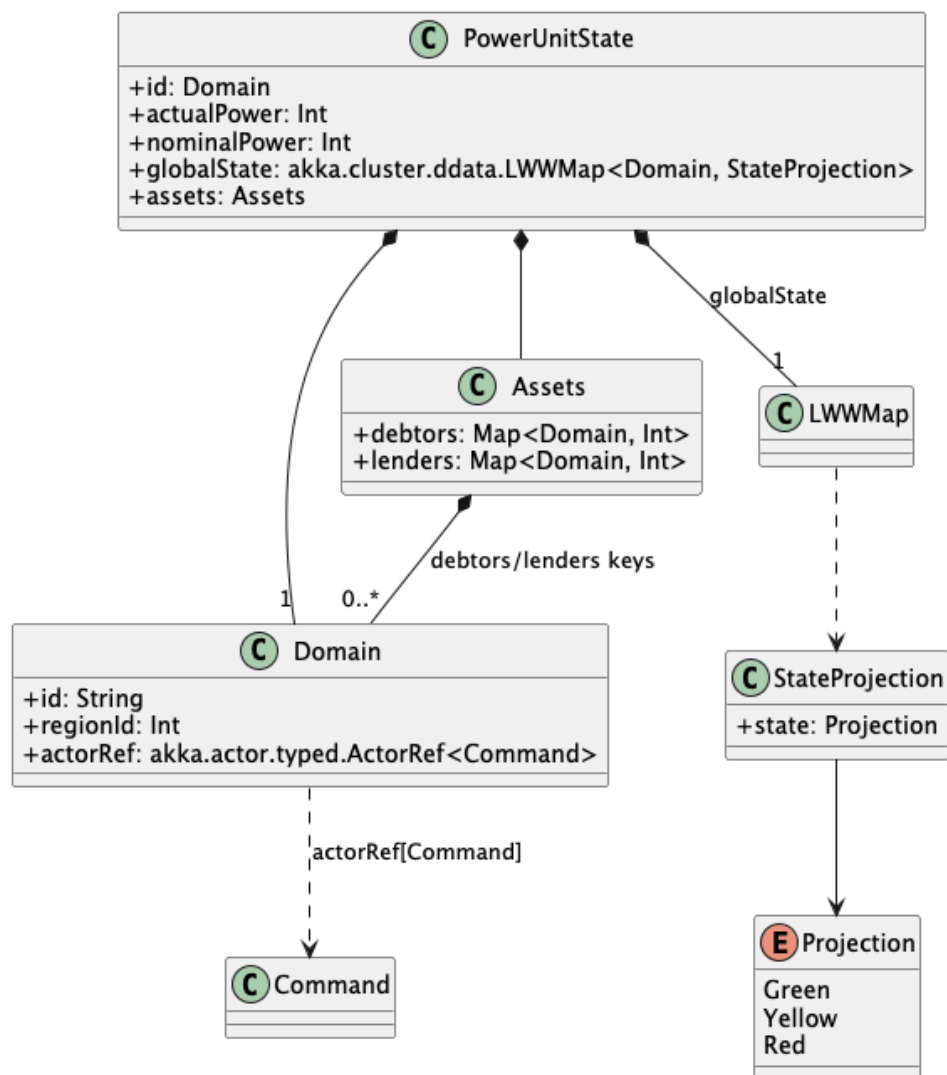


Рисунок 3.5. Діаграма класів **PowerUnitState**

Де: **Domain** містить інформацію про унікальний ідентифікатор, географічний регіон та посилання на актора який представляє собою відповідного агента.

Assets - містить два асоціативних масиви, що використовуються для зберігання даних щодо запозичень в інших агентів.

StateProjection - перелічувальний тип, що має одне з трьох значень: *Green, Yellow, Red* і представляє собою поточний стан агента.

LWWMap - асоціативний масив який являється CRDT типом даних і використовується для отримання і зберігання інформації про стан системи в цілому.

Command - інтерфейс що задає множину можливих повідомлень отриманих актором.

Поведінку агента змодельовано як кінцевий автомат із двома станами: **звичайний стан (regular behaviour)**, що відповідає відсутності дефіциту потужності, та **аварійний стан (emergence behaviour)**, що характеризується дефіцитом потужності або наявністю запозичених ресурсів, які забезпечують його працездатність. Зміна режиму роботи агента відбувається при зміні його стану, представлено за допомогою класу **PowerUnitState**. Зауважимо, що цю поведінку можна також подати у вигляді орієнтованого графа станів (із вагами/ймовірностями переходів чи контекстними умовами), що є доцільним для складніших політик. У такому поданні природно моделюються паралельні гілки виконання та конкурентні переходи [108]. Водночас у межах цієї роботи обрано двостановий автомат як мінімально достатню форму, що спрощує верифікацію та інтерпретацію.

На рисунку 3.6. наведена діаграма класів, що демонструє структуру **PowerUnit**, її зв'язок з **PowerUnitState** та реплікатором даних:

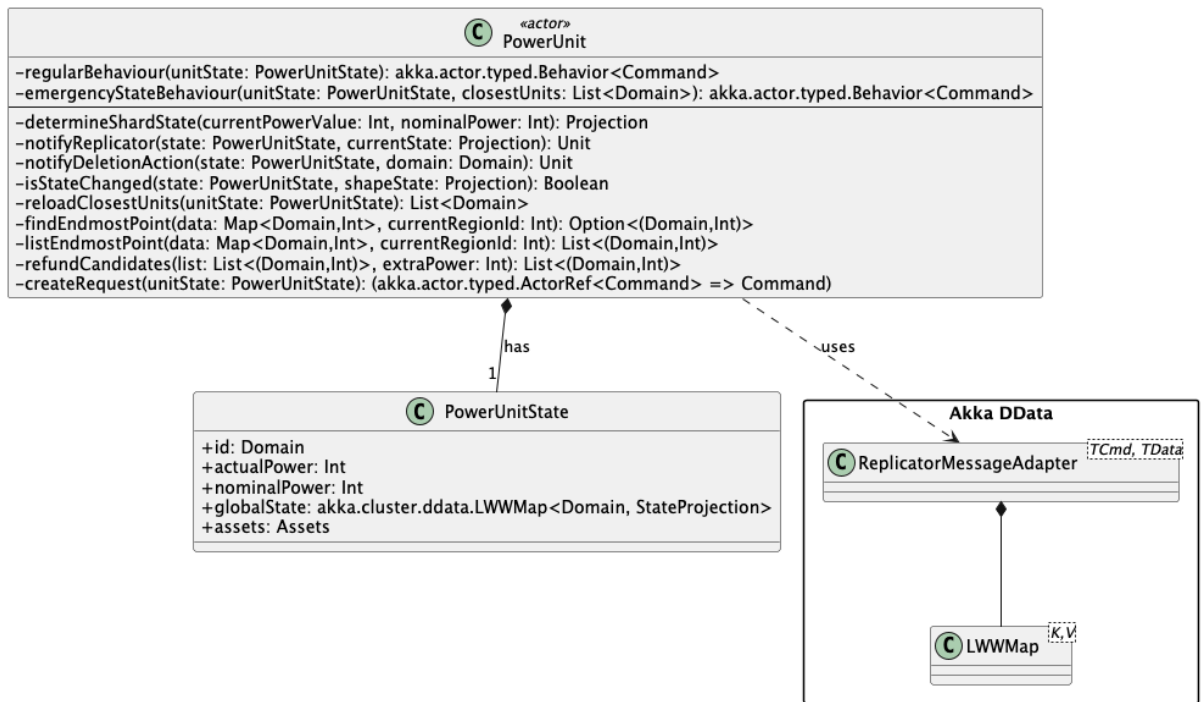


Рисунок 3.6. Діаграма класів PowerUnit

На Рисунку 3.6 зображено **PowerUnit** який включає методи перемикання режимів роботи та методи для забезпечення визначеної логіки режимів.

Akka DData модуль відповідальний за роботу з розподіленими даними. Він включає **ReplicatorMessageAdapter** який надає змогу отримувати **LWWMap** при підписці.

Вся комунікація між агентами та реплікатором здійснюється винятково за допомогою асинхронних повідомлень.

Окрім цього в системі на стороні кожного агента присутній клас **MetricManager** який відповідальний за зняття метрик, їх агрегування та запис в csv файли.

Для зовнішнього збурення агенту розроблений **EmbeddedClusterManagement** клас, відповідальністю якого є відкриття кінцевої точки HTTP, що надає можливість вносити зміни в значення актуальної потужності агента.

Симуляційна модель реалізована з урахуванням двох сценаріїв виконання: локального одно процесного та розподіленого кластерного. Такий підхід дає

змогу використовувати єдиний кодовий базис для налагодження, відтворюваних експериментів і масштабованих обчислень.

Інженерну інфраструктуру симулятора доповнено практиками керування сумісністю компонентів, що узгоджується з підходом оркестрації версій, продемонстрованим у [106], і мінімізує ризики регресій під час еволюції сервісів/акторів.

Усі вихідні коди проєкту доступні у відкритому доступі в репозиторії GitHub [109].

Висновки розділу 3

У розділі III дисертаційної роботи було розглянуто та обґрунтовано вибір інструментів програмної реалізації агентної моделі, а також здійснено безпосередню імплементацію симуляційної моделі високоінтелектуальної енергетичної мережі. Зокрема, було визначено, що використання мови програмування Scala забезпечує поєднання об'єктно-орієнтованої та функціональної парадигм, що є важливим для моделювання складних систем з асинхронними потоками даних. Її інтеграція з екосистемою JVM гарантує продуктивність і сумісність із перевіреними індустріальними бібліотеками, що підвищує практичну значущість розробленого програмного забезпечення.

Було розглянуто фреймворк Akka, який реалізує акторну модель обчислень та забезпечує підтримку конкурентності, масштабованості й відмовостійкості. Показано доцільність використання фреймворку Akka для ефективного моделювання сценаріїв роботи BIEM.

Імплементовано симуляційну модель, що включає доменний опис агентів та механізми синхронізації станів через CRDT. Розроблена модель підтримує відтворюваність експериментів, гнучке налаштування сценаріїв та інтеграцію з моніторинговими інструментами, що створює умови для її застосування як у дослідницьких, так і в практичних завданнях.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА МОДЕЛІ СИНХРОНІЗАЦІЇ СТАНІВ У РОЗПОДІЛЕНИХ ЕНЕРГЕТИЧНИХ СИСТЕМАХ

Експериментальна перевірка оцінює, чи забезпечує запропонована агентна модель на основі CRDT збіжність станів, стійкість до збоїв і ефективну синхронізацію у динамічному середовищі з нерівномірним навантаженням та стохастичними збуреннями.

4.1. Постановка експериментальних досліджень

Мета експериментів — продемонструвати працездатність і ефективність моделі у типових сценаріях роботи високоінтелектуальної енергетичної мережі: підтвердити збіжність локальних представлень до узгодженого глобального стану, зберігання стабільності під збуреннями, а також відмовостійкість і масштабованість під час зміни топології та навантаження.

Набір показників продуктивності, які дозволяють кількісно оцінити роботу моделі:

- узгодженість станів — здатність системи забезпечувати однаковий глобальний стан у всіх вузлах після завершення процесу синхронізації;
- продуктивність — швидкість реакції системи на збурення;
- затримки (англ.: latency) — час поширення повідомлень між вузлами та час досягнення узгодженості;
- масштабованість — зміна характеристик роботи системи при зростанні кількості вузлів і навантаження;

Постановка експериментів дозволяє комплексно перевірити можливості запропонованої моделі в умовах, наближених до реальних сценаріїв роботи високоінтелектуальної енергетичної мережі.

4.2. Методика проведення експериментів

Методика експериментів передбачала побудову симульованого кластера з множиною агентів, кожен з яких моделював окрему енергетичну одиницю з власними статичними та динамічними характеристиками. Для відтворення реалістичних сценаріїв було задано початковий збалансований стан системи, після чого вносилися зовнішні збурення, що спричиняли перехід окремих вузлів у критичні стани. Подальше спостереження за процесом синхронізації дозволяло визначити швидкість та якість відновлення глобальної узгодженості.

Увага приділялася параметрам масштабованості та впливу коефіцієнта деградації затримки (англ.: Latency Degradation Coefficient, далі - LDK) на час поширення станів (англ.: Full State Distribution Time, далі - FSDT). Отримані результати аналізувалися як кількісно, через вимірювання часових характеристик синхронізації, так і якісно, через спостереження за динамікою переходів вузлів між станами.

4.2.1. Вибір сценаріїв моделювання.

Для проведення експериментів було обрано сценарії, що відображають типовий перебіг процесів у ВІЕМ, де порушення балансу може виникати локально, але його наслідки поширюються на всю систему. Основою для моделювання став кластер зі 100 вузлів, кожен з яких відображає енергетичну одиницю з власним станом та правилами переходів. Початково всі вузли перебували у “green” стані, що символізує баланс між номінальною та фактичною потужністю або наявність надлишкового ресурсу.

Далі вводилося зовнішнє збурення, внаслідок якого один або декілька агентів переходили у “red” стан, тобто фактична потужність зменшувалася відносно номінальної, що створювало критичний дефіцит. Це збурення запускало каскадний процес балансування в системі: агенти в “red” стані почали запозичувати потужність у сусідів і переходити в “yellow” стан (частковий баланс), тоді як “green” вузли також змінювали свій стан на «жовтий» після надання допомоги іншим агентам. Таким чином, спостерігався

процес розподілу ресурсу між елементами системи для досягнення нового рівня глобальної рівноваги.

Особливістю сценарію є те, що спостереження здійснювалося з перспективи одного випадково обраного агента, який отримував інформацію про власний стан та дельти змін, що надходили від інших вузлів. Це дозволяє моделювати локальне бачення агента в умовах обмеженої інформації та оцінювати, наскільки швидко і коректно формується глобальне уявлення про стан мережі внаслідок обміну повідомленнями.

На рисунку 4.1 зображена схема перебігу експерименту з точки зору внутрішньої комунікації агентів: *observed* — агент-спостерігач; **D** — фактичне збурення (різниця між фактичною та номінальною потужністю вузла); ΔS — дельта стану, що поширюється до найближчих агентів (**No**); **Dk** — коефіцієнт глибини (кількість шарів агентів відносно спостерігача); **R** — результуючий стан; $f(\mathbf{D}, \mathbf{O})$ — процес розповсюдження CRDT-повідомлень між агентами; **NoO** — режим, у якому кожен агент **No** стає спостерігачем; **M** — функція злиття (англ.: merge function), що приймає ΔS та відповідний **R** для оновлення внутрішнього стану та формування проміжного результату.

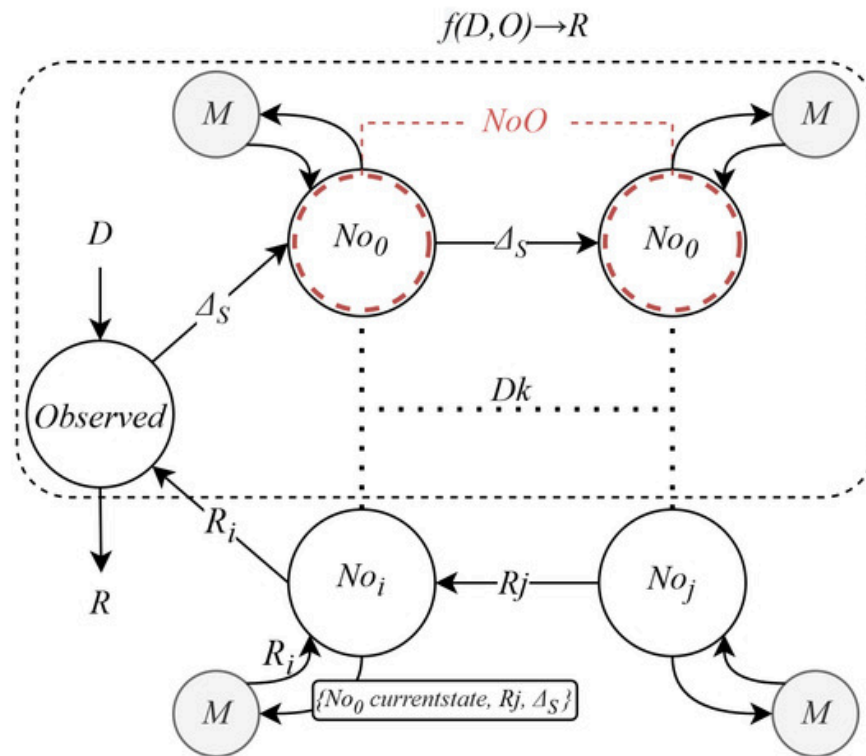


Рисунок 4.1. Схема внутрішньої комунікації агентів

Слід зазначити, що кожен агент реалізує функцію злиття між власним поточним станом, ΔS та вхідним результатом; отже, можливі колізії між ΔS і R . Проте, як було показано у попередніх розділах цієї роботи, можна вважати, що всі конфлікти будуть розв'язані в процесі розповсюдження CRDT-повідомлень. Таким чином, вибраний сценарій дозволяє дослідити ключові властивості системи:

- швидкість відновлення балансу після локальних збурень
- ефективність CRDT-підходу для забезпечення збіжності станів у динамічному середовищі.

4.2.2. Параметри симуляцій: кількість агентів, затримки мережі.

Параметри симуляційного середовища визначають складність, достовірність і відтворюваність експериментів, а також формують основу для подальшого аналізу масштабованості та стійкості розподіленої енергетичної системи.

Кількість агентів. У моделі було симульовано кластер зі 100 агентів, що дозволяє відтворити характерні процеси у розподіленій мережі. Така кількість забезпечує достатню складність для виникнення характерних для ефектів ВІЕМ (асинхронність оновлень, неоднорідність розподілу навантаження, локальні кризи), але водночас не перевантажує симуляційне середовище, зберігаючи можливість детального спостереження за динамікою переходів. Кожен агент визначався унікальним ідентифікатором, регіональною приналежністю та набором характеристик (номінальна й фактична потужність, множини взаємодіючих вузлів). Це дозволяло моделювати як локальні особливості вузлів, так і глобальні механізми балансування ресурсів.

Затримки мережі. У реальних розподілених системах комунікаційні канали піддаються впливу факторів, що викликають затримки: перевантаження, тимчасові відмови, переривчасті з'єднання. Для імітації таких умов у моделі було введено параметр LDK, що визначає штучне збільшення затримки під час обміну повідомленнями між вузлами.

Вибір значень LDK дозволив відтворювати різні режими роботи системи:

- низькі значення — умови стабільної мережі з мінімальними затримками;
- середні значення — сценарії з нерівномірним навантаженням і локальними комунікаційними затримками;
- високі значення — граничні умови з деградацією швидкості розповсюдження станів і ризиком втрати синхронізації.

Поєднання параметрів. Кількість вузлів і моделювання затримок є взаємопов'язаними параметрами. Зі збільшенням кількості агентів відчутнішими стають комунікаційні обмеження, що дозволяє відтворити реальні умови роботи високоінтелектуальної енергетичної мережі, де кількість учасників системи постійно зростає. Варіювання LDK у поєднанні зі 100 вузлами дає змогу оцінити, наскільки ефективно система підтримує глобальну узгодженість у випадку зростання масштабів та ускладнення умов функціонування.

Таблиця 4.1. Основні параметри симуляційного середовища

| Параметр | Значення / Діапазон | Призначення |
|-----------------------------------|----------------------------------|---|
| Кількість агентів | 100 | Баланс між складністю та керованістю експерименту |
| Ідентифікатор вузла (ID) | Унікальний для кожного агента | Забезпечує адресність у процесі обміну повідомленнями |
| Регіональна приналежність (RI) | Випадковий розподіл між вузлами | Моделює географічну неоднорідність |
| Номінальна та фактична потужність | Встановлюються для кожного вузла | Визначають його локальний стан |

| Параметр | Значення / Діапазон | Призначення |
|--------------------------------|--|--|
| Коефіцієнт деградації затримки | Низькі (1–2), середні (2–4), високі (5+) | Моделювання умов від стабільної мережі до сценаріїв із високими затримками |
| Тип взаємодії | Асинхронний обмін повідомленнями | Відтворює характер роботи BIEM |

4.2.3. Методи збору та аналізу результатів.

Для оцінки ефективності запропонованої моделі синхронізації станів у BIEM було визначено низку показників і методів обробки експериментальних даних.

Збір даних. У ході симуляційного експерименту кожен агент генерував журнали подій із часовими відмітками, у яких фіксувалися: локальні характеристики номінальної та фактичної потужності, поточний стан вузла, а також бачення цим вузлом цілісного стану мережі.

Зібрані дані агрегувалися у центральному модулі моніторингу, що дозволяло формувати узагальнене уявлення про перебіг процесів у мережі.

Метрики аналізу. Основними параметрами, що підлягали вимірюванню, були:

- час повного поширення стану — інтервал, необхідний для досягнення узгодженості після локального збурення;
- кількість переходів станів між різними рівнями (green → yellow → red), що характеризує стабільність та швидкість балансування;
- затримка повідомлень (англ.: message latency), яка дозволяє оцінити вплив параметра LDK на роботу системи.

Методи обробки та візуалізації. Зібрані дані оброблялися статистичними методами з використанням агрегування та обчислення середніх значень. Для порівняння ефективності різних параметрів симуляцій використовувалося графічне подання результатів у вигляді графіків (за допомогою Excel та python скрипта - Додаток Б) та залежностей FSDT від вхідних змінних. Візуалізація

даних дозволяла наочно продемонструвати динаміку процесів синхронізації та ефективність CRDT-підходу.

Узагальнення результатів. Застосовані методи збору та аналізу даних дали змогу формалізовано оцінити ефективність запропонованої моделі, зіставити поведінку системи за різних сценаріїв і параметрів, а також виявити ключові чинники, що впливають на швидкість і стабільність досягнення глобальної узгодженості.

4.3. Результати експериментальних досліджень

Проведені симуляції засвідчили працездатність запропонованої агентної моделі з використанням CRDT у різних сценаріях функціонування високоінтелектуальної енергетичної мережі. Основна увага приділялася здатності системи забезпечувати збіжність станів у глобальному масштабі, стійкість до локальних збурень та ефективність синхронізації за умов варіативних параметрів затримок.

Експериментальне дослідження було організовано у кілька фаз із використанням різних часових інтервалів, що дозволяло змодельовати поступове зростання мережевих затримок. Симуляції виконувалися на апаратній платформі з процесором Apple M3 Pro, 36 ГБ оперативної пам'яті та SSD-накопичувачем обсягом 512 ГБ. Сукупність проведених експериментів охоплювала варіації параметрів деградації затримок та щільності розподілу повідомлень.

Для моделювання впливу відстані між вузлами було введено коефіцієнт деградації затримки, що виконував роль динамічного множника та забезпечував урахування додаткових затримок у доставці повідомлень, роблячи середовище більш наближеним до реальних, неідеальних умов.

FSDT визначався як тривалість, необхідна для розповсюдження зміни стану одного вузла до 100% інших вузлів. Дана метрика застосовувалася для якісної оцінки впливу затримок на швидкість поширення станів у системі.

Для візуалізації процесів поширення повідомлень було використано перспективу незбурених вузлів, що дозволило оцінити динаміку інформаційних потоків у мережі. У дослідженні розглядалася повнозв'язна однорангова топологія; параметр LDK для всіх каналів було уніфіковано.

Використання детермінованого або стано-орієнтованого ініціалізування, неблокуючого моніторингу та інкапсульованих станів забезпечувало відтворюваність і надійність результатів навіть за різних сценаріїв збурень у мережі, що стало можливим завдяки властивостям CRDT та акторно-орієнтованій реалізації системи.

Результати симуляцій подано на рисунках 4.2 та 4.3, де нульовий момент часу визначається як момент досягнення всіма вузлами початкового стану балансу. Цей баланс характеризується повною синхронізацією станів системи та безперервним відображенням комунікації через передавання дельт змін станів.

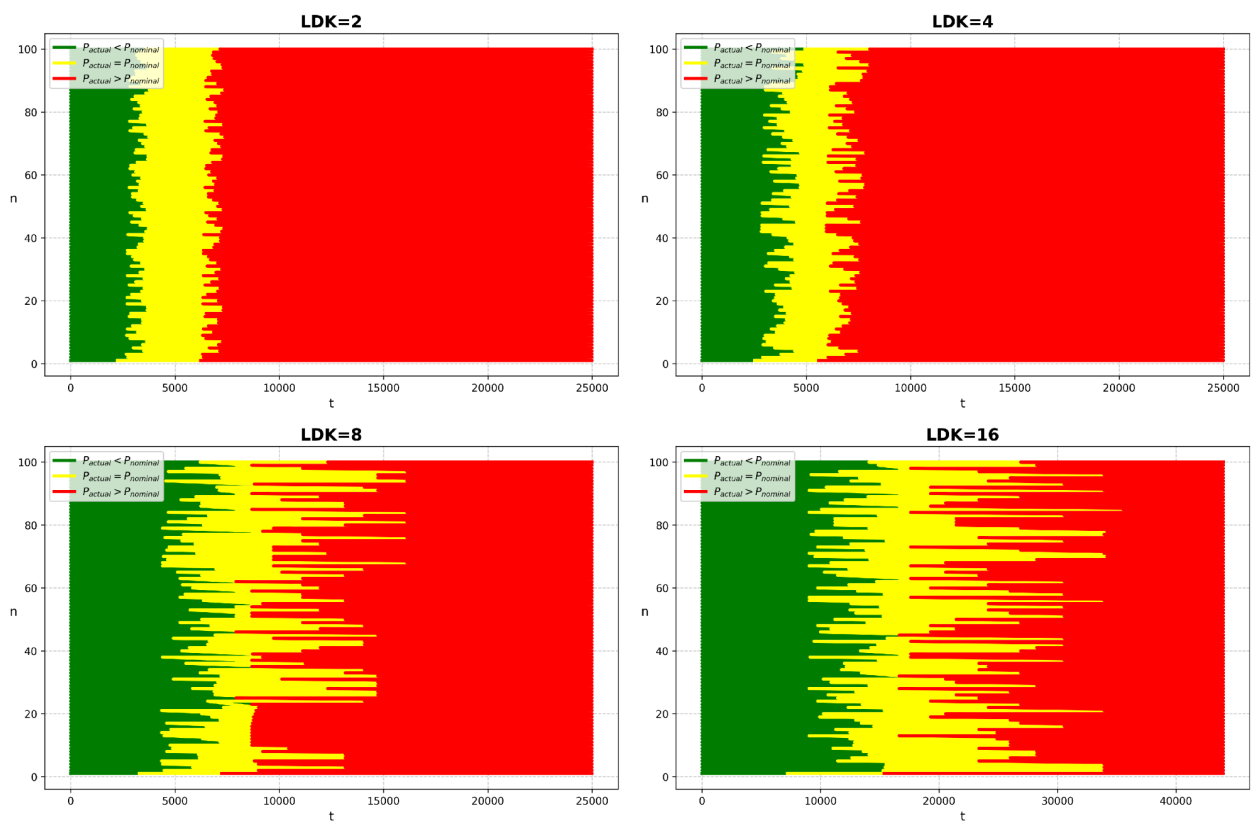


Рисунок 4.2. Результати симуляції щільності розподілу повідомлень при $LDK = 2, 4, 8, 16$.

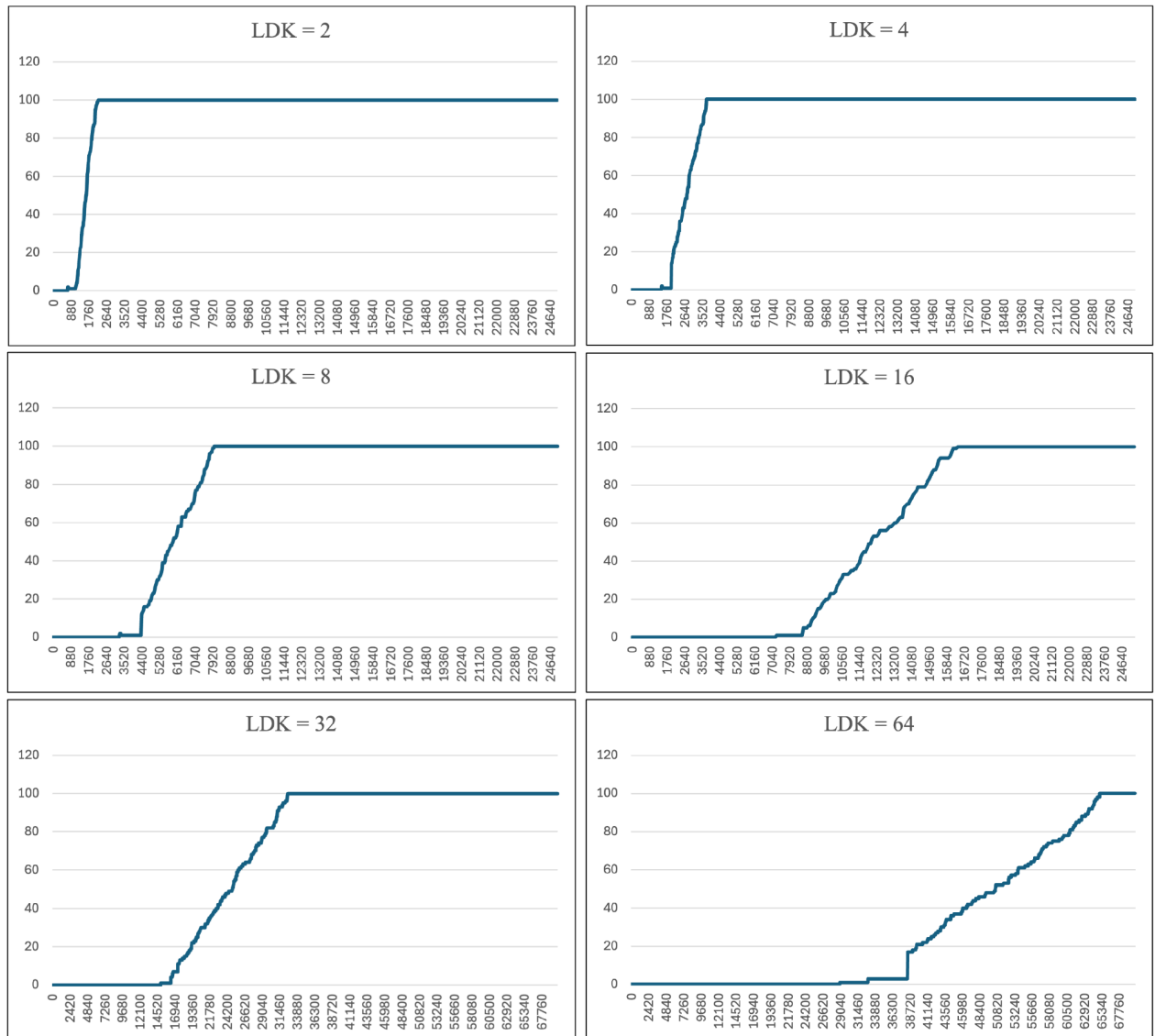


Рисунок 4.2. Результати симуляції FSD при LDK = 2, 4, 8, 16, 32, 64.

На графіках вісь **X (n)** відображає кількість вузлів, що є статичною величиною, **Y (t)** показує час у мілісекундах.

Після внесення збурення на один із агентів здійснюється моніторинг швидкості поширення оновленого стану до інших вузлів. Це реалізується шляхом неблокуючого запису показників стану у окремий файл у форматі: $\{nodeid, P_{actual}, P_{nominal}, state_{current}, time_{current}\}$. Надалі ці дані використовуються для побудови графіків та обчислення відсоткового співвідношення поінформованих вузлів, а також для визначення показника FSDT, оскільки фактична зміна стану має числове вираження, тоді як інтервал між першим та останнім оновленням відображає час поширення стану.

У ході дослідження було встановлено, що збільшення дистанції між агентами з урахуванням деградації затримок може призводити до виникнення турбулентних процесів, що ускладнюють керування системою у різних її аспектах. Разом із тим було ідентифіковано існування ефективного значення коефіцієнта деградації затримки $LDK_{\text{effective}} = \{2, 4\}$, за якого мережа функціонує без істотних збурень. Для досягнення оптимізації параметра $LDK_{\text{effective}}$ доцільним виявляється поділ кластера на менші підкластерні утворення. Очевидно, що реалізація такого підходу потребує визначеної топології та спеціалізованого протоколу міжкластерної взаємодії з метою забезпечення цілісної зв'язності в межах більших мережевих об'єднань. Водночас варто підкреслити, що за умов функціонування із зазначеним $LDK_{\text{effective}}$ система демонструвала плавну, стабільну та передбачувану динаміку.

Було встановлено, що зі зростанням коефіцієнта деградації затримки відповідно збільшується час повного поширення стану. Водночас процес розповсюдження повідомлень між іншими вузлами залишається стабільним і передбачуваним. По суті, спостерігається лінійна залежність між FSDT та LDK, що забезпечує масштабованість запропонованого рішення. Варто також відзначити, що у цьому експерименті було використано підвищене середнє значення LDK з метою продемонструвати ефективність функціонування системи на основі CRDT за умов загальної деградації мережевої зв'язності.

Висновки розділу 4

У розділі 4 дисертаційної роботи було здійснено постановку, реалізацію та аналіз експериментальних досліджень, спрямованих на перевірку ефективності запропонованої агентної моделі високоінтелектуальної енергетичної мережі із застосуванням CRDT. Запропонована методика моделювання, що базувалася на кластері зі 100 агентів з асинхронним обміном повідомленнями, дала змогу відтворити ключові процеси, характерні для BIEM, зокрема локальні збурення, нерівномірний розподіл навантаження та

варіативність мережесих затримок. Використання коефіцієнта деградації затримки та метрики часу повного поширення стану дозволило формалізувати оцінку впливу комунікаційних умов на ефективність синхронізації.

Результати експериментів підтвердили здатність моделі забезпечувати збіжність локальних станів у глобальний, підтримувати стабільність функціонування системи за умов стохастичних збурень та демонструвати відмовостійкість у динамічному середовищі. Встановлено, що зі зростанням LDK пропорційно збільшується FSDT, при цьому сам процес розповсюдження повідомлень серед вузлів залишається передбачуваним та стабільним, що свідчить про масштабованість розробленого рішення. Було також ідентифіковано існування ефективного діапазону значень коефіцієнта деградації затримки $LDK_{\text{effective}} = \{2, 4\}$, за якого система функціонує без істотних збурень, що відкриває можливості оптимізації шляхом поділу великих кластерів на підкластерні групи з використанням спеціалізованих протоколів міжкластерної взаємодії.

Завдяки детермінованому ініціалізуванню, неблокуючому моніторингу та інкапсульованим станам отримано відтворювані й надійні результати, що підтверджують доцільність підходу синхронізації станів агентів на базі CRDT у поєднанні з акторною моделлю для моделювання BIEM. Отримані результати експериментів підтвердили ефективність запропонованої моделі у забезпеченні масштабованості та відмовостійкості під час моделювання BIEM, що дає змогу підтримувати узгоджені стани за умов збурень і вибуття вузлів. Виявлені закономірності поширення станів та впливу мережесих затримок можуть слугувати основою для розроблення оптимальних стратегій керування BIEM в реальних умовах.

ВИСНОВКИ

Дисертаційної робота присвячена розробці та верифікації агентної моделі BIEM із використанням CRDT для опису та синхронізації станів системи. Запропонований підхід поєднав теоретичні засади БАС, акторну модель програмування та математичні властивості CRDT, що забезпечують збіжність станів у розподілених середовищах з асинхронною взаємодією. Основні наукові та практичні результати проведених теоретичних та експериментальних досліджень полягають в наступному:

1. Проаналізовано існуючі підходи до синхронізації інформації про стан мультиагентної системи у кожного агента. Встановлено, що застосування мультиагентних систем має низку обмежень. Зокрема, через складність забезпечення узгодженості станів у розподіленому середовищі, так як традиційні методи узгодженості, засновані на централізованих координаторах або блокуючих протоколах, демонструють обмежену ефективність у масштабних і динамічних мережах. Використання неконфліктних структур даних своєю чергою надасть змогу забезпечити детерміновану конвергенцію локальних подань стану без глобальної координації, зменшить потребу в блокуваннях і підвищить стійкість до збоїв. Тому розроблення підходу до агентного моделювання високоінтелектуальних енергетичних мереж із використанням неконфліктних реплікованих структур даних для опису станів системи є актуальним науковим завданням.
2. Розроблено формальну модель представлення станів вузлів високоінтелектуальних енергетичних мережі на основі неконфліктних структур даних у якій енергетичні вузли подано як автономних агентів, здатних до локального прийняття рішень і координації через обмін повідомленнями.
3. Розроблено підхід синхронізації станів агентів мультиагентної системи на основі неконфліктних структур даних, який, на відміну від

централізованих/консенсусних підходів, забезпечує детерміновану збіжність без глобальної координації.

4. Розроблено комплекс програмного забезпечення для агентного моделювання ВІЕМ на основі акторної моделі, який забезпечує автоматизацію прийняття рішень щодо балансування, верифікацію децентралізованих алгоритмів керування та відтворюване оцінювання стійкості до мережових затримок і відмов, у якому агенти ухвалюють локальні рішення та координуються обміном повідомлень із застосуванням неконфліктних реплікованих структур даних.
5. Експериментально підтверджено ефективність запропонованого підходу за допомогою розробленого програмного забезпечення. Показано, що система забезпечує збіжність локальних станів у глобальний, стійкість до збоїв і відмовостійкість навіть у випадку підвищених комунікаційних затримок.
6. Виявлено лінійну залежність між повним поширення стану і коефіцієнтом деградації затримки, що засвідчує передбачувану масштабованість рішення.
7. Визначено оптимальні параметри роботи розробленої системи в умовах різних мережових затримок. Ідентифіковано ефективний діапазон параметрів коефіцієнта деградації затримки, за якого мережа функціонує без суттєвих збурень.
8. Розроблене програмне забезпечення придатне до практичного застосування як інженерний інструмент для аналізу, проектування та оптимізації моделей високоінтелектуальних енергетичних мереж. Крім того, розроблене ПЗ може бути використано при розробленні стратегій керування такими мережами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Y. Izmirlioglu, L. Pham, T. C. Son, and E. Pontelli, “A Survey of Multi-Agent Systems for Smartgrids,” *Energies*, vol. 17, no. 15, p. 3620, July 2024, doi: 10.3390/en17153620.
- [2] D. Moneta, “Smart grids: enabler for the energy transition,” *EPJ Web Conf.*, vol. 189, p. 00012, 2018, doi: 10.1051/epjconf/201818900012.
- [3] X. Fang, S. Misra, G. Xue, and D. Yang, “Smart Grid — The New and Improved Power Grid: A Survey,” *IEEE Commun. Surv. Tutor.*, vol. 14, no. 4, pp. 944–980, 2012, doi: 10.1109/SURV.2011.101911.00087.
- [4] M. Wooldridge and N. R. Jennings, “Intelligent agents: theory and practice,” *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, June 1995, doi: 10.1017/S0269888900008122.
- [5] P. Shi, Y. Cui, K. Xu, M. Zhang, and L. Ding, “Data Consistency Theory and Case Study for Scientific Big Data,” *Information*, vol. 10, no. 4, p. 137, Apr. 2019, doi: 10.3390/info10040137.
- [6] M. Kleppmann, “A Critique of the CAP Theorem,” 2015, *arXiv*. doi: 10.48550/ARXIV.1509.05393.
- [7] A. Prymushko, I. Puchko, M. Yaroshynskyi, D. Sinko, H. Kravtsov, and V. Artemchuk, “Efficient State Synchronization in Distributed Electrical Grid Systems Using Conflict-Free Replicated Data Types,” *IoT*, vol. 6, no. 1, p. 6, Jan. 2025, doi: 10.3390/iot6010006.
- [8] A. Joseph and P. Balachandra, “Smart Grid to Energy Internet: A Systematic Review of Transitioning Electricity Systems,” *IEEE Access*, vol. 8, pp. 215787–215805, 2020, doi: 10.1109/ACCESS.2020.3041031.
- [9] S. Howell, Y. Rezgui, J.-L. Hippolyte, B. Jayan, and H. Li, “Towards the next generation of smart grids: Semantic and holonic multi-agent management of distributed energy resources,” *Renew. Sustain. Energy Rev.*, vol. 77, pp. 193–214, Sept. 2017, doi: 10.1016/j.rser.2017.03.107.

- [10] A. L. Kulasekera, R. A. R. C. Gopura, K. T. M. U. Hemapala, and N. Perera, “A review on multi-agent systems in microgrid applications,” in *ISGT2011-India*, Kollam, Kerala, India: IEEE, Dec. 2011, pp. 173–177. doi: 10.1109/ISET-India.2011.6145377.
- [11] “Comprehensive overview of multi-agent systems for controlling smart grids,” *CSEE J. Power Energy Syst.*, 2020, doi: 10.17775/CSEEJPES.2020.03390.
- [12] P. Kiran, K. R. M. V. Chandrakala, and T. N. P. Nambiar, “Multi-agent based systems on micro grid — A review,” in *2017 International Conference on Intelligent Computing and Control (I2C2)*, Coimbatore: IEEE, June 2017, pp. 1–6. doi: 10.1109/I2C2.2017.8321880.
- [13] M. Hasanuzzaman Shawon, S. M. Mueen, A. Ghosh, S. M. Islam, and M. S. Baptista, “Multi-Agent Systems in ICT Enabled Smart Grid: A Status Update on Technology Framework and Applications,” *IEEE Access*, vol. 7, pp. 97959–97973, 2019, doi: 10.1109/ACCESS.2019.2929577.
- [14] M. Panteli and P. Mancarella, “The Grid: Stronger, Bigger, Smarter?: Presenting a Conceptual Framework of Power System Resilience,” *IEEE Power Energy Mag.*, vol. 13, no. 3, pp. 58–66, May 2015, doi: 10.1109/MPE.2015.2397334.
- [15] A. Kantamneni, L. E. Brown, G. Parker, and W. W. Weaver, “Survey of multi-agent systems for microgrid control,” *Eng. Appl. Artif. Intell.*, vol. 45, pp. 192–203, Oct. 2015, doi: 10.1016/j.engappai.2015.07.005.
- [16] T. B. González-Castro and C. A. Tovilla-Zárate, “Meta-analysis: A tool for clinical and experimental research in psychiatry,” *Nord. J. Psychiatry*, vol. 68, no. 4, pp. 243–250, May 2014, doi: 10.3109/08039488.2013.830773.
- [17] M. Kiasari, M. Ghaffari, and H. Aly, “A Comprehensive Review of the Current Status of Smart Grid Technologies for Renewable Energies Integration and Future Trends: The Role of Machine Learning and Energy Storage Systems,” *Energies*, vol. 17, no. 16, p. 4128, Aug. 2024, doi: 10.3390/en17164128.

- [18] Y. Duan, Z. Xu, H. Chen, and Y. Wang, “Novel machine learning approach for enhanced smart grid power use and price prediction using advanced shark Smell-Tuned flexible support vector machine,” *Sci. Rep.*, vol. 15, no. 1, p. 20909, July 2025, doi: 10.1038/s41598-025-05083-0.
- [19] A. Stocker and H. de Meer, “A Tutorial on Resilience in Smart Grids,” in *2022 12th International Workshop on Resilient Networks Design and Modeling (RNDM)*, Sept. 2022, pp. 1–14. doi: 10.1109/RNDM55901.2022.9927711.
- [20] P. Zhang, F. Li, and N. Bhatt, “Next-Generation Monitoring, Analysis, and Control for the Future Smart Control Center,” *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 186–192, Sept. 2010, doi: 10.1109/TSG.2010.2053855.
- [21] F. Blik *et al.*, “PowerMatching City, a living lab smart grid demonstration,” in *2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe)*, Gothenburg, Sweden: IEEE, Oct. 2010, pp. 1–8. doi: 10.1109/ISGTEUROPE.2010.5638863.
- [22] L. Gomes, Z. Vale, and J. M. Corchado, “Microgrid management system based on a multi-agent approach: An office building pilot,” *Measurement*, vol. 154, p. 107427, Mar. 2020, doi: 10.1016/j.measurement.2019.107427.
- [23] N. R. Jennings, “An agent-based approach for building complex software systems,” *Commun. ACM*, vol. 44, no. 4, pp. 35–41, Apr. 2001, doi: 10.1145/367211.367250.
- [24] N. R. Jennings, K. Sycara, and M. Wooldridge, “A Roadmap of Agent Research and Development,” *Auton. Agents Multi-Agent Syst.*, vol. 1, no. 1, pp. 7–38, Mar. 1998, doi: 10.1023/A:1010090405266.
- [25] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. in Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 1995.
- [26] G. Rohbogner, U. Hahnel, P. Benoit, and S. Fey, “Multi-agent systems’ asset for smart grid applications,” *Comput. Sci. Inf. Syst.*, vol. 10, no. 4, pp. 1799–1822, 2013, doi: 10.2298/CSIS130224072R.

- [27] H. Ahmed Abbas, “Organization of Multi-Agent Systems: An Overview,” *Int. J. Intell. Inf. Syst.*, vol. 4, no. 3, p. 46, 2015, doi: 10.11648/j.ijis.20150403.11.
- [28] B. Horling and V. Lesser, “A survey of multi-agent organizational paradigms,” *Knowl. Eng. Rev.*, vol. 19, no. 4, pp. 281–316, Dec. 2004, doi: 10.1017/S0269888905000317.
- [29] M. J. Wooldridge, *An introduction to multiagent systems*, 2nd ed. Chichester, U.K: John Wiley & Sons, 2009.
- [30] S. Karnouskos and T. N. D. Holanda, “Simulation of a Smart Grid City with Software Agents,” in *2009 Third UKSim European Symposium on Computer Modeling and Simulation*, Athens: IEEE, Nov. 2009, pp. 424–429. doi: 10.1109/EMS.2009.53.
- [31] W. Shen and D. H. Norrie, “Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey,” *Knowl. Inf. Syst.*, vol. 1, no. 2, pp. 129–156, May 1999, doi: 10.1007/BF03325096.
- [32] H. N. S. Aldin, H. Deldari, M. H. Moattar, and M. R. Ghods, “Consistency models in distributed systems: A survey on definitions, disciplines, challenges and applications,” 2019, *arXiv*. doi: 10.48550/ARXIV.1902.03305.
- [33] Pradeep Bhosale, “Data Consistency Models in Distributed Systems: CAP Theorem Revisited,” July 2023, doi: 10.5281/ZENODO.14631471.
- [34] S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services,” *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, June 2002, doi: 10.1145/564585.564601.
- [35] P. Viotti and M. Vukolić, “Consistency in Non-Transactional Distributed Storage Systems,” *ACM Comput. Surv.*, vol. 49, no. 1, pp. 1–34, Mar. 2017, doi: 10.1145/2926965.
- [36] Z. Guo *et al.*, “Cornus: atomic commit for a cloud DBMS with storage disaggregation,” *Proc. VLDB Endow.*, vol. 16, no. 2, pp. 379–392, Oct. 2022, doi: 10.14778/3565816.3565837.

- [37] A. Rodriguez and W. Osborn, “Distributed Locking: Performance Analysis and Optimization Strategies,” Apr. 03, 2025, *arXiv*: arXiv:2504.03073. doi: 10.48550/arXiv.2504.03073.
- [38] S. Lungu and M. Nyirenda, “Current Trends in the Management of Distributed Transactions in Micro-Services Architectures: A Systematic Literature Review,” *Open J. Appl. Sci.*, vol. 14, no. 09, pp. 2519–2543, 2024, doi: 10.4236/ojapps.2024.149167.
- [39] Y. Huang, H. Bai, E. Kohler, B. Liskov, and L. Shriram, “The Impact of Timestamp Granularity in Optimistic Concurrency Control,” 2018, *arXiv*. doi: 10.48550/ARXIV.1811.04967.
- [40] J. Wang *et al.*, “Polyjuice: High-Performance Transactions via Learned Concurrency Control,” June 15, 2021, *arXiv*: arXiv:2105.10329. doi: 10.48550/arXiv.2105.10329.
- [41] L. Lamport, “The part-time parliament,” *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, May 1998, doi: 10.1145/279227.279229.
- [42] V. Yodaiken, “Understanding Paxos and other distributed consensus algorithms,” Feb. 13, 2022, *arXiv*: arXiv:2202.06348. doi: 10.48550/arXiv.2202.06348.
- [43] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002, doi: 10.1145/571637.571640.
- [44] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, “HotStuff: BFT Consensus with Linearity and Responsiveness,” in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, Toronto ON Canada: ACM, July 2019, pp. 347–356. doi: 10.1145/3293611.3331591.
- [45] W. Vogels, “Eventually consistent,” *Commun. ACM*, vol. 52, no. 1, pp. 40–44, Jan. 2009, doi: 10.1145/1435417.1435432.
- [46] P. Bailis and A. Ghodsi, “Eventual Consistency Today: Limitations, Extensions, and Beyond: How can applications be built on eventually consistent

infrastructure given no guarantee of safety?,” *Queue*, vol. 11, no. 3, pp. 20–32, Mar. 2013, doi: 10.1145/2460276.2462076.

[47] H.-R. Ouyang, H.-F. Wei, H.-X. Li, A.-Q. Pan, and Y. Huang, “Checking Causal Consistency of MongoDB,” *J. Comput. Sci. Technol.*, vol. 37, no. 1, pp. 128–146, Feb. 2022, doi: 10.1007/s11390-021-1662-8.

[48] M. Perrin, A. Mostefaoui, and C. Jard, “Causal consistency: beyond memory,” in *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Barcelona Spain: ACM, Feb. 2016, pp. 1–12. doi: 10.1145/2851141.2851170.

[49] X. Jiang, H. Wei, and Y. Huang, “Tunable Causal Consistency: Specification and Implementation,” Nov. 07, 2022, *arXiv*: arXiv:2211.03501. doi: 10.48550/arXiv.2211.03501.

[50] R. A. Campêlo, M. A. Casanova, D. O. Guedes, and A. H. F. Laender, “A brief survey on replica consistency in cloud environments,” *J. Internet Serv. Appl.*, vol. 11, no. 1, p. 1, Dec. 2020, doi: 10.1186/s13174-020-0122-y.

[51] M. Kleppmann and H. Howard, “Byzantine Eventual Consistency and the Fundamental Limits of Peer-to-Peer Databases,” Dec. 01, 2020, *arXiv*: arXiv:2012.00472. doi: 10.48550/arXiv.2012.00472.

[52] M. Abdelmalak, V. Venkataramanan, and R. Macwan, “A Survey of Cyber-Physical Power System Modeling Methods for Future Energy Systems,” *IEEE Access*, vol. 10, pp. 99875–99896, 2022, doi: 10.1109/ACCESS.2022.3206830.

[53] I. Mashal, O. A. Khashan, M. Hijjaw, and M. Alshinwan, “The determinants of reliable smart grid from experts’ perspective,” *Energy Inform.*, vol. 6, no. 1, p. 10, Apr. 2023, doi: 10.1186/s42162-023-00266-3.

[54] F. Yuan *et al.*, “The Evolution and Optimization Strategies of a PBFT Consensus Algorithm for Consortium Blockchains,” *Information*, vol. 16, no. 4, p. 268, Mar. 2025, doi: 10.3390/info16040268.

- [55] M. Stübs, “Hierarchical Distributed Consensus for Smart Grids,” 2020, doi: 10.18420/SICHERHEIT2020_15.
- [56] K. Kumar Kondru and S. Rajiakodi, “RaftOptima: An Optimised Raft With Enhanced Fault Tolerance, and Increased Scalability With Low Latency,” *IEEE Access*, vol. 12, pp. 105974–105989, 2024, doi: 10.1109/ACCESS.2024.3435978.
- [57] W. Zhong *et al.*, “Byzantine Fault-Tolerant Consensus Algorithms: A Survey,” *Electronics*, vol. 12, no. 18, p. 3801, Sept. 2023, doi: 10.3390/electronics12183801.
- [58] R. Carareto, M. S. Baptista, and C. Grebogi, “Natural synchronization in power-grids with anti-correlated units,” *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 4, pp. 1035–1046, Apr. 2013, doi: 10.1016/j.cnsns.2012.08.030.
- [59] H. Taher, S. Olmi, and E. Schöll, “Enhancing power grid synchronization and stability through time-delayed feedback control,” *Phys. Rev. E*, vol. 100, no. 6, p. 062306, Dec. 2019, doi: 10.1103/PhysRevE.100.062306.
- [60] L. Cuadra, S. Salcedo-Sanz, J. Del Ser, S. Jiménez-Fernández, and Z. Geem, “A Critical Review of Robustness in Power Grids Using Complex Networks Concepts,” *Energies*, vol. 8, no. 9, pp. 9211–9265, Aug. 2015, doi: 10.3390/en8099211.
- [61] Z. Budrikis, “Stability of power grids,” *Nat. Rev. Phys.*, vol. 4, no. 10, pp. 635–635, Sept. 2022, doi: 10.1038/s42254-022-00526-3.
- [62] A. Barreto, H. Paulino, J. A. Silva, and N. Preguiça, “PS-CRDTs: CRDTs in highly volatile environments,” *Future Gener. Comput. Syst.*, vol. 141, pp. 755–767, Apr. 2023, doi: 10.1016/j.future.2022.12.013.
- [63] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, “A comprehensive study of Convergent and Commutative Replicated Data Types,” 2011, [Online]. Available: <https://inria.hal.science/inria-00555588v1>
- [64] N. Preguiça, C. Baquero, and M. Shapiro, “Conflict-Free Replicated Data Types CRDTs,” in *Encyclopedia of Big Data Technologies*, S. Sakr and A.

Zomaya, Eds., Cham: Springer International Publishing, 2018, pp. 1–10. doi: 10.1007/978-3-319-63962-8_185-1.

[65] L. F. Galesky, L. A. Rodrigues, E. P. Duarte Jr., and L. Arantes, “Efficient Synchronization of CRDTs using VCube-PS,” in *12th Latin-American Symposium on Dependable and Secure Computing*, La Paz Bolivia: ACM, Oct. 2023, pp. 50–59. doi: 10.1145/3615366.3615421.

[66] L. D. F. Galesky and L. A. Rodrigues, “Efficient CRDT Synchronization at Scale using a Causal Multicast over a Virtual Hypercube Overlay,” in *Proceedings of the 11th Latin-American Symposium on Dependable Computing*, Fortaleza/CE Brazil: ACM, Nov. 2022, pp. 84–88. doi: 10.1145/3569902.3569948.

[67] M. Simić, M. Stojkov, G. Sladic, and B. Milosavljević, *CRDTs as replication strategy in large-scale edge distributed system: An overview*. 2020.

[68] Y. Zhang, J. K. Wang, and Y. J. Han, “CChain: a high throughput blockchain system,” in *Second International Conference on Digital Society and Intelligent Systems (DSInS 2022)*, J. Hu and X. Yang, Eds., Chendgu, China: SPIE, Apr. 2023, p. 5. doi: 10.1117/12.2673351.

[69] Q. Acher, C.-L. Ignat, and S. Ibrahim, “Quantifying the Performance of Conflict-free Replicated Data Types in InterPlanetary File System,” in *Proceedings of the 4th International Workshop on Distributed Infrastructure for the Common Good*, Bologna Italy: ACM, Dec. 2023, pp. 19–24. doi: 10.1145/3631310.3633488.

[70] W. Cai, F. He, and X. Lv, “Multi-core accelerated CRDT for large-scale and dynamic collaboration,” *J. Supercomput.*, vol. 78, no. 8, pp. 10799–10828, May 2022, doi: 10.1007/s11227-022-04308-7.

[71] V. B. F. Gomes, M. Kleppmann, D. P. Mulligan, and A. R. Beresford, “Verifying strong eventual consistency in distributed systems,” *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, pp. 1–28, Oct. 2017, doi: 10.1145/3133933.

- [72] A. Ahuja, G. Gupta, and S. Sidhanta, “Edge Applications: Just Right Consistency,” in *2019 38th Symposium on Reliable Distributed Systems (SRDS)*, Lyon, France: IEEE, Oct. 2019, pp. 351–3512. doi: 10.1109/SRDS47363.2019.00047.
- [73] J. Bauwens and E. Gonzalez Boix, “From causality to stability: understanding and reducing meta-data in CRDTs,” in *Proceedings of the 17th International Conference on Managed Programming Languages and Runtimes*, Virtual UK: ACM, Nov. 2020, pp. 3–14. doi: 10.1145/3426182.3426183.
- [74] L. F. R. Almas, “Replicated Data Types for Graph Databases,” masterThesis, 2023. Accessed: Aug. 25, 2025. [Online]. Available: <https://run.unl.pt/handle/10362/163513>
- [75] P. Mondal and E. Tilevich, “Undoing CRDT Operations Automatically,” in *2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Naples, Italy: IEEE, Dec. 2023, pp. 246–251. doi: 10.1109/CloudCom59040.2023.00047.
- [76] G. A. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott, “A foundation for actor computation,” *J. Funct. Program.*, vol. 7, no. 1, pp. 1–72, Jan. 1997, doi: 10.1017/S095679689700261X.
- [77] G.E. Pukhov Institute for Modelling in Energy Engineering National Academy of Sciences of Ukraine, Kyiv *et al.*, “Software Design of a Distributed High-Load Power Grid System Based on the Actor Model with the Use of Smart Contracts,” *Elektron. Model.*, vol. 46, no. 3, pp. 57–72, June 2024, doi: 10.15407/emodel.46.03.057.
- [78] C. Camilleri, J. G. Vella, and V. Nezval, “Horizontally Scalable Implementation of a Distributed DBMS Delivering Causal Consistency via the Actor Model,” *Electronics*, vol. 13, no. 17, p. 3367, Aug. 2024, doi: 10.3390/electronics13173367.

- [79] M. Yaroshynskyi, A. Prymushko, I. Puchko, O. Sirotkin, and D. Sinko, “Akka as a tool for modelling and managing a smart grid system,” *J. Edge Comput.*, vol. 4, no. 1, pp. 105–115, May 2025, doi: 10.55056/jec.822.
- [80] National Aviation University, I. V. Puchko, A. M. Prymushko, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», H. O. Kravtsov, and G.E. Pukhov Institute for Modelling in Energy Engineering National Academy of Sciences of Ukraine, “Development of Methodical Recommendations Usage of Functional Paradigm Programming in Scala,” *Elektron. Model.*, vol. 43, no. 6, pp. 95–106, Dec. 2021, doi: 10.15407/emodel.43.06.095.
- [81] P. Hudak, “Conception, evolution, and application of functional programming languages,” *ACM Comput. Surv.*, vol. 21, no. 3, pp. 359–411, Sept. 1989, doi: 10.1145/72551.72554.
- [82] Z. Hu, J. Hughes, and M. Wang, “How functional programming mattered,” *Natl. Sci. Rev.*, vol. 2, no. 3, pp. 349–370, Sept. 2015, doi: 10.1093/nsr/nwv042.
- [83] M. C. Benton and N. M. Radziwill, “Improving Testability and Reuse by Transitioning to Functional Programming,” 2016, *arXiv*. doi: 10.48550/ARXIV.1606.06704.
- [84] P. Haller, H. Miller, and N. Müller, “A programming model and foundation for lineage-based distributed computation,” *J. Funct. Program.*, vol. 28, p. e7, 2018, doi: 10.1017/S0956796818000035.
- [85] P. Haller and H. Miller, “Distributed Programming via Safe Closure Passing,” *Electron. Proc. Theor. Comput. Sci.*, vol. 203, pp. 99–107, Feb. 2016, doi: 10.4204/EPTCS.203.8.
- [86] G. Audrito, R. Casadei, F. Damiani, G. Salvaneschi, and M. Viroli, “Functional Programming for Distributed Systems with XC,” *LIPICs Vol. 222 ECOOP 2022*, vol. 222, p. 20:1-20:28, 2022, doi: 10.4230/LIPICS.ECOOP.2022.20.

- [87] J. S. Moore, “Functional formal methods,” in *Proceedings of the seventh ACM SIGPLAN international conference on Functional programming*, Pittsburgh PA USA: ACM, Sept. 2002, pp. 123–123. doi: 10.1145/581478.581490.
- [88] A. I. Al-Darrab and A. M. A. Rushdi, “Multi-State Reliability Evaluation of Local Area Networks,” in *2021 National Computing Colleges Conference (NCCC)*, Taif, Saudi Arabia: IEEE, Mar. 2021, pp. 1–6. doi: 10.1109/NCCC49330.2021.9428843.
- [89] D. Weikert, C. Steup, and S. Mostaghim, “Availability-Aware Multiobjective Task Allocation Algorithm for Internet of Things Networks,” *IEEE Internet Things J.*, vol. 9, no. 15, pp. 12945–12953, Aug. 2022, doi: 10.1109/JIOT.2022.3170482.
- [90] F. Masood and A. R. Faridi, “Consensus Algorithms In Distributed Ledger Technology For Open Environment,” in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, India: IEEE, Dec. 2018, pp. 1–6. doi: 10.1109/CCAA.2018.8777695.
- [91] Y. Saito and M. Shapiro, “Optimistic replication,” *ACM Comput. Surv.*, vol. 37, no. 1, pp. 42–81, Mar. 2005, doi: 10.1145/1057977.1057980.
- [92] P. S. Almeida, “Approaches to Conflict-free Replicated Data Types,” *ACM Comput. Surv.*, vol. 57, no. 2, pp. 1–36, Feb. 2025, doi: 10.1145/3695249.
- [93] M. Letia, N. Preguiça, and M. Shapiro, “CRDTs: Consistency without concurrency control,” 2009, *arXiv*. doi: 10.48550/ARXIV.0907.0929.
- [94] N. Saquib, C. Krintz, and R. Wolski, “Log-Based CRDT for Edge Applications,” in *2022 IEEE International Conference on Cloud Engineering (IC2E)*, CA, USA: IEEE, Sept. 2022, pp. 126–137. doi: 10.1109/IC2E55432.2022.00021.
- [95] M. Kleppmann, D. P. Mulligan, V. B. F. Gomes, and A. R. Beresford, “A Highly-Available Move Operation for Replicated Trees,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1711–1724, July 2022, doi: 10.1109/TPDS.2021.3118603.

- [96] Glushkov Institute of Cybernetics NAS of Ukraine and B. O. Biletsky, “Horizontal and Vertical Scalability of Machine Learning Methods,” *Probl. Program.*, no. 2, pp. 069–080, 2019, doi: 10.15407/pp2019.02.069.
- [97] M. Balazinska, H. Balakrishnan, S. R. Madden, and M. Stonebraker, “Fault-tolerance in the borealis distributed stream processing system,” *ACM Trans. Database Syst.*, vol. 33, no. 1, pp. 1–44, Mar. 2008, doi: 10.1145/1331904.1331907.
- [98] A. Oshnoei, S. Peyghami, H. Mokhtari, and F. Blaabjerg, “Grid Synchronization for Distributed Generations,” in *Encyclopedia of Sustainable Technologies*, Elsevier, 2024, pp. 517–537. doi: 10.1016/B978-0-323-90386-8.00079-6.
- [99] S. Olmi, L. V. Gambuzza, and M. Frasca, “Multilayer control of synchronization and cascading failures in power grids,” *Chaos Solitons Fractals*, vol. 180, p. 114412, Mar. 2024, doi: 10.1016/j.chaos.2023.114412.
- [100] B. Portela, H. Pacheco, P. Jorge, and R. Pontes, “General-Purpose Secure Conflict-free Replicated Data Types,” in *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, Dubrovnik, Croatia: IEEE, July 2023, pp. 521–536. doi: 10.1109/CSF57540.2023.00030.
- [101] N. Preguiça, “Conflict-free Replicated Data Types: An Overview,” 2018, *arXiv*. doi: 10.48550/ARXIV.1806.10254.
- [102] V. Enes, P. S. Almeida, C. Baquero, and J. Leita, “Efficient Synchronization of State-Based CRDTs,” in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, Macao, Macao: IEEE, Apr. 2019, pp. 148–159. doi: 10.1109/ICDE.2019.00022.
- [103] “Scala language”. Accessed: Jun. 01, 2025.[Online]. Available: <https://www.scala-lang.org/>
- [104] “Akka”. Accessed: Jun. 01, 2025. [Online]. Available: <https://doc.akka.io/libraries/akka-core/2.6/>

[105] “The Reactive Principles”. Accessed: Jun. 01, 2025. [Online]. Available: <https://www.reactiveprinciples.org/>

[106] M. Yaroshynskyi, I. Puchko, A. Prymushko, H. Kravtsov, and V. Artemchuk, “Investigating the Evolution of Resilient Microservice Architectures: A Compatibility-Driven Version Orchestration Approach,” *Digital*, vol. 5, no. 3, p. 27, July 2025, doi: 10.3390/digital5030027.

[107] M. Larsson, *Microservices with Spring Boot 3 and Spring Cloud: build resilient and scalable microservices using Spring Cloud, Istio, and Kubernetes, third edition*, 3rd ed. Place of publication not identified: Packt Publishing, 2023.

[108] O. Sirotkin, A. Prymushko, I. Puchko, H. Kravtsov, M. Yaroshynskyi, and V. Artemchuk, “Parallel Simulation Using Reactive Streams: Graph-Based Approach for Dynamic Modeling and Optimization,” *Computation*, vol. 13, no. 5, p. 103, Apr. 2025, doi: 10.3390/computation13050103.

[109] *Vigilant Hawk*. Accessed: Jun. 01, 2025. [Online]. Available: <https://github.com/ipk0/vigilant-hawk>

ДОДАТОК А

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

1. **I.V. Пучко**, А.М. Примушко, Г.О. Кравцов, Розробка методичних рекомендацій використання функціональної парадигми програмування в мові Scala. *Електронне моделювання*, Том 43, № 6 (2021) с. 95-106, doi: 10.15407/emodel.43.06.095. Фахове видання категорії Б. (Особистий внесок – структурування та написання методичних рекомендацій).
2. А.М. Примушко, **I.V. Пучко**, М.С. Ярошинський, Д.П. Сінько, Програмний дизайн розподіленої високонавантаженої системи електроенергетичної мережі на базі моделі акторів із застосуванням смарт-контрактів; *Електронне моделювання*, Том 46, № 3 (2024) с. 57-72 doi: 10.15407/emodel.46.03.057. Фахове видання категорії Б. (Особистий внесок – брав участь в розробці високорівневого дизайну розподіленої електроенергетичної системи на основі моделі акторів та кластерної топології).
3. A. Prymushko , **I. Puchko**, M. Yaroshynskyi, D. Sinko, H. Kravtsov, and V . Artemchuk, ‘Efficient State Synchronization in Distributed Electrical Grid Systems Using Conflict-Free Replicated Data Types’, *IoT*, vol. 6, no. 1, p. 6, Jan. 2025, doi: 10.3390/iot6010006. **Indexed in Scopus Q1**. (Особистий внесок – брав участь в розробці моделі розподілених електроенергетичних систем, що базується на неконфліктних реплікованих типах даних. Спроектував і виконав основну частину розроблення програмного забезпечення “Vigilant Hawk” для симуляції розподілених електроенергетичних систем на базі акторної моделі та інструментарію Akka. Провів експериментальні дослідження, отримав та обробив результати, виконав їхню візуалізацію у вигляді графіків).
4. O. Sirotkin; A. Prymushko; **I. Puchko**; H. Kravtsov; M. Yaroshynskyi; V. Artemchuk. Parallel Simulation Using Reactive Streams: Graph-Based Approach for Dynamic Modeling and Optimization. *Computation*, vol. 13, no. 5, p. 103, Apr. 2025, doi: 10.3390/computation13050103. **Indexed in Scopus**

Q2. (Особистий внесок – брав участь в оптимізації симуляції на основі графів та реактивних потоків за допомогою визначених функцій переходу).

5. M. Yaroshynskyi, A. Prymushko , **I. Puchko**, O. Sirotkin, and D. Sinko, ‘Akka as a tool for modelling and managing a smart grid system’, *Journal of Edge Computing*, vol. 4, no. 1, pp. 105–115, May 2025, doi: 10.55056/jec.822.

Indexed in Scopus. (Особистий внесок – брав участь у проєктуванні моделі управління інтелектуальною мережею, побудованої на ієрархічній структурі акторів. Провів експериментальні дослідження, обробив та візуалізував результати).

6. M. Yaroshynskyi, **I. Puchko**, A. Prymushko , H. Kravtsov, and V . Artemchuk, ‘Investigating the Evolution of Resilient Microservice Architectures: A Compatibility-Driven Version Orchestration Approach’, *Digital*, vol. 5, no. 3, p. 27, July 2025, doi: 10.3390/digital5030027. **Indexed in Scopus Q2.**

(Особистий внесок – брав участь у дослідженні та порівняльному аналізі існуючих стратегій управління еволюцією API. Розробив експериментальні сервіси на базі Spring Boot з використанням gRPC та для перевірки моделі).

ПРОДОВЖ. ДОД. А

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ

1. **І.В. Пучко**, А.М. Примушко, М.С. Ярошинський, Г.О. Кравцов, ‘Підвищення резильєнтності динамічних систем при синхронізації станів за допомогою CRDT’, Матеріали науково-практичної конференції ‘Резильєнтність динамічних систем’, с. 50–52, Київ, Україна, 2024 URL: <https://ipme.kiev.ua/konferencii/naukovo-praktichna-konferenciya-rds-2024/>.
2. A. Prymushko, M. Yaroshynskyi, and **I. Puchko**, ‘Representation and synchronization of states of distributed electrical grid systems based on conflict free replicated data types’, in 2024 *14th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece: IEEE, Oct. 2024, pp. 1–5. doi: 10.1109/DESSERT65323.2024.11122143.
Indexed in Scopus.

ДОДАТОК Б

Код для обробки даних та візуалізації графіків*

```

import pandas as pd
import matplotlib.pyplot as plt

csv_file_path = '../data.csv'
data = pd.read_csv(csv_file_path, header=None, names=["x", "y", "value"])
data["x"] = pd.to_numeric(data["x"], errors='coerce')
data["y"] = pd.to_numeric(data["y"], errors='coerce')
data["value"] = pd.to_numeric(data["value"], errors='coerce')
new_data = data.copy()
new_data = new_data['value'] = new_data['value'].replace({
    0: "g",
    1: "y",
    2: "r"
})
red_points = data[data["value"] == 2]
yellow_points = data[data["value"] == 1]
green_points = data[data["value"] == 0]
plt.figure(figsize=(10, 6))
plt.plot(green_points["x"], green_points["y"], color="green", label=r'$P_{actual} < P_{nominal}$', linewidth=1)
plt.plot(yellow_points["x"], yellow_points["y"], color="yellow", label=r'$P_{actual} = P_{nominal}$', linewidth=1)
plt.plot(red_points["x"], red_points["y"], color="red", label=r'$P_{actual} > P_{nominal}$', linewidth=1)
plt.title("LDK={value}", fontsize=16, weight='bold')
plt.xlabel("t", fontsize=12)
plt.ylabel("n", fontsize=12, rotation=0)
plt.grid(visible=True, linestyle='--', alpha=0.7)
plt.legend(loc="upper left", fontsize=10)
output_file_path = '../data/output_chart.png'
plt.savefig(output_file_path, dpi=300, bbox_inches='tight', format='png')

```

* Всі вихідні коди проєкту доступні у відкритому доступі в репозиторії GitHub [109]