МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

Кваліфікаційна наукова праця
на правах рукопису

Сиволовський Ілля Михайлович

УДК 621.39:004.6:004.9:004.272

ДИСЕРТАЦІЯ
МЕТОДИ КЕРУВАННЯ СКЛАДНОСТРУКТУРОВАНИМИ ДАНИМИ В РОЗПОДІЛЕНИХ ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ

Спеціальність 172 – Телекомунікації та радіотехніка
Галузь знань 17 – Електроніка та телекомунікації

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.М. Сиволовський

Науковий керівник:
Лисечко Володимир Петрович,
доктор технічних наук, професор

Харків – 2025

АНОТАЦІЯ

Сиволовський І.М. Методи керування складноструктурованими даними в розподілених телекомунікаційних системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії (PhD) за спеціальністю 172 – Телекомунікації та радіотехніка. – Український державний університет залізничного транспорту, Україна, Харків, 2025.

В дисертаційній роботі вирішується актуальне науково-технічне завдання підвищення ефективності та надійності обробки даних у розподілених телекомунікаційних системах (РТС) шляхом розробки методів і моделей ієрархічної кластеризації, динамічної самоорганізації обчислювальних вузлів, вибору головного вузла та формування обчислювальних конвеєрів, що забезпечує зниження затримок, рівномірний розподіл навантаження та підвищення відмовостійкості мережі.

Об'єктом дослідження є процес організації обробки даних у розподілених телекомунікаційних системах.

Предметом дослідження є методи, моделі та алгоритми ієрархічної кластеризації та динамічної самоорганізації вузлів у розподілених телекомунікаційних системах, спрямовані на підвищення їх ефективності, надійності та адаптивності до змінних умов.

Метою дисертаційного дослідження є підвищення ефективності та надійності обробки даних у розподілених телекомунікаційних системах шляхом розробки методів і моделей ієрархічної кластеризації, багатокритеріального розподілу потоків даних та динамічної самоорганізації обчислювальних вузлів в умовах змінного середовища.

У вступі обґрунтовано актуальність підвищення ефективності та стійкості обробки складноструктурованих даних у РТС в умовах динамічних змін навантаження, параметрів мережі та зростаючих вимог до обчислювальних ресурсів. Представлено наукову новизну, що полягає в розробці методів ієрархічної кластеризації вузлів, вибору головного вузла, формування конвеєрів обробки даних

і багатокритеріального розподілу потоків даних з використанням удосконаленого генетичного алгоритму. Обґрунтовано практичне значення отриманих результатів для забезпечення масштабованості, зниження затримок, рівномірного завантаження ресурсів та підвищення стійкості функціонування РТС. Наведено особистий внесок здобувача та перелік наукових публікацій, виконаних за темою дисертації.

Перший розділ присвячено аналізу сучасних підходів до обробки складноструктурованих даних у РТС. Розглянуто архітектури хмарних, туманних та гібридних середовищ, а також чинники, що впливають на ефективність РТС. Виявлено недоліки традиційних підходів, які стали підґрунтям для формулювання вимог до нових адаптивних та масштабованих методів організації і керування РТС. Обґрунтовано необхідність врахування при розробці нових адаптивних методів структурної складності телекомунікаційної інфраструктури, динаміки навантаження і параметрів мережевих з'єднань для забезпечення стійкої та ефективної роботи РТС.

У **другому розділі** вперше запропоновано комплексний метод самоорганізації РТС, який реалізується поетапно: від первинного об'єднання вузлів з урахуванням апаратних можливостей і параметрів мережевих з'єднань – до динамічного балансування навантаження, переназначення і вибору головного вузла. Розроблено та реалізовано перший рівень запропонованого методу на основі удосконаленого алгоритму Лувена з адаптивним параметром роздільної здатності та рекурсивним уточненням кластерної структури. Це дозволяє уникати надмірної агрегації вузлів, збільшити збалансованість навантаження і зберігати стабільну продуктивність при зміні топології мережі. Експериментально доведено, що застосування методу знижує міжкластерні затримки обробки і передачі даних, а також зменшує розкид у розмірах кластерів, що, в свою чергу, сприяє рівномірному розподілу обчислювальних ресурсів і стійкості РТС до змін трафіку.

У **третьому розділі** розроблено інтегрований метод керування інформаційними потоками в умовах нестабільної мережі, який складається з методу вибору головного вузла і методу формування обчислювальних конвеєрів потоків обробки даних. Запропоновано модифікацію алгоритму вибору лідера – «Пліткар» (Gossip) з ресурсно-затримковою оцінкою вузлів та асинхронно-узгодженим вибором головного вузла, що забезпечує стійкість до збоїв і скорочення службового

трафіку. Запропоновано метод формування обчислювальних конвеєрів, що забезпечує послідовну маршрутизацію потоків між кластерами з урахуванням продуктивності вузлів, ресурсних обмежень та пропускної здатності. Доведено, що за допомогою методу можна уникнути перевантаження, зменшити затримки та ефективно розподілити завдання між етапами обробки. Розроблено алгоритм реалізації на основі спрямованого багатошарового графа з урахуванням міжкластерних затримок, штрафів за перевантаження та паралельного прорахунку маршрутів. Запропоновано механізм фільтрації непрохідних вузлів і пріоритетного вирішення конфліктів, що забезпечує збільшення масштабованості і стабільності роботи при пікових навантаженнях. Експериментально доведено ефективність запропонованих методів до вимог РТС.

Четвертий розділ присвячено розробці методу інтелектуального розподілу інформаційних потоків у РТС, який ґрунтується на принципах еволюційної багатокритеріальної оптимізації та удосконаленому алгоритмі NSGA-III. Запропоновано формалізацію задачі розподілу потоків з урахуванням обмежених ресурсів, динаміки трафіку, ризику перевантаження та потреби в делегуванні. Метод передбачає самоорганізовану координацію між кластерами, використання цілочисельного кодування рішень, гібридної функції пристосованості та адаптивного прогнозування навантаження. Наведено результати експериментального моделювання, які доводять ефективність запропонованого методу та алгоритму реалізації за умов нерівномірного навантаження, нестабільної ресурсної доступності та потреби в швидкому розподілі потоків.

Висновки до дисертаційної роботи містять основні результати, які вирішують актуальне науково-технічне завдання та розв’язують часткові задачі, поставлені в дисертаційному дослідженні.

Основні результати дисертаційного дослідження сформульовано у вигляді пунктів **наукової новизни**:

1. **Вперше** розроблено комплексний метод самоорганізації розподілених телекомунікаційних систем, який, на відміну від традиційних підходів, враховує продуктивність вузлів, топологію мережі, затримки та пропускну здатність каналів передачі, що досягається за рахунок адаптивного налаштування

параметрів кластеризації та багаторівневого аналізу мережевих зв’язків.

2. **Вперше** розроблено інтегрований метод керування інформаційними потоками в умовах нестабільної мережі із забезпеченням вибору головного вузла та одночасного формування обчислювальних конвеєрів у розподілених телекомунікаційних системах, шляхом поєднання ресурсно-затримкової оцінки характеристик вузлів та модифікованого алгоритму з асинхронно-узгодженим вибором керуючого вузла, що забезпечує мінімізацію часових затримок.

3. **Удосконалено** метод багатокрокової ієрархічної кластеризації на основі застосування модифікованого алгоритму Лувена, який, на відміну від відомих підходів, забезпечує контроль розміру та щільності кластерів завдяки динамічному регулюванню параметра роздільної здатності та рекурсивному уточненню кластерної структури.

4. **Удосконалено** метод багатокритеріального розподілу потоків обробки даних у телекомунікаційних системах на основі модифікованого генетичного алгоритму NSGA-III, з метою забезпечення підвищення точності розподілу та зниження числа делегацій між кластерами, шляхом використання механізму гібридного ранжування та адаптивного регулювання частоти мутацій в залежності від прогнозованого навантаження.

Практичні результати, отримані в дисертаційній роботі, полягають у розробці та впровадженні запропонованих методів у вигляді технічних рішень та алгоритмів для підвищення ефективності та надійності РТС, а саме:

– на основі порівняльного аналізу сучасних архітектур і методів обробки даних у РТС обґрунтовано доцільність розробки нових адаптивних методів та алгоритмів кластеризації, самоорганізації та багатокритеріальної оптимізації, що враховують динаміку навантаження, структурну складність і параметри мережевих з’єднань, включаючи затримки, пропускну здатність та продуктивність вузлів;

– розроблено алгоритм і програмну реалізацію методу самоорганізації РТС, що дозволяє знизити затримки в обробці та передачі даних до 17 % в порівнянні з методами Лувена і Лейдена та одночасно покращити узгодженість

структури кластерів, що, в свою чергу, зменшує витрати ресурсів на маршрутизацію на 10–15 % залежно від топології мережі і навантаження;

– удосконалено алгоритм Лувена шляхом формування кластерів з адаптивною глибиною, що усуває надмірну агрегацію, забезпечує рівномірне навантаження вузлів і зменшує стандартне відхилення розмірів кластерів на 53 – 66 %, що необхідно для стабільної роботи РТС в умовах змінної топології;

– в межах інтегрованого методу керування інформаційними потоками удосконалено алгоритм вибору головного вузла, за рахунок застосування асинхронно-узгодженого вибору координатора, що скорочує час відновлення після відмови головного вузла на 43 % та знижує службовий трафік до 30 %, завдяки виключенню явної фази виборів і використанню наперед сформованих списків резервних кандидатів;

– розроблено алгоритм та програмну реалізацію методу формування обчислювальних конвеєрів для обробки потокових даних, що у поєднанні з удосконаленим алгоритмом вибору координатора забезпечує динамічну самоорганізацію мережі зі скороченням затримок на 5–7%, зменшенням кількості перевантажень до 20% і підвищенням рівномірності завантаження ресурсів вузлів на 38–54%;

– розроблено модифікований алгоритм формування обчислювальних конвеєрів на основі спрямованого багатошарового графа з параметричним урахуванням затримок, ресурсних обмежень та штрафів за перевантаження, що забезпечує паралельну маршрутизацію потоків і адаптацію до змінних умов навантаження в РТС;

– реалізовано модифікований варіант генетичного алгоритму NSGA-III, який поєднує гібридне ранжування і адаптивне регулювання частоти мутацій, що забезпечує зменшення середнього перевантаження вузлів на 53,7%, скорочення пікових перевантажень втричі та підвищення рівномірності завантаження ресурсів у 2,1 рази.

Отримані результати підтверджують теоретичну і прикладну цінність розроблених методів кластеризації, самоорганізації та оптимізації в задачах

обробки даних у розподілених телекомунікаційних системах, що відкриває перспективи їх подальшого впровадження та розвитку.

Ключові слова: розподілені телекомунікаційні системи, комп'ютерні інформаційні системи, самоорганізація, взаємодія підсистем, архітектура, передача та обробка потоку даних, адаптація, оптимізація, математичне моделювання, еволюційний підхід, генетичний алгоритм, моніторинг, топологія мережі, трафік, маршрутизація, графові методи, керуючий вузол, сервер, підтримка прийняття рішень, обчислювальна складність, ієрархія, кластеризація, розподіл навантаження, продуктивність, рівень затримок, відмовостійкість, бази даних.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати:

1. Mazurova, O., Syvolovskyi, I., & Syvolovska, O. (2022). NoSQL database logic design methods for MongoDB and Neo4j// Innovative technologies and scientific solutions for industries/Kharkiv national university of radio electronics/ №2 (20), (2022) P. 52–63. https://doi.org/10.30837/ITSSI.2022.20.052.

2. Syvolovskyi, I. M., Lysechko, V. P., Komar, O. M., Zhuchenko, O. S., Pastushenko, V. V. (2024) Analysis of methods for organizing distributed telecommunication systems using the paradigm of Edge Computing. *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, 1(75), (2024) P. 206-211, https://doi.org/10.26906/SUNZ.2024.1.206.

3. Syvolovskyi, I., & Komar, O. (2025). A method of multicriteria data stream distribution in telecommunication networks based on an evolutionary approach// Computer-integrated technologies: education, science, production/ Lutsk National Technical University. – Lutsk. – 2025. (№59), P.P. 230-239. https://doi.org/10.36910/6775-2524-0560-2025-59-41.

4. Syvolovskyi, I. M., Lysechko V. P. (2025) A method of hierarchical clustering of nodes in distributed telecommunication systems using graph algorithms// *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, Vol. 2, № 80 (2025), P.P. 255-262, https://doi.org/10.26906/SUNZ.2025.2.255.

5. Syvolovskyi I.M., Lysechko V.P. (2025) Method for leader node selection and processing pipeline formation in distributed telecommunication systems. – National Aviation University. Science-intensive Technologies. Series: «Electronics, telecommunications and radio engineering», Kyiv, 2025. Vol. 66, № 2, PP. 190-200 https://doi.org/10.18372/2310-5461.66.20311.

Опубліковані праці апробаційного характеру:

6. Lysechko V., Syvolovskyi I., Shevchenko B., Nikitska A., Cherneva G. Research of modern NoSQL databases to simplify the process of their design. Mechanics, Transport, Communications. Journal article № 2363, vol. 21, issue 2, 2023 https://mtc-aj.com/library/2363_EN.pdf. ***[Міжнародне видання]***

7. Syvolovskyi I., Pastushenko V. Analysis of edge computing architectures in distributed telecommunication systems// Інформаційно-керуючі системи на залізничному транспорті: тези доповідей та виступів учасників 36-ї Міжнародної науково-практичної конференції «Інформаційно-керуючі системи на залізничному транспорті». – 2023. – № 3 (додаток). – С. 10-12.

8. Syvolovskyi I.M., Frolov D.Y. Analysis of load distribution methods in distributed telecommunication systems //Перспективи розвитку озброєння та військової техніки сухопутних військ. Збірник тез доповідей Міжнародної науково-технічної конференції (Львів 15-16 травня 2024). Національна академія сухопутних військ імені гетьмана Петра Сагайдачного, Issue 30 P. 308-309.

9. Syvolovskyi, I.M. Methods to improve the performance of distributed telecommunication systems by changing their architecture/ I.M. Syvolovskyi, O.S. Zhuchenko, R.O. Sarapin// Інформаційно-керуючі системи на залізничному транспорті: тези доповідей та виступів учасників 37-ї Міжнародної науково-практичної конференції «Інформаційно-керуючі системи на залізничному транспорті» (Харків, 10-11 жовтня, 2024 р.). – 2024. – № 3 (додаток). – С. 95-96.

10. Syvolovskyi I.M., Lysechko V.P. Methods of network optimization for streaming data processing in distributed systems // Тези доповіді за матеріалами I міжвузівською наукової конференції *«Стан та перспективи розвитку інфокомунікацій в сучасних умовах»*. Харківський національний університет повітряних сил ім. І. Кожедуба. Харків: 22 листопада 2024. С. 87-88.

11. Syvolovskyi I.M., Zhuchenko O.S. Methods of formation and routing of a distributed telecommunication networks // Тези доповіді за матеріалами науково-практичної конференції «Сектор безпеки і оборони України на захисті

національних інтересів: актуальні проблеми та завдання в умовах воєнного стану». Національна академія Державної прикордонної служби України імені Богдана Хмельницького, 22-23 листопада 2024, С. 709-712.

12. Syvolovskyi I.M., Lysechko V.P. Topology optimization method of a distributed telecommunication system for stream data processing // XXI Міжнародна наукова конференція «Новітні технології для захисту повітряного простору». Харківський національний університет повітряних сил ім. І. Кожедуба. Харків: 9 – 10 квітня 2025 року. С. 333-334.

13. Сиволовський І.М., Лисечко В.П., Пастушенко В.В. Багатокритеріальний розподіл інформаційних потоків у телекомунікаційних системах на основі модифікованого генетичного алгоритму// Перспективи розвитку озброєння та військової техніки сухопутних військ. Збірник тез доповідей Міжнародної науково-технічної конференції (Львів 14-15 травня 2025). Національна академія сухопутних військ імені гетьмана Петра Сагайдачного. – Львів: НАСВ, 2025, С. 91.

14. Мазурова О.О., Сиволовський І.М. Дослідження методів логічного моделювання NOSQL баз даних для ігрових серверних систем // Тези доповідей 12 міжнародної науково-технічної конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління», 27 – 28 квітня 2022 року, Том 2: секція 5, С.148-149.

ABSTRACT

Syvolovskyi I.M. Methods for Managing Complexly Structured Data in Distributed Telecommunication Systems. Qualifying scientific work on manuscript rights.

Dissertation for obtaining a scientific doctor of philosophy (PhD) in specialty 172 – Telecommunications and Radio Engineering – Ukrainian State University of Railway Transport, Ukraine, Kharkiv, 2025.

The dissertation addresses a relevant scientific and technical problem of improving the efficiency and reliability of data processing in distributed telecommunication systems (DTS) by developing methods and models for hierarchical clustering, dynamic self-organization of computational nodes, leader node selection, and computational pipeline formation, which ensure reduced latency, balanced load distribution, and enhanced network fault tolerance.

Object of the research is the process of organizing data processing in distributed telecommunication systems (DTS).

Subject of the research is the methods, models, and algorithms of hierarchical clustering and dynamic self-organization of nodes in DTS, aimed at improving their efficiency, reliability, and adaptability to changing conditions.

The aim of the dissertation is to improve the efficiency and reliability of data processing in DTS by developing methods and models of hierarchical clustering, multi-criteria data flow distribution, and dynamic self-organization of computational nodes under variable environmental conditions.

In the introduction the relevance of improving the efficiency and resilience of processing complex-structured data in DTS under conditions of dynamic changes in load, network parameters, and increasing demands on computational resources is substantiated. The scientific novelty lies in the development of methods for hierarchical node clustering, head node selection, construction of data processing pipelines, and multi-criteria data flow distribution using an enhanced genetic algorithm. The practical significance of the obtained results is justified in terms of ensuring scalability, reducing

delays, balancing resource utilization, and enhancing the resilience of DTS operation. The personal contribution of the author and the list of scientific publications on the dissertation topic are presented.

The first chapter is devoted to the analysis of modern approaches to processing complex-structured data in DTS. It examines the architectures of cloud, fog, and hybrid environments, as well as the factors influencing DTS efficiency. The shortcomings of traditional approaches are identified, which served as the basis for formulating the requirements for new adaptive and scalable methods of DTS organization and management. The necessity of considering the structural complexity of telecommunication infrastructure, load dynamics, and network connection parameters in the development of new adaptive methods is substantiated to ensure stable and efficient operation of DTS.

The second chapter introduces, for the first time, a comprehensive method of self-organization for DTS, implemented step by step: from the initial grouping of nodes based on hardware capabilities and network connection parameters to dynamic load balancing, task reassignment, and leader node selection. The first level of the proposed method has been developed and implemented using an enhanced Louvain algorithm with an adaptive resolution parameter and recursive refinement of the cluster structure. This approach prevents excessive node aggregation, improves load balance, and maintains stable performance under network topology changes. Experimental results demonstrate that applying the method reduces inter-cluster delays in data processing and transmission, as well as decreases cluster size variance, which in turn ensures more uniform resource allocation and enhances DTS resilience to traffic fluctuations.

The third chapter, an integrated method for managing information flows under unstable network conditions was developed, which consists of a method for selecting the main node and a method for forming computational pipelines for data stream processing. A modification of the Gossip-based leader election algorithm is proposed, incorporating resource–latency evaluation of nodes and asynchronously coordinated leader selection, which ensures fault tolerance and reduces control traffic. A method for forming computational pipelines is also proposed, enabling sequential

routing of data streams between clusters while considering node performance, resource constraints, and channel capacity. It is demonstrated that this method prevents overloads, reduces delays, and enables efficient task distribution across processing stages. An implementation algorithm has been developed based on a directed multilayer graph, taking into account inter-cluster delays, overload penalties, and parallel route computation. Additionally, a mechanism for filtering non-viable nodes and prioritizing conflict resolution is introduced, which increases scalability and operational stability under peak loads. Experimental evaluation confirmed that the proposed methods meet the requirements of DTS.

The fourth chapter is devoted to the development of a method for intelligent distribution of information flows in DTS, based on the principles of evolutionary multi-criteria optimization and an improved NSGA-III algorithm. The problem of flow distribution is formalized with consideration of limited resources, traffic dynamics, overload risk, and delegation requirements. The method provides for self-organized coordination between clusters, the use of integer-coded solutions, a hybrid fitness function, and adaptive load prediction. The feasibility of applying dynamic reservation mechanisms depending on changing traffic parameters is substantiated. Experimental modeling results are presented, demonstrating the effectiveness of the proposed method and its implementation algorithm under conditions of uneven load, unstable resource availability, and the need for rapid flow allocation.

The conclusions of the dissertation summarize the main results that address the relevant scientific and technical problem and solve the specific tasks set in the research.

The main results of the dissertation are formulated as the key points of **scientific findings**:

1. For the first time, a comprehensive method of self-organization for distributed telecommunication systems has been developed, which, unlike traditional approaches, takes into account node performance, network topology, delays, and channel capacity. This is achieved through adaptive adjustment of clustering parameters and multi-level analysis of network connections.

2. For the first time, an integrated method for managing information flows under unstable network conditions has been developed, providing both the selection of the main node and the simultaneous formation of computational pipelines in distributed telecommunication systems. The method combines resource–latency evaluation of node characteristics with a modified algorithm for asynchronously coordinated leader node selection, ensuring the minimization of time delays.

3. The method of multi-step hierarchical clustering based on a modified Louvain algorithm, developed within the dissertation, has been improved. Unlike other approaches, it provides control over cluster size and density through dynamic regulation of the resolution parameter and recursive refinement of the cluster structure.

4. The method of multi-criteria distribution of data processing flows in telecommunication systems has been improved on the basis of a modified NSGA-III genetic algorithm. The improvement aims to increase distribution accuracy and reduce the number of delegations between clusters by employing a hybrid ranking mechanism and adaptive regulation of mutation frequency depending on the predicted load.

The practical results obtained in the dissertation consist in the development and implementation of the proposed methods in the form of technical solutions and algorithms to improve the efficiency and reliability of distributed telecommunication systems (DTS), namely:

– based on a comparative analysis of modern architectures and data processing methods in DTS, the feasibility of developing new adaptive methods and algorithms for clustering, self-organization, and multi-criteria optimization has been substantiated. These methods take into account load dynamics, structural complexity, and network connection parameters, including delays, bandwidth, and node performance;

– an algorithm and software implementation of the self-organization method for DTS have been developed, which reduce data processing and transmission delays by up to 17% compared to the Louvain and Leiden methods, and improve cluster structure consistency. This, in turn, decreases routing resource consumption by 10–15%, depending on network topology and load;

– the Louvain algorithm has been improved by introducing adaptive cluster depth formation, which eliminates excessive aggregation, ensures even node load distribution, and reduces the standard deviation of cluster sizes by 53–66%, which is essential for stable DTS operation under variable topology;

– the leader election algorithm based on the modified «Gossip» protocol with asynchronous coordinator selection has been enhanced, reducing recovery time after the failure of the main node by 43% and lowering service traffic by up to 30% due to the exclusion of the explicit election phase and the use of pre-formed lists of backup candidates;

– an algorithm and software implementation of the method for forming computational pipelines for stream data processing have been developed, which, in combination with the improved coordinator selection algorithm, provide dynamic self-organization of the network with a reduction of delays by 5–7%, a decrease in the number of overloads by up to 20%, and an increase in the uniformity of node resource utilization by 38–54%;

– a modified algorithm for forming computational pipelines based on a directed multilayer graph has been proposed, which parametrically considers delays, resource constraints, and overload penalties. This ensures parallel flow routing and adaptation to changing load conditions in DTS;

– a modified version of the NSGA-III genetic algorithm has been implemented, combining hybrid ranking and adaptive mutation rate adjustment, which resulted in a 53.7% reduction in average node overload, a threefold decrease in peak overloads, and a 2.1-fold improvement in the uniformity of resource utilization. On this basis, the feasibility of using distributed databases as a platform for storing the results of computational pipeline processing has been substantiated.

The obtained results confirm the theoretical and applied value of the developed methods of clustering, self-organization, and optimization for data processing tasks in distributed telecommunication systems, opening up prospects for their further implementation and development.

Keywords: distributed telecommunication systems, computer information systems, self-organization, subsystem interaction, architecture, data flow transmission and processing, adaptation, optimization, mathematical modeling, evolutionary approach, genetic algorithm, monitoring, network topology, traffic, routing, graph-based methods, leader node, server, decision support, computational complexity, hierarchy, clustering, load distribution, performance, latency level, fault tolerance, databases.

LIST OF PUBLICATIONS OF THE ACQUIRER

Scientific works in which the main scientific results were published:

1. Mazurova, O., Syvolovskyi, I., & Syvolovska, O. (2022). NoSQL database logic design methods for MongoDB and Neo4j// Innovative technologies and scientific solutions for industries/Kharkiv national university of radio electronics/ №2 (20), (2022), P. 52–63. https://doi.org/10.30837/ITSSI.2022.20.052.

2. Syvolovskyi, I. M., Lysechko, V. P., Komar, O. M., Zhuchenko, O. S., Pastushenko, V. V. (2024) Analysis of methods for organizing distributed telecommunication systems using the paradigm of Edge Computing. 2024. *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, 1(75), (2024) P. 206-211, https://doi.org/10.26906/SUNZ.2024.1.206.

3. Syvolovskyi , I., & Komar O. (2025). A method of multicriteria data stream distribution in telecommunication networks based on an evolutionary approach// Computer-integrated technologies: education, science, production/ Lutsk National Technical University. – Lutsk. – 2025. (№59), P.P. 230-239. https://doi.org/10.36910/6775-2524-0560-2025-59-41.

4. Syvolovskyi, I. M., Lysechko V. P. (2025) A method of hierarchical clustering of nodes in distributed telecommunication systems using graph algorithms// *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, Vol. 2, № 80 (2025), P.P. 255-262, https://doi.org/10.26906/SUNZ.2025.2.255.

5. Syvolovskyi I.M., Lysechko V.P. (2025) Method for leader node selection and processing pipeline formation in distributed telecommunication systems. – National Aviation University. Science-intensive Technologies. Series: «Electronics, telecommunications and radio engineering», Kyiv, 2025. Vol. 66, № 2, PP. 190-200 https://doi.org/10.18372/2310-5461.66.20311.

Published works of an approbation nature:

6. Lysechko V., Syvolovskyi I., Shevchenko B., Nikitska A., Cherneva G. Research of modern NoSQL databases to simplify the process of their design. Mechanics, Transport, Communications. Journal article № 2363, vol. 21, issue 2, 2023 https://mtc-aj.com/library/2363_EN.pdf. ***[International journal]***

7. Syvolovskyi I., Pastushenko V. Analysis of edge computing architectures in distributed telecommunication systems // Information-Control Systems on Railway Transport: Abstracts of the 36th International Scientific and Practical Conference. – 2023. – No. 3. – P. 10–12.

8. Syvolovskyi I.M., Frolov D.Y. Analysis of load distribution methods in distributed telecommunication systems // Prospects for the Development of Weapons and Military Equipment of the Ground Forces: Proceedings of the International Scientific and Technical Conference (Lviv, May 15–16, 2024). Hetman Petro Sahaidachnyi National Army Academy. – Issue 30. – P. 308–309.

9. Syvolovskyi I.M., Zhuchenko O.S., Sarapin R.O. Methods to improve the performance of distributed telecommunication systems by changing their architecture // Information-Control Systems on Railway Transport: Abstracts of the 37th International Scientific and Practical Conference (Kharkiv, October 10–11, 2024). – 2024. – No. 3. – P. 95–96.

10. Syvolovskyi I.M., Lysechko V.P. Methods of network optimization for streaming data processing in distributed systems // Abstracts of the 1st Inter-University Scientific Conference «State and Prospects for the Development of Infocommunications in Modern Conditions». Ivan Kozhedub Kharkiv National Air Force University, Kharkiv, November 22, 2024. – P. 87–88.

11. Syvolovskyi I.M., Zhuchenko O.S. Methods of formation and routing of distributed telecommunication networks // Abstracts of the Scientific and Practical Conference «Ukraine's Security and Defense Sector in Protecting National Interests: Current Problems and Tasks in Martial Law Conditions». Bohdan Khmelnytskyi

National Academy of the State Border Guard Service of Ukraine, November 22–23, 2024. – P. 709–712.

12. Syvolovskyi I.M., Lysechko V.P. Topology optimization method of a distributed telecommunication system for stream data processing // 21st International Scientific Conference «Advanced Technologies for Airspace Protection». Ivan Kozhedub Kharkiv National Air Force University, Kharkiv, April 9–10, 2025. – P. 333–334.

13. Syvolovskyi I.M., Lysechko V.P., Pastushenko V.V. Multi-objective distribution of information flows in telecommunication systems based on a modified genetic algorithm //Prospects for the Development of Weapons and Military Equipment of the Ground Forces: Proceedings of the International Scientific and Technical Conference (Lviv, May 14–15, 2025). Hetman Petro Sahaidachnyi National Army Academy. – Lviv: NASV, 2025. – P. 91.

14. Mazurova O.O., Syvolovskyi I.M. Research on logical modeling methods of NoSQL databases for game server systems // Proceedings of the 12th International Scientific and Technical Conference «Modern Trends in the Development of Information and Communication Technologies and Control Systems», April 27–28, 2022, Vol. 2, Section 5. – P. 148–149.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

РТС – розподілені телекомунікаційні системи;

ГА – генетичний алгоритм;

IoT – Internet of Things – Інтернет речей;

F2c2C – Fog-to-cloudlet-to-Cloud – Туман-підхмара-хмара;

MELINDA – Multilevel Information Distributed Processing Architecture – багаторівнева архітектура розподіленої обробки інформації;

MLT / FLT / DLT – Measurement / Feature / Decision Level Task –завдання рівня вимірювання / функціоналу / рішення;

SDN – Software-Defined Network – програмно-визначена мережа;

NFV – Network Functions Virtualization – віртуалізація функцій мережі;

SDNFV – Software-Defined NFV – програмно визначені NFV;

QoS – Quality of Service – якість обслуговування;

SuVMF – Software-defined Unified Virtual Monitoring Function – програмно-визначена уніфікована віртуальна функція моніторингу.

RAM – Random Access Memory – оперативна пам’ять;

CPU – Central Processing Unit – центральний процесор;

Edge – Edge computing – обчислення на периферії мережі;

Fog computing – проміжний рівень обробки між хмарою та пристроями;

Dew computing – обробка даних безпосередньо на кінцевих пристроях;

IaaS – Infrastructure as a Service – інфраструктура як сервіс;

PaaS – Platform as a Service – платформа як сервіс;

SaaS – Software as a Service – програмне забезпечення як сервіс;

SCALE – Security, Cognition, Agility, Latency, Efficiency – принципи OpenFog Reference Architecture;

NSGA-III – Non-dominated Sorting Genetic Algorithm III – генетичний алгоритм з багатокритеріальним ранжуванням третього покоління;

MOEA/D – Multi-Objective Evolutionary Algorithm based on Decomposition – еволюційний алгоритм з декомпозицією задачі.

ВСТУП

Актуальність теми дослідження. Сучасний розвиток інформаційно-комунікаційних технологій супроводжується швидким зростанням обсягів даних, розширенням телекомунікаційної інфраструктури та підвищенням вимог до її надійності, масштабованості та безперебійної роботи. Особливу роль у цій сфері відіграють розподілені телекомунікаційні системи (РТС), які забезпечують обробку даних у децентралізованому середовищі з великою кількістю взаємопов’язаних вузлів. Зростаюча складність таких систем зумовлює потребу у нових методах керування обчислювальними процесами, здатних адаптуватися до змінних умов і обмежених ресурсів.

Особливої актуальності проблема набуває в умовах війни в Україні, де телекомунікаційна інфраструктура зазнає фізичних пошкоджень, а надійність, відмовостійкість і швидкість обробки даних мають особливе значення. РТС є основою побудови резервних і мобільних обчислювальних рішень, які можуть функціонувати незалежно від центральних серверів. Їхня ефективність визначається здатністю до динамічного перепризначення задач, швидкого формування кластерів вузлів та підтримки обчислювальних конвеєрів у динамічному телекомунікаційному середовищі.

Крім того, сучасні виклики в РТС, пов’язані з масовим впровадженням концепцій Fog/Edge computing, Інтернету речей (IoT) та інтелектуальних телекомунікаційних сервісів вимагають мінімізації затримок, ефективного використання обчислювальних ресурсів та забезпечення високого рівня безпеки. У таких умовах особливо важливими стають механізми самоорганізації обчислювальних вузлів, автоматизованого вибору координаторів кластерів, а також методи оптимальної маршрутизації та розподілу потоків даних у реальному часі.

Реалізація цих функцій потребує розробки нових адаптивних методів до кластеризації, оптимізаційного керування завантаженням вузлів та формування обчислювальних конвеєрів, що функціонують в умовах обмежених ресурсів і

динамічних змін топології. Застосування еволюційних методів дозволяє підвищити ефективність прийняття рішень у РТС, враховуючи численні критерії, зокрема пропускну здатність каналів, рівень навантаження, затримки та відмовостійкість.

Таким чином, вирішення **актуального науково-технічного завдання** підвищення ефективності та надійності обробки даних у РТС передбачає розробку нових методів і моделей, спрямованих на ієрархічну кластеризацію, динамічну самоорганізацію обчислювальних вузлів, визначення координаторів та побудову обчислювальних конвеєрів. Запропоновані рішення забезпечують зменшення затримок, збалансоване навантаження та стійкість роботи телекомунікаційної інфраструктури.

Зв’язок роботи з науковими програмами, планами, темами.

Тема дисертаційного дослідження відповідає предметної області ОНП «Телекомунікації та радіотехніка» третього (освітньо-наукового) рівня, оскільки охоплює теоретичні та прикладні засади побудови і експлуатації систем зв’язку, враховує технічне, програмне та інформаційне забезпечення з орієнтацією на підвищення ефективності функціонування телекомунікаційних систем, а також з використанням цифрових технологій, мікропроцесорних засобів і спеціалізованого програмного забезпечення.

Дисертаційна робота виконана на кафедрі «Транспортний зв’язок» Українського державного університету залізничного транспорту. Результати роботи впроваджено в навчальний процес Українського державного університету залізничного транспорту при викладанні дисциплін: «Телекомунікаційні системи передачі», «Комп’ютерно-інтегровані мережеві системи», «Системи керування в телекомунікаціях», «Хмарні мережеві технології систем залізничного транспорту» (Додаток Б).

Науково-практичні результати, отримані в ході дисертаційного дослідження, також застосовуються у службовій діяльність військової частини А7223 (Додаток Б).

Об’єктом дослідження є процес організації обробки даних у РТС.

Предметом дослідження є методи, моделі та алгоритми ієрархічної кластеризації та динамічної самоорганізації вузлів у РТС, спрямовані на підвищення їх ефективності, надійності та адаптивності до змінних умов.

Метою дисертаційного дослідження є підвищення ефективності та надійності обробки даних у РТС шляхом розробки методів і моделей ієрархічної кластеризації, багатокритеріального розподілу потоків даних та динамічної самоорганізації обчислювальних вузлів в умовах змінного середовища.

Згідно з метою дисертаційної роботи визначено **часткові наукові задачі**, які повністю відповідають структурі дослідження та очікуваним науковим результатам.

1. Провести аналіз сучасних архітектур та методів розподілу навантаження у РТС, виявити чинники, які знижують ефективність управління обчислювальними процесами, та сформулювати вимоги до розробки адаптивних методів ієрархічної кластеризації, динамічної самоорганізації та багатокритеріальної оптимізації розподілу потоків.

2. Розробити комплексний метод самоорганізації РТС, що враховує особливості мережевої топології і продуктивність вузлів; обґрунтувати алгоритм реалізації методу та створити до нього програмні рішення.

3. Удосконалити метод багатокрокової ієрархічної кластеризації на основі алгоритму Лувена за рахунок впровадження рекурсивного механізму з адаптивним параметром роздільної здатності; розробити відповідний алгоритм реалізації і експериментально дослідити його ефективність в умовах змін навантаження.

4. У межах розробленого інтегрованого методу керування інформаційними потоками удосконалити метод вибору головного вузла в РТС на основі розробки модифікованого варіанту алгоритму «Пліткар» (Gossip) з ресурсно-затримковою оцінкою вузлів і асинхронно-узгодженим вибором координатора, що забезпечує підвищену адаптивність та стійкість до відмов системи.

5. Розробити метод формування обчислювальних конвеєрів у кластеризованому середовищі, який забезпечує поетапну обробку потоків даних,

оптимальне використання ресурсів кластерів, балансування навантаження та мінімізацію затримок на основі модифікованого алгоритму побудови маршрутів.

6. Розробити алгоритм реалізації методу формування обчислювальних конвеєрів на основі спрямованого багатошарового графа з урахуванням міжетапних затримок, ресурсних обмежень та штрафів за перевантаження; створити програмну модель для забезпечення паралельної маршрутизації потоків та експериментального аналізу ефективності адаптації до змінного навантаження у РТС.

7. Удосконалити метод багатокритеріального розподілу потоків у РТС за рахунок модифікованого генетичного алгоритму NSGA-III з гібридним ранжуванням і адаптивним регулюванням частоти мутацій в залежності від прогнозованого навантаження, і, на цій основі, обґрунтувати доцільність використання розподілених баз даних у якості платформи зберігання результатів обробки обчислювальних конвеєрів.

Методи дослідження. Для вирішення задач, поставлених у дисертації, було застосовано: методи оптимізації – для побудови та вдосконалення алгоритмів кластеризації, вибору координаторів і розподілу обчислювального навантаження; теорії ймовірностей – для моделювання поведінки вузлів та процесів у РТС; теорії інформації – для оцінки ефективності передачі та обробки даних в умовах обмежених ресурсів і динамічних змін мережевих характеристик; методи імітаційного моделювання – для верифікації працездатності запропонованих методів і алгоритмів у змодельованих умовах; статистичні методи – для аналізу результатів моделювання, оцінки відмовостійкості, затримок і збалансованості навантаження; методи еволюційного моделювання – для реалізації модифікованого генетичного алгоритму NSGA-III з гібридним ранжуванням і адаптивним керуванням параметрами; а також методи математичного моделювання топології та структур даних – для формалізації процесів самоорганізації обчислювальних вузлів і побудови обчислювальних конвеєрів у РТС.

Комплексне застосування зазначених методів забезпечило наукову

обґрунтованість і достовірність отриманих результатів, підтверджених моделюванням та апробацією запропонованих рішень.

Наукова новизна отриманих результатів. Базується на запропонованих методах, які забезпечують підвищення ефективності функціонування РТС, за допомогою яких було отримано такі наступні наукові результати:

1. **Вперше** розроблено комплексний метод самоорганізації розподілених телекомунікаційних систем, який, на відміну від традиційних підходів, враховує продуктивність вузлів, топологію мережі, затримки та пропускну здатність каналів передачі, що досягається за рахунок адаптивного налаштування параметрів кластеризації та багаторівневого аналізу мережевих зв’язків.

2. **Вперше** розроблено інтегрований метод керування інформаційними потоками в умовах нестабільної мережі із забезпеченням вибору головного вузла та одночасного формування обчислювальних конвеєрів у розподілених телекомунікаційних системах, шляхом поєднання ресурсно-затримкової оцінки характеристик вузлів та модифікованого алгоритму з асинхронно-узгодженим вибором керуючого вузла, що забезпечує мінімізацію часових затримок.

3. **Удосконалено** метод багатокрокової ієрархічної кластеризації на основі застосування модифікованого алгоритму Лувена, який, на відміну від відомих підходів, забезпечує контроль розміру та щільності кластерів завдяки динамічному регулюванню параметра роздільної здатності та рекурсивному уточненню кластерної структури.

4. **Удосконалено** метод багатокритеріального розподілу потоків обробки даних у телекомунікаційних системах на основі модифікованого генетичного алгоритму NSGA-III, з метою забезпечення підвищення точності розподілу та зниження числа делегацій між кластерами, шляхом використання механізму гібридного ранжування та адаптивного регулювання частоти мутацій в залежності від прогнозованого навантаження.

Практичні результати, отримані в дисертаційній роботі, полягають у розробці та впровадженні запропонованих методів у вигляді технічних рішень та алгоритмів для підвищення ефективності та надійності РТС, а саме:

– на основі порівняльного аналізу сучасних архітектур і методів обробки даних у РТС обґрунтовано доцільність розробки нових адаптивних методів та алгоритмів кластеризації, самоорганізації та багатокритеріальної оптимізації, що враховують динаміку навантаження, структурну складність і параметри мережевих з'єднань, включаючи затримки, пропускну здатність та продуктивність вузлів;

– розроблено алгоритм і програмну реалізацію методу самоорганізації РТС, що дозволяє знизити затримки в обробці та передачі даних до 17 % в порівнянні з методами Лувена і Лейдена та одночасно покращити узгодженість структури кластерів, що, в свою чергу, зменшує витрати ресурсів на маршрутизацію на 10–15 % залежно від топології мережі і навантаження;

– удосконалено алгоритм Лувена шляхом формування кластерів з адаптивною глибиною, що усуває надмірну агрегацію, забезпечує рівномірне навантаження вузлів і зменшує стандартне відхилення розмірів кластерів на 53 – 66 %, що необхідно для стабільної роботи РТС в умовах змінної топології;

– в межах інтегрованого методу керування інформаційними потоками удосконалено алгоритм вибору головного вузла, за рахунок застосування асинхронно-узгодженого вибору координатора, що скорочує час відновлення після відмови головного вузла на 43 % та знижує службовий трафік до 30 %, завдяки виключенню явної фази виборів і використанню наперед сформованих списків резервних кандидатів;

– розроблено алгоритм та програмну реалізацію методу формування обчислювальних конвеєрів для обробки потокових даних, що у поєднанні з удосконаленим алгоритмом вибору координатора забезпечує динамічну самоорганізацію мережі зі скороченням затримок на 5–7%, зменшенням кількості перевантажень до 20% і підвищенням рівномірності завантаження ресурсів вузлів на 38–54%;

– розроблено модифікований алгоритм формування обчислювальних конвеєрів на основі спрямованого багатошарового графа з параметричним урахуванням затримок, ресурсних обмежень та штрафів за перевантаження, що

забезпечує паралельну маршрутизацію потоків і адаптацію до змінних умов навантаження в РТС;

– реалізовано модифікований варіант генетичного алгоритму NSGA-III, який поєднує гібридне ранжування і адаптивне регулювання частоти мутацій, що забезпечує зменшення середнього перевантаження вузлів на 53,7%, скорочення пікових перевантажень втричі та підвищення рівномірності завантаження ресурсів у 2,1 рази. Обґрунтовано доцільність використання розподілених баз даних у якості платформи зберігання результатів обробки обчислювальних конвеєрів.

Отримані в дисертаційній роботі науково-практичні результати впроваджено в службову діяльність військової частини А7223, що підтверджується Актом впровадження, наведеним у додатку Б.

Особистий внесок здобувача Особистий внесок здобувача полягає у формулюванні наукової концепції дослідження, розробці методів, алгоритмів і технічних рішень, спрямованих на підвищення ефективності обробки даних у РТС. Здобувачем самостійно здійснено моделювання, алгоритмічну реалізацію і верифікацію запропонованих методів, а також аналіз отриманих результатів.

У публікаціях, виконаних у співавторстві, здобувачу належать такі результати: [1] – здійснено формалізацію логіки побудови NoSQL-баз для реалізації структури даних, що підтримують обчислювальні процеси у РТС; [2] – проведено системний аналіз архітектур і методів організації обчислень у розподілених телекомунікаційних середовищах із застосуванням Edge-підходу; [3] – запропоновано метод ієрархічної кластеризації обчислювальних вузлів у РТС на основі графових алгоритмів із урахуванням затримок і пропускної здатності; [4] – розроблено метод багатокритеріального розподілу потоків обробки даних із використанням модифікованого генетичного алгоритму NSGA-III; [5] – удосконалено алгоритм вибору головного вузла в РТС та метод формування обчислювальних конвеєрів з урахуванням динаміки мережі та відмовостійкості; [6] – проведено аналіз сучасних NoSQL баз даних і визначено особливості їх логічного проектування для обробки складноструктурованих

даних у РТС; [7] – проаналізовано переваги архітектур периферійних обчислень для підвищення відмовостійкості та масштабованості телекомунікаційної інфраструктури; [8] – здійснено класифікацію методів розподілу навантаження в РТС та запропоновано напрямки їх удосконалення для адаптації до динамічних умов; [9] – представлено методичні підходи до перебудови архітектури РТС з метою підвищення їх ефективності в умовах обмежених ресурсів; [10] – розроблено концепцію оптимізації мережевих маршрутів обробки потокових даних у розподіленому середовищі з урахуванням характеристик вузлів; [11] – запропоновано підходи до формування топології та маршрутизації у розподілених телекомунікаційних мережах, що орієнтовані на підвищення гнучкості і адаптивності інфраструктури в умовах змін середовища; [12] – запропоновано метод оптимізації топології розподіленої телекомунікаційної системи для підвищення ефективності обробки потокових даних за рахунок покращення структури взаємодії між вузлами; [13] – розроблено підхід до багатокритеріального розподілу інформаційних потоків у РТС на основі модифікованого генетичного алгоритму, що враховує динаміку навантажень і обмеженість ресурсів; в [14] проведено аналіз методів логічного моделювання NOSQL баз даних для ігрових серверів.

Апробація результатів. Основні наукові положення і результати дослідження апробовано на науково-практичних конференціях.

– 36 Міжнародна науково-практична конференція «Інформаційно-керуючі системи на залізничному транспорті». Харків: УкрДУЗТ, 16-17 листопада 2023;

– 37 міжнародна науково-практична конференція «Інформаційно-керуючі системи на залізничному транспорті». Харків: УкрДУЗТ, 10-11 жовтня 2024;

– XII міжнародна науково-практична конференція «Людина, суспільство, комунікативні технології» Харків: УкрДУЗТ, 25 жовтня 2024;

– І міжвузівська наукова конференція *«Стан та перспективи розвитку інфокомунікацій в сучасних умовах»*. Харківський національний університет повітряних сил ім. І. Кожедуба. Харків: 22 листопада 2024;

– Міжнародна науково-технічна конференція «Перспективи розвитку озброєння та військової техніки сухопутних військ». Національна академія сухопутних військ імені гетьмана Петра Сагайдачного Львів 15-16 травня 2024;

– Науково-практична конференція «Сектор безпеки і оборони України на захисті національних інтересів: актуальні проблеми та завдання в умовах воєнного стану». Національна академія Державної прикордонної служби України імені Богдана Хмельницького, 22-23 листопада 2024;

– XXI Міжнародна наукова конференція «Новітні технології для захисту повітряного простору». Харківський національний університет повітряних сил ім. І. Кожедуба. Харків: 9 – 10 квітня 2025 року.

– 12 Міжнародна науково-технічна конференція «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління», 27-28 квітня 2022 року.

– Міжнародна науково-технічна конференція «Перспективи розвитку озброєння та військової техніки Сухопутних військ», 14-15 травня 2025 рік.

Публікації. Матеріали дисертаційного дослідження викладено у 14 наукових публікаціях автора, з яких 5 статей опубліковано у фахових наукових виданнях України, а 8 – у збірниках матеріалів міжнародних науково-практичних конференцій. Публікації відображають основні етапи, результати та наукові положення дисертаційної роботи.

Структура і обсяг дисертації. Дисертаційна робота складається з анотації двома мовами, вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Робота містить 179 сторінок основного тексту, у тому числі, 39 рисунків, 22 таблиці, 159 найменувань у списку використаних джерел, 4 Додатків (А-Г). Загальний обсяг дисертаційної роботи займає 196 сторінок.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ОРГАНІЗАЦІЇ ОБРОБКИ ДАНИХ У РОЗПОДІЛЕНИХ ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ

1.1 Аналіз сучасних архітектур для оптимізації серверних та телекомунікаційних систем

Сучасні телекомунікаційні системи, які опрацьовують великі обсяги даних і вимагають мінімальних затримок, стикаються з необхідністю впровадження спеціалізованих архітектур, що дозволяють оптимально використовувати ресурси, підвищувати якість обслуговування та зменшувати затримки.

Упродовж останніх років проведено значну кількість досліджень, присвячених питанням розвитку архітектур РТС, зокрема в напрямку використання Fog Computing та Edge Computing. У роботах [1, 12, 25, 32] розглядаються принципи побудови ефективного середовища обробки потокових даних з динамічним балансуванням навантаження та використанням контейнерних технологій. Праці [51, 110, 135] акцентують увагу на архітектурних і технологічних перевагах Fog Computing, серед яких – зменшення затримок, оптимізація обміну та зниження навантаження на центральні вузли.

Дослідження [34, 54, 85] аналізують інтеграцію Fog – і Cloud – рівнів у середовищах Smart City та системах відеоаналітики, що особливо актуально для задач у реальному часі. Автори [3, 73, 148] звертають увагу на питання кластеризації, вибору координаторів та оптимізації топологій розподілених обчислень. У роботах [2, 13, 14, 23] доведено складність забезпечення QoS і управління ресурсами в умовах динаміки трафіку та гетерогенності середовища.

Також окрему увагу приділено підходам до самоконфігурації архітектур з використанням SDN/NFV [36, 125, 141] і впровадженню еволюційних механізмів оптимізації у хмарному й туманному середовищах [33, 84, 119]. Підходи до самоорганізації та масштабованості в інфраструктурних рішеннях аналізуються в [5, 11, 145], а у [6, 18, 92] запропоновано моделі, що враховують неповну

інформацію, непередбачувані навантаження та необхідність адаптації в умовах невизначеності.

Таким чином, наявна наукова база підтверджує актуальність подальших досліджень архітектур Fog та Edge Computing у контексті розподілених телекомунікаційних систем, які поєднують низькі затримки, децентралізовану обробку, адаптивне управління ресурсами та підтримку високонавантажених сценаріїв.

В табл. 1.1 представлений еволюційний прогрес архітектур розподілених телекомунікаційних систем, з відображенням ключових етапів розвитку та впровадження інноваційних рішень.

Впровадження архітектур Fog Computing, що розміщує ресурси та сервіси обчислень ближче до кінцевих користувачів або джерел даних, є необхідною умовою для створення більш розподілених, ефективних та масштабованих систем. Це дозволяє знизити навантаження на центральні обчислювальні центри, зменшити затримку в обробці даних та покращити загальну швидкість реакції системи на запити користувачів, що є критично важливим для застосунків Інтернету речей (IoT), мобільних додатків та інших сучасних технологій, які потребують швидкого та надійного обміну даними [110, 135].

Протягом останніх років доступність хмарних технологій призвела до широкого розповсюдження інтеграції Cloud Computing у серверні системи, що докорінно змінило парадигму інфраструктури та обчислювальних середовищ у бізнес- та технологічних сферах, у тому числі в галузі телекомунікацій. Популяризація цієї парадигми та її широке розповсюдження відбулося у першу чергу завдяки практично необмеженому розширенню ресурсів серверних систем за рахунок віртуалізації всіх компонентів [3, 56].

Таблиця 1.1 – Аналіз сучасних архітектур розподілених систем

Етап	Характеристика	Переваги	Недоліки
Дворівнева Cloud Computing архітектура [51, 52, 110]	Складається з хмари та периферійного обчислювального шару, який обробляє дані ближче до джерела.	Зниження затримки, ефективне використання ресурсів.	Складність управління, потреба в координації між рівнями.
N-рівнева архітектура OpenFog [51, 110, 152]	Розподілена архітектура, яка розширює хмарні послуги до периферійних пристроїв у мережі.	Гнучкість та масштабованість, підвищена безпека та надійність	Ускладнення мережі, вищі вимоги до широкосмугового зв’язку.
Cloudlets – архітектура [75, 111, 134]	Міні-хмари з розміщенням «edge computing» для підвищення продуктивності мобільних додатків.	Зменшення затримки, покращення відгуку додатків.	Обмежена мобільність, можлива нерівномірність покриття.
MELINDA [75, 111]	Система управління даними для туманних обчислень, що забезпечує координацію між різними вузлами.	Краща синхронізація даних, оптимізація ресурсів.	Висока складність системи, потреба в сильній координації.
Архітектура системи з використанням SDNFV [36, 125, 141]	Інтеграція програмно-визначеної мережі (SDN) та віртуалізації мережевих функцій (NFV) для гнучкого управління мережевими ресурсами.	Висока адаптивність до змін у мережевому трафіку, зниження витрат на апаратне забезпечення.	Складність реалізації та управління, потреба в кваліфікованих IT-фахівцях.

Щодо архітектурних рішень у контексті серверних систем з використанням хмарних технологій, популярність інтеграції Cloud-рішень призвела до формування конкретних шаблонів і типових архітектур, які застосовуються для організації таких систем.

Дворівнева Cloud Computing архітектура

Одним із перших кроків у розвитку архітектур розподілених систем стало впровадження дворівневої Cloud Computing архітектури, що стало фундаментом для підвищення ефективності та забезпечення відповіді в реальному часі. Ця модель передбачає створення серверної системи, розділеної на два рівні: передній (frontend), який взаємодіє з користувачами та обробляє їх запити, та задній (backend), що займається обробкою даних і виконанням обчислень [51, 52, 110]. Дворівнева структура стала широко прийнятим підходом, оскільки вона сприяє стандартизації процесів та ефективному розподілу функцій, забезпечуючи при цьому масштабованість системи (рис. 1.1).

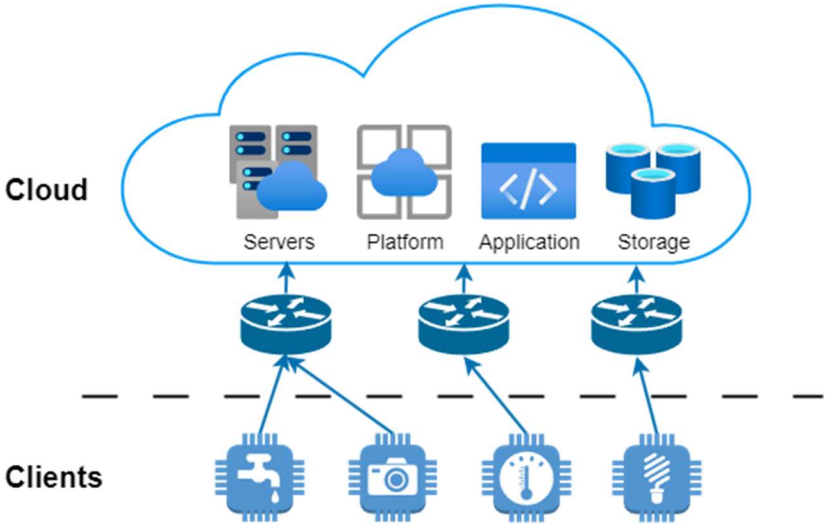


Рисунок 1.1 – Дворівнева Cloud Computing архітектура

Архітектура, подібна до зображеної на рисунку 1.1, відповідає потребам традиційних клієнт-серверних застосунків, де користувачі-клієнти взаємодіють з централізованими серверами. Проте, у контексті Інтернету речей (IoT), ця

модель може виявитись недостатньо ефективною через ряд специфічних викликів, таких як.

1. Пропускна здатність мережі. Велика кількість підключених IoT-пристроїв, які постійно генерують дані, може створити значні проблеми з пропускною здатністю хмарної мережі, призвести до перевантаження і зниження якості обслуговування.

2. Затримка. Відстань між IoT-пристроями та хмарними серверами може викликати затримки у обміні даними, що критично для застосунків, які потребують миттєвої реакції, наприклад, у системах безпеки або медичних пристроях.

3. Безпека даних. Збільшена кількість підключених пристроїв створює можливості для кібератак і порушення конфіденційності даних.

4. Скальованість і адміністрування. При наявності великої кількості підключених пристроїв, виникають проблеми з адмініструванням і керуванням ними.

5. Забезпечення життєздатності пристроїв. Низка IoT-пристроїв мають обмежені ресурси, такі як батареї. Забезпечення тривалої роботи інтернету речей і забезпечення їх надійності вимагає розробки ефективних стратегій керування енергоспоживанням та моніторингу стану пристроїв.

6. Сумісність і стандартизація. Різноманітність протоколів та стандартів, що використовуються різними виробниками, може ускладнити інтеграцію пристроїв та систем.

Для розв’язання визначених задач була сформована парадигма Edge Computing з метою перенесення частини обчислень (функціоналу) на вузли, що знаходяться ближче до пристроїв ніж хмара. Водночас, обчислювальним вузом може бути не тільки датацентр, а будь-які пристрої з можливостями обчислення [52, 110, 135].

Виникнення Edge Computing задало загальну концепцію подібних систем, що сприяло, згодом, розробці нових архітектур.

1.2 Адаптивні архітектури для обробки даних у середовищах Fog, Edge та Dew Computing

Fog Computing є концепцією архітектури, в якій між шаром хмари та шаром пристроїв додається додатковий шар вузлів для обробки. Часто згадується як синонім Edge Computing і розглядається в залежності від інтерпретації: і як додатковий рівень хмарного шару, і як додатковий рівень поруч із пристроями.

Dew Computing є концепцією мікросервісного характеру, яка втілюється у платформі, де пристрої можуть взаємодіяти між собою безперервно в межах єдиної «локальної» мережі, і ця взаємодія відбувається без пересилання даних до хмарних ресурсів. Одним із прикладів використання цієї концепціє є смарт-пристрої.

Fog-Dew Computing є синтезом вищезгаданих архітектур, що використовує їх основні переваги: пристрої функціонують як автономні та не потребують постійного з'єднання з Інтернетом (з хмарою), проте підключені до локального сервера. Проте локальний сервер взаємодіє з хмарними ресурсами та відповідає за надання послуг пристроям.

На сьогоднішній день, згідно з статистики Google Scholar, Scopus і Web of Science, Fog Computing привертає найбільшу увагу дослідників, внаслідок своєї універсальності та придатності до застосування в різних сферах Інтернету речей (IoT) [3, 110].

Зауважимо, що концепція Dew Computing є обмеженою, з можливостями для застосування лише у визначених розподілених телекомунікаційних системах. Тому подальше дослідження доцільно зосередити на архітектурах Fog Computing.

Під час аналізу архітектур слід дотримуватися чіткого визначення понять. Важливо розрізняти поняття шар (layer) і рівень (tier), які на практиці використовуються як синоніми, але вони мають значну відмінність. Шар - це спосіб логічного структурування компонентів програмного рішення, тоді як рівень - спосіб фізичного структурування інфраструктури [135].

OpenFog N-tier architecture

У 2017 році, Прінстонський університет і деякі провідні IT компанії з різних галузей сформували консорціум OpenFog. Результатом колаборації став документ OpenFog Reference Achitecture, на основі якого був сформований протокол IEEE 1934-2018 (Institute of Electrical and Electronics Engineers). Даний документ представляє собою нову модель сервісної архітектури - FaaS (Fog as a Service), яка включає раніше відомі: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), але з адаптацією під Fog Computing, а також способи розширення [51, 110, 152].

У порівнянні з іншими архітектурами, OpenFog визначає свої переваги через використання абревіатури SCALE, яка розшифровується як.

1. Security - додаткові способи досягнення безпеки даних.

2. Cognition - забезпечення автономності за рахунок розуміння цілей клієнтів системи.

3. Agility - забезпечення швидкого і доступного масштабування.

4. Latency - зниження затримки для забезпечення обробки в реальному часі.

5. Efficiency - динамічний розподіл ресурсів системи для досягнення максимальної ефективності.

Архітектура ґрунтується на 8 основних принципах, які називають «стовпами»: Security, Scalability, Openness, Autonomy, Programmability, RAS (Reliability, Availability, Serviceability), Agility і Hierarchy. Опис кожного стовпа є набором рекомендацій і вимог до системи. На рис. 1.4 представлена архітектура OpenFog з одним рівнем Fog вузлів.

Як видно з рис. 1.2, архітектура OpenFog не встановлює жорстких обмежень щодо кількості рівнів у структурі Fog-вузлів. Це означає, що архітектура є гнучкою та масштабованою, дозволяючи інженерам і архітекторам систем адаптувати її під конкретні прикладні сценарії, галузі чи географічні особливості. Наприклад, у системах Smart City доцільно реалізовувати більше рівнів обробки для забезпечення локального реагування, тоді як у промислових

системах реального часу можлива спрощена дворівнева конфігурація для мінімізації затримок.

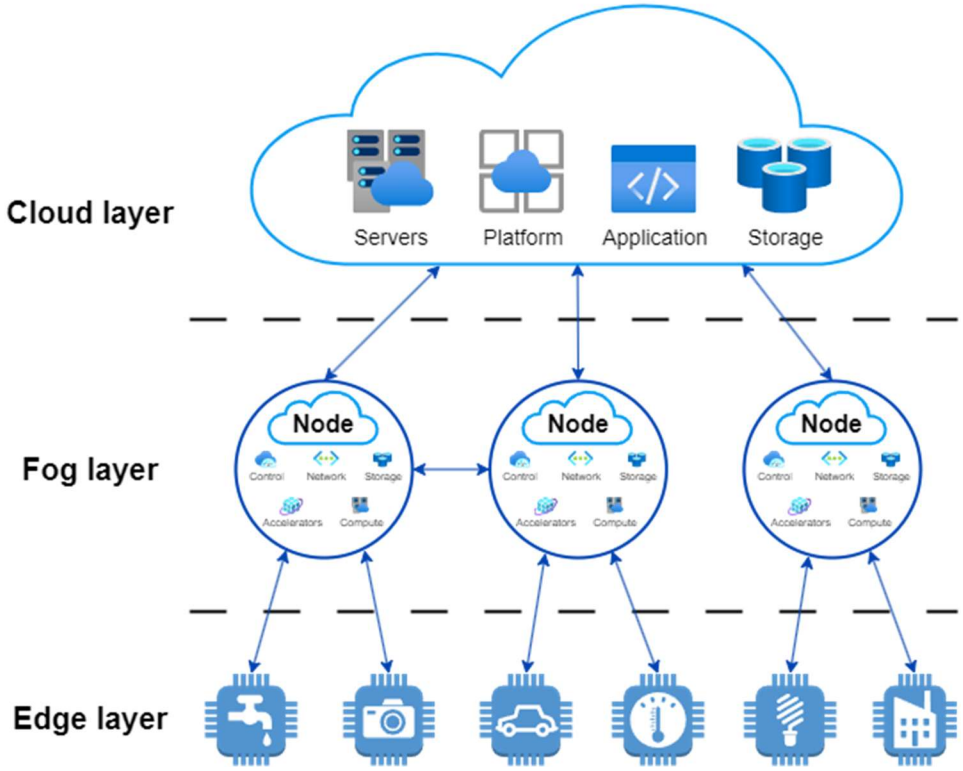


Рисунок 1.2 – N-рівнева архітектура OpenFog

У зв’язку з цим, кількість рівнів хмарних або туманних вузлів може бути довільною, залежно від складності задач, що вирішуються, та вимог до продуктивності. Крім того, архітектура допускає як повну реалізацію всіх трьох рівнів (cloud, fog, edge), так і часткову, а саме з пропусканням одного з них, якщо це виправдано з точки зору ефективності або витрат. Наявність хмарного та туманного шарів є необов’язковою умовою, що підкреслює універсальність архітектури OpenFog та її здатність підтримувати як централізовану, так і децентралізовану моделі обробки даних.

Але на сьогоднішній день при високому рівні стандартизації і описання цієї архітектури, її рідко застосовують на практиці внаслідок відсутності чіткого спрямування на певну галузь. Як результат, основна увага актуальних науково-практичних досліджень приділяється архітектурам Fog Computing, які є більш адаптованими до конкретних сфер застосування [51, 110, 152].

Cloudlets

Дослідження архітектури OpenFog довели, що однією з основних проблем під час розробки систем Fog Computing є визначення оптимальної кількості рівнів Fog вузлів, їх розташування та виділення ресурсів для їх функціонування. Така галузь як Smart City традиційно використовувала централізований підхід до організації системи і даних за допомогою технологій Cloud Computing. Але, такі проблеми як: захист даних, збільшені вимоги до затримки та енергоефективність, стали підставою для розгляду децентралізованих архітектур [75, 111, 134].

У відповідь на ці виклики, галузі, що активно інтегрують IoT, у тому числі, телекомунікаційні системи, почали адаптувати концепції розподілених архітектур. В контексті Smart City, це призвело до виникнення нових архітектурних рішень, які передбачають територіальний поділ ресурсів і обчислювальних потужностей. Це дозволяє реалізувати більш ефективний обмін даними і керування ресурсами, а також забезпечити необхідний рівень автономії для різних частин міської інфраструктури [111].

Зокрема, у містах з високим рівнем розвитку та щільністю населення, децентралізовані системи дозволяють знизити навантаження на центральні сервери, розосереджуючи процес обробки даних на багаточисельні локальні вузли. Це, у свою чергу, веде до покращення реакції систем на локальні події та забезпечення більшої стійкості до зовнішніх перебоїв та атак, збільшуючи загальну надійність інфраструктури Smart City.

На рис. 1.3 представлена ієрархічна багаторівнева архітектура, що втілює принципи розподілу ресурсів обчислень за територіальним принципом. У цій моделі обчислювальні вузли розподілені відповідно до географічного розташування, що оптимізує процеси обробки даних та знижує затримки, властиві для централізованих систем [145].

Для розподілу архітектури системи за територіальною ознакою введено три концептуальні рівні: micro (охоплює інфраструктуру на рівні окремої будівлі або об’єкта), meso (відповідає за обробку даних на рівні району або кластеру

пристроїв) та macro (реалізує централізовану обробку на рівні міста або хмарного середовища).

Крім того, у межах цієї архітектури використовується поняття cloudlet – як невеликого локального датацентру, розміщеного безпосередньо поблизу кінцевих користувачів. На відміну від традиційної хмари, cloudlet забезпечує обробку даних з мінімальними затримками завдяки географічній близькості до джерел трафіку, що є важливим для latency-чутливих застосунків, зокрема в IoT, Smart City та мобільних сервісах.

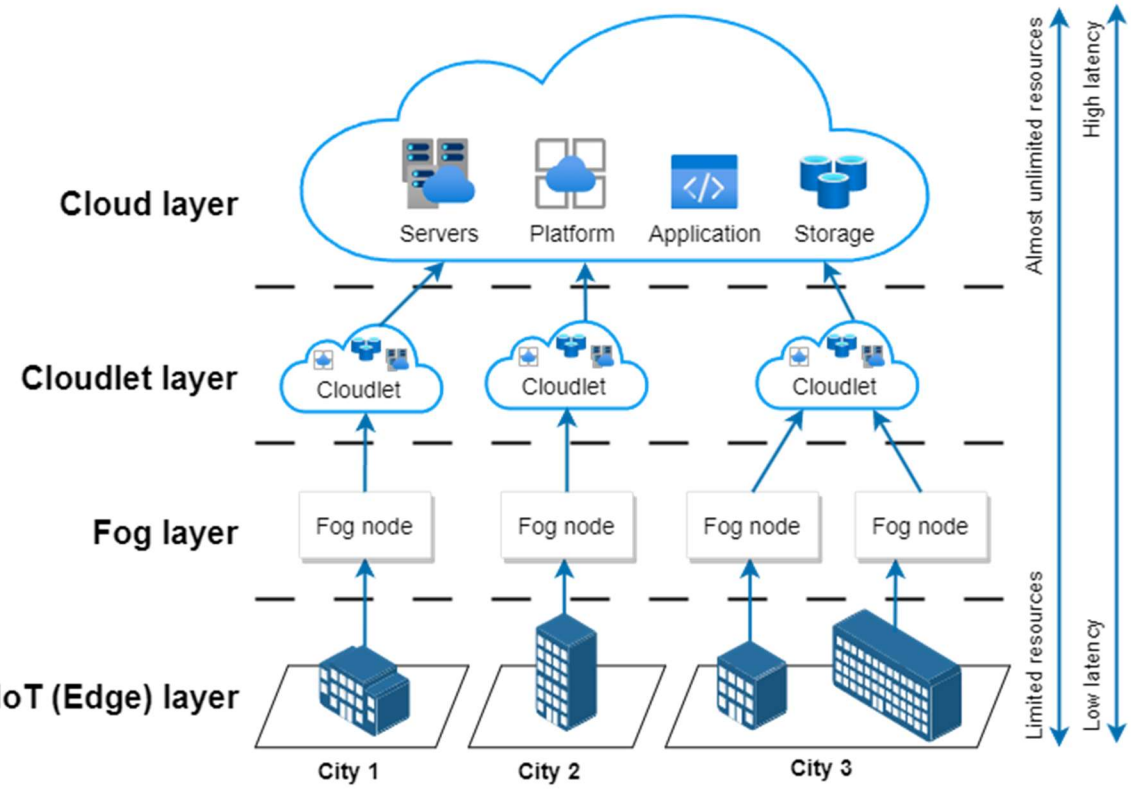


Рисунок 1.3 – Графічне представлення F2c2C (Cloudlet) архітектури на прикладі системи Smart City

Оскільки ця архітектура фокусується на управлінні даними, запроваджується три види даних відповідно до їх віку:

1. Дані в реальному часі: виробляються пристроями і вузлами на micro і meso рівнях у місцях, де потрібна мінімальна затримка.

2. Останні дані: генеруються на meso рівні, є результатом обробки даних у реальному часі.

3. Історичні дані: дані, які зберігаються в хмарі (macro рівні).

Керуючись цими поняттями, можна визначити три основні шари цієї архітектури (крім шару пристроїв).

1. Туманний шар: максимально близький до пристроїв, працює з даними в реальному часі, перебуває на micro і meso рівнях.

2. Шар cloutlet-вузлів: проміжний шар між хмарою і туманом, перебуває в тому самому місті, що й пристрої (macro рівень), виконує завдання з опрацювання «найближчих» даних.

3. Хмарний шар: обробляє та зберігає історичні дані, має необмежену кількість ресурсів.

Таким чином, ця архітектура поєднує в собі переваги централізованих і децентралізованих архітектур: робота в гетерогенному IoT-середовищі, низьке навантаження на мережу хмари, можливість оброблення критичних даних у реальному часі тощо.

MELINDA

Одними з найскладніших систем у сфері телекомунікацій є системи відеомоніторингу з виявленням об’єктів у реальному часі. Традиційний підхід з використанням Cloud Computing передбачає передачу необробленого відеопотоку в хмарні датацентри, де відбувається його обробка з подальшим передаванням відповіді на клієнт. Такий підхід має серйозні недоліки, пов'язані з інфраструктурою, а саме.

1. Висока насиченість мережі (кожна Full-HD камера генерує відеопотік до 12 Мбіт/с), що створює проблеми в масштабуванні системи у формі обмеженої пропускної здатності мережі хмари.

2. Висока та нестабільна затримка під час передачі даних від клієнта до хмари.

3. Високе ресурсо- та енергоспоживання, викликане необхідністю зберігання великого обсягу малокорисних даних (фреймів відеопотоку без об'єктів або без змін).

Для вирішення цих проблем, Neto A.R. [111] запропонував трирівневу Fog Computing архітектуру, яка була адаптована та графічно реалізована в даному дослідженні відповідно до умов (рис. 1.4) [145].

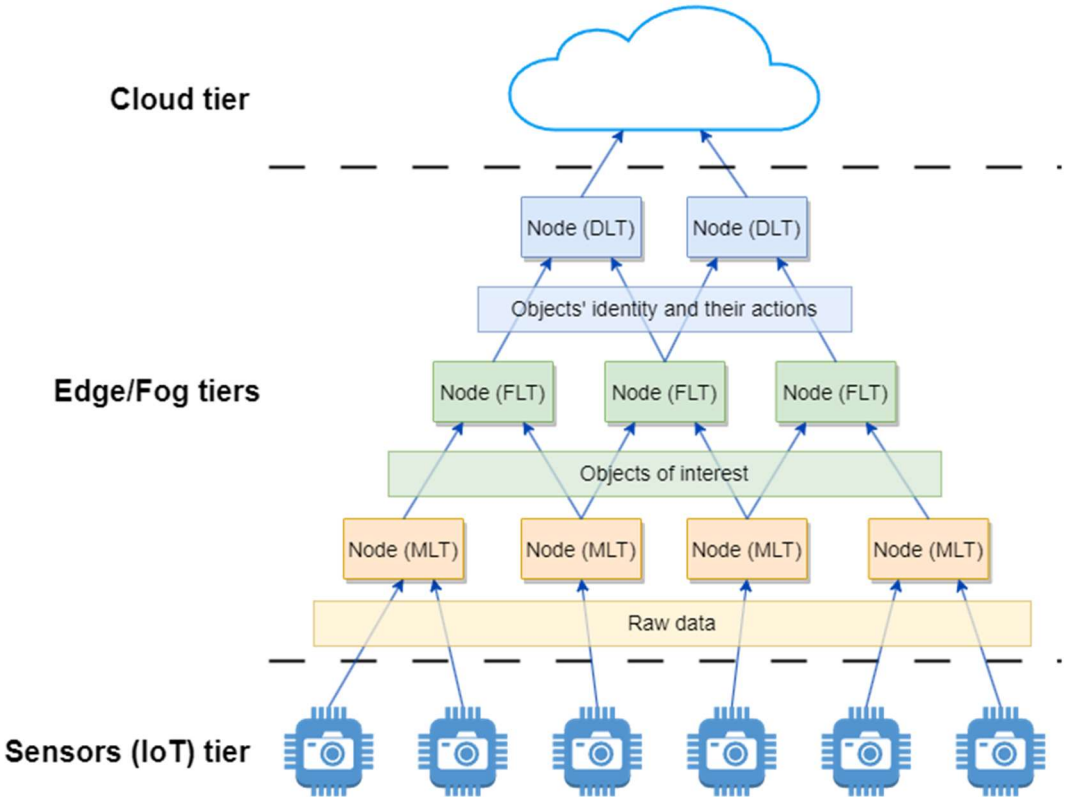


Рисунок 1.4 – 3-рівнева Fog Computing архітектура, оптимізована під обробку відеопотоку

З рис. 1.4 можна зробити висновок, що основний принцип роботи цієї архітектури полягає в зниженні обсягу даних шляхом багатоетапного оброблення відеопотоку. Обробка відеопотоку складається з трьох кроків:

– фільтрація відеопотоку з виділенням тільки тих кадрів, які можуть містити об'єкт;

– ідентифікація об'єкта;

– інтерпретація появи об’єкта (прив’язування до певної події) для прийняття рішень згодом.

Відповідно, архітектура містить три види обробних вузлів, які виконують завдання різного рівня: Measurement Level Task (MLT), Feature Level Task (FLT) і Decision Level Task (DLT).

Водночас, така інфраструктура підтримується програмною архітектурою MELINDA (Multilevel Information Distributed Processing Architecture), яка складається з двох підсистем.

1. Processing Subsystem, яка складається з вузлів для моніторингу, виділення ресурсів і обробки.

2. Management Subsystem, яка складається з вузлів для обробки запитів кінцевих користувачів (вилучення даних) і компонентів для високорівневого моніторингу системи.

Для зв'язування використовується компонент Data Communication Manager, спільний для обох підсистем. Передбачається, що кожен із компонентів розміщується на окремому вузлі, а самих компонентів одного типу може бути декілька.

SDN/NFV

Наведені вище архітектури розглядають Fog Computing здебільшого з концептуальної та програмної точок зору, ґрунтуючись на традиційних підходах до мережевої взаємодії. В основі цих рішень зазвичай лежать деревоподібні структури, побудовані на базі Ethernet-маршрутизаторів та базових станцій мобільного зв’язку. Водночас, сучасні концепції для IoT дедалі частіше орієнтовані на впровадження новітніх технологій, зокрема 5G – як основи для високошвидкісної передачі даних, а також SDN (Software Defined Networking) та NFV (Network Functions Virtualization) – для гнучкої організації мережевих функцій та управління ресурсами [36, 125, 141].

Використання 5G спрямоване на зменшення затримок, зниження вартості та енергоспоживання пристроїв, а також відкриває можливості для реалізації

нових типів IoT-систем, що раніше були складними або неможливими в реалізації.

Зі свого боку, технології SDN і NFV дозволяють переосмислити підходи до побудови Fog Computing-інфраструктур, забезпечуючи централізоване логічне управління трафіком, динамічне масштабування та адаптивне балансування навантаження.

На рис. 1.5 представлено архітектуру системи з використанням Software-Defined NFV на шарі з Fog-вузлами, адаптовану до умов даного дослідження [145].

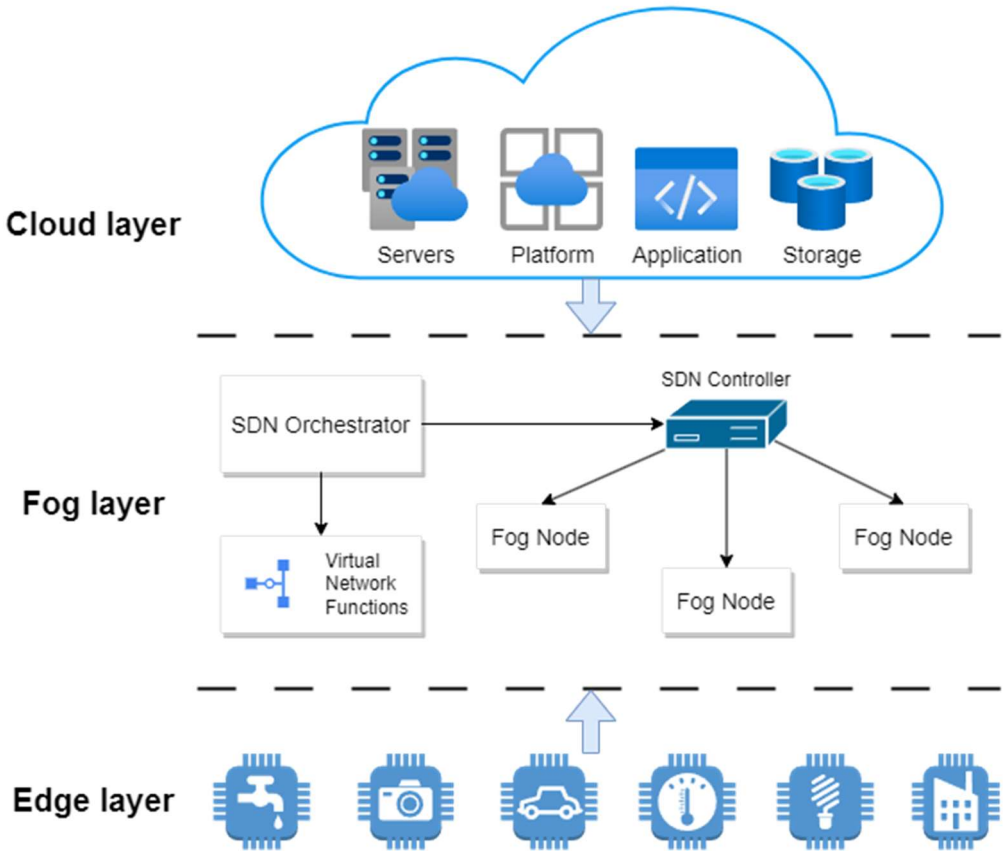


Рисунок 1.5 – Архітектура системи з використанням SDN/NFV на шарі з Fog вузлами

З рис. 1.5 видно, що основою туманного шару є SDN-контролер, який відповідає за обробку всього мережевого трафіку. Передбачається, що цей

контролер є проміжною ланкою між оброблювальними вузлами та клієнтами, і він керується 4 видами дій, а саме: створення, модифікація, виконання і завершення завдання.

Таким чином, коли клієнт запитує виконання завдання в оркестратора, він може перевірити валідність цього запиту і запросити певну кількість ресурсів на виконання цього завдання.

Крім того, треба звернути увагу, що під час виконання задач у системі беруть до уваги 5 індексів.

1. Індекс вартості – відстань від джерела до вузла. Чим менша, тим краща архітектура.
2. Індекс часу – сума витраченого часу на пересилання даних і часу виконання завдання. Чим менше, тим краще.
3. Індекс пропускної здатності – рівномірність розподілу користувачів відносно трафіку.
4. Індекс споживання енергії – витрачена енергія з урахуванням витрат енергії на простоювання системи. Чим менше, тим краще.
5. Ємність машин у хмарному шарі.

Для перевірки продуктивності зазначеної архітектури було створено середовище моделювання, що складається з програмного забезпечення на мові MATLAB та симулятора EstiNet. Були виміряні основні параметри, що характеризують якість системи: загальний час затримки, відсоток успішно виконаних завдань (надійність) та швидкість обробки завдань (Quality of Service).

Для аналізу продуктивності запропонованої архітектури було обрано дві існуючі аналогічні архітектури, що також побудовані на використанні SDN: ASTP (Adaptive Selection and Task Priority) та SuVMF (Software-defined Unified Virtual Monitoring Function) [62].

Результати вимірювань показали, що вищенаведена архітектура має більші значення досліджуваних показників в порівнянні з аналогічними архітектурами:

– 90% показник надійності (проти 85% у ASTP та 70% у SuVMF);

– 90% показник QoS (проти 82% у ASTP та 68% у SuVMF);

Результати доводять, що використання SDN як засобу балансування навантаженням між ресурсами може значно підвищити продуктивність системи, що має високу актуальність у різних сферах IoT, пов’язаних із обробкою в реальному часі, наприклад, Industrial IoT [141]. Таким чином, подальші дослідження щодо використання цієї технології в IoT та системах Fog Computing є актуальними.

Серед перспективних напрямів подальшого розвитку архітектур розподілених телекомунікаційних систем особливо виділяються інноваційних технологій, які здатні суттєво підвищити ефективність, гнучкість та безпеку інфраструктури, до них належать наступні. Квантові обчислення у хмарі. Зважаючи на активний розвиток квантових технологій, аналіз впливу квантових обчислень на хмарні архітектури може відкрити нові можливості для підвищення швидкості та безпеки обчислень.

1. Blockchain-технології. Поєднання блокчейну з хмарними платформами та IoT сприяє децентралізованій обробці, прозорості та підвищенню рівня довіри до переданих і збережених даних.

2. Автономні системи на основі AI та машинного навчання. Інтеграція механізмів штучного інтелекту та машинного навчання дозволяє автоматизувати управління ресурсами та оптимізувати роботу в умовах змінного навантаження.

3. Цифрові двійники для IoT та хмарних систем. Віртуальні копії фізичних об’єктів у хмарному середовищі забезпечують детальне моделювання, тестування та прогнозування поведінки об'єктів у розподілених телекомунікаційних інфраструктурах.

4. Гібридні хмарні рішення. Використання комбінації приватних і публічних хмар дозволяє досягти високої адаптивності, масштабованості та ефективного використання обчислювальних ресурсів.

5. Edge AI для розумних міст та Індустрії 4.0: Застосування алгоритмів AI безпосередньо на вузлах мережі дає змогу реалізовувати локальну обробку даних у реальному часі, що є критичним для автономного управління інфраструктурою та виробничими процесами.

6. Квантові обчислення у хмарному середовищі. Дослідження потенціалу квантових технологій відкриває нові можливості для підвищення швидкодії, паралелізації процесів і стійкості до кібератак.

За результатами проведеного аналізу можна зробити висновок, що розглянуті архітектури дозволяють ефективно вирішувати ключові проблеми традиційних хмарних підходів, зокрема зменшення затримок та зниження перевантаженості мережі, шляхом перенесення частини обчислювальних операцій на вузли, розміщені ближче до джерел даних. Для РТС такі архітектури відкривають нові можливості для підвищення продуктивності, адаптивності та стійкості інфраструктури.

Водночас на практиці доцільно враховувати специфіку кожної телекомунікаційної платформи при виборі архітектурного рішення. Розвиток і вдосконалення Fog-архітектур залишається актуальним завданням, оскільки вони мають бути адаптовані до конкретних сценаріїв застосування, вимог галузі та обмежень середовища функціонування.

1.3 Аналіз методів розподілу навантаження в розподілених телекомунікаційних системах

Методи розподілу навантаження в розподілених телекомунікаційних системах є критичними для забезпечення надійності, продуктивності та доступності сервісів. Вони дозволяють системам ефективно масштабуватися, реагувати на коливання навантаження та оптимізувати використання ресурсів. На відміну від хостованих оперативних віртуальних середовищ (ХОВ), розподілені системи пропонують гнучкість у розміщенні та масштабуванні додатків, відокремлюючи їх від прямої прив’язки до фізичних серверів. Це сприяє підвищенню щільності додатків та дозволяє адміністраторам ефективніше управляти ресурсами без жорсткої залежності від конкретного апаратного забезпечення.

Процес розподілу ресурсів у розподілених телекомунікаційних системах, хоча і є аналогічним з точки зору викликів і проблем у ХОВ, проте вимагає спеціалізованого підходу через специфіку мережевої інфраструктури. Такі виклики, як необхідність синхронізації між вузлами, управління залежностями між сервісами та адаптація до мінливості трафіку, є особливо важливими в контексті розподілених систем, де навантаження може динамічно змінюватися та бути нерівномірно розподілене між різними точками обробки.

На рис. 1.6 класифіковані основні проблеми РТС та можливі методи їх вирішення.

Рисунок 1.6 – Аналіз проблем та методів вирішення для РТС

Таким чином, ефективне управління РТС досягається через автоматизацію, використання Edge computing для оптимізації реакції системи, децентралізоване управління для підвищення стабільності та зосередження на посиленні безпеки і

стандартизації для підвищення сумісності компонентів телекомунікаційної мережі.

Управління РТС базується на методах розподілу навантаження, які поділяються на динамічні та статичні. Динамічні методи засновані на адаптивності та гнучкості, оскільки дозволяють системі реагувати на змінні умови в реальному часі для оптимального розподілу ресурсів. Статичні методи – забезпечують попередньо визначене розподілення ресурсів, базуються на стабільних очікуваннях навантаження. Це забезпечує простоту та передбачуваність управління. Обидва підходи використовуються на практиці в розподілених телекомунікаційних системах, а вибір між ними залежить від специфічних потреб та цілей конкретної мережевої інфраструктури. Розглянемо ці підходи більш докладно.

1.3.1 Аналіз статичних методів розподілу навантаження в розподілених телекомунікаційних системах

Статичні методи розподілу базового навантаження в розподілених телекомунікаційних системах не враховують поточний стан вузлів або серверів при розподілі трафіку. Вони працюють на основі попередньо заданих правил і не змінюють свою стратегію відповідно до змін у системі. До найбільш розповсюджених статичних методів розподілу навантаження у РТС належать.

1. Round-robin (RR) – найпростіший метод розподілу навантаження, який рівномірно розподіляє запити між серверами у списку, з переходом від одного до іншого без урахування їх поточного навантаження або доступності.

До переваг методу відносять простоту в управлінні, рівномірний розподіл запитів за умови, що всі сервери мають однакову потужність).

Недоліками є те, що Round-robin (RR) не враховує поточне навантаження або стан серверів і тому з’являється нерівномірне навантаження на сервери, які мають різну потужність. Крім того, в RR відсутній механізм пріоритету запитів.

Покроковий алгоритм роботи методу Round-robin (RR) для розподілу навантаження між серверами у РТС виглядає наступним чином (рис. 1.7.)

1. Ініціалізація. Формується список усіх доступних серверів, які беруть участь у розподілі навантаження. Кожен сервер у цьому списку має унікальний ідентифікатор або адресу. Обрається початковий сервер для розподілу запитів. Це може бути перший сервер у списку.

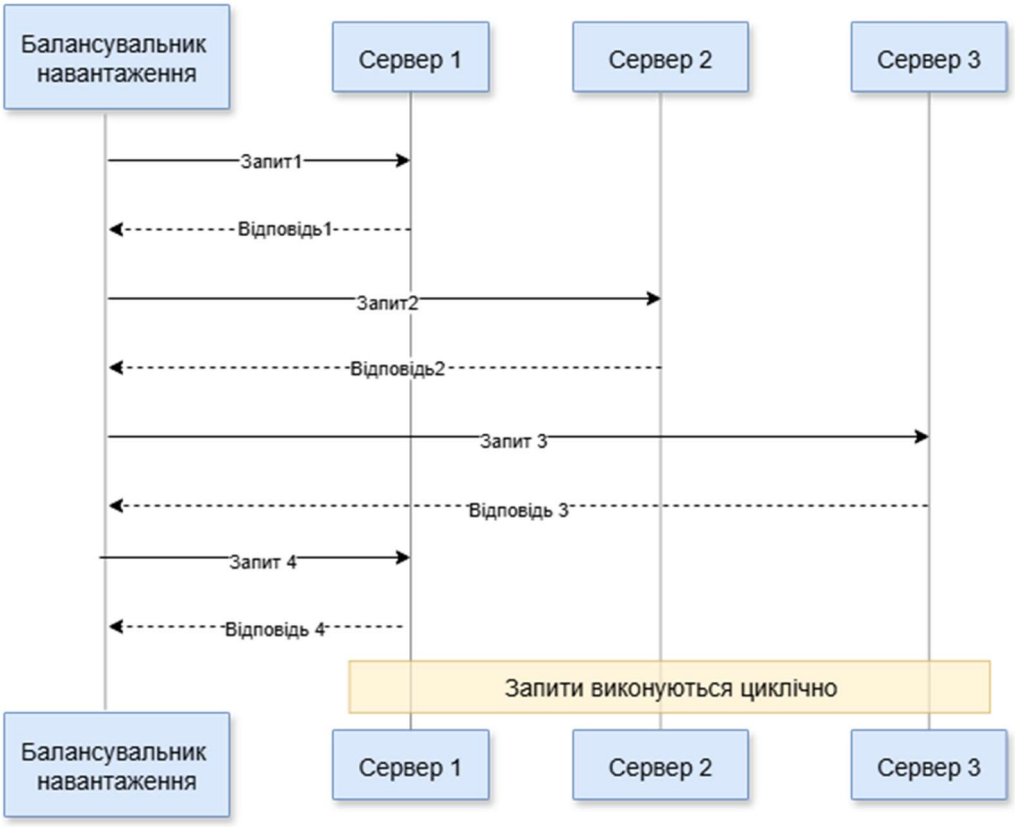


Рисунок 1.7 – Робота алгоритму Round-robin в РТС

2. Обробка запитів. Коли система отримує запит від клієнта, вона визначає, на який сервер направити цей запит, використовуючи алгоритм Round-robin. Обирається наступний сервер зі списку на основі поточної позиції. Якщо останній сервер у списку вже був використаний, алгоритм повертається до першого сервера у списку. Направляється запит на вибраний сервер для обробки. Сервер приймає запит і виконує необхідні операції. Після того, як запит було направлено на сервер, алгоритм оновлює свою позицію у списку серверів, з переміщенням до наступного сервера у списку.

3. Повторення процесу. Кожен наступний запит обробляється аналогічно, з вибором наступного сервера у списку.

4. Завершення обробки запитів. Процес продовжується до того моменту, поки є запити для обробки. Коли запити закінчуються, алгоритм залишається у стані очікування нових запитів.

2. Least Connections (LC). Обирається сервер з найменшою кількістю активних з’єднань і відправляється запит на сервер. Не враховується реальний стан серверів, використовується лише інформація про кількість з’єднань.

Алгоритм роботи методу LC полягає в систематичному виборі сервера з найменшою кількістю активних з’єднань для обробки нового запиту від клієнта. Після вибору сервера, запит відправляється для подальшої обробки. Після завершення обробки, сервер оновлює статистику, збільшуючи кількість активних з’єднань. Процес повторюється для кожного нового запиту, що забезпечує розподіл навантаження між серверами.

Недоліком є те, що може виникнути ситуація, коли обраний сервер має велику кількість активних з’єднань і обробка запитів виконується повільно через низьку пропускну здатність чи/або велике навантаження процесора (рис. 1.8).

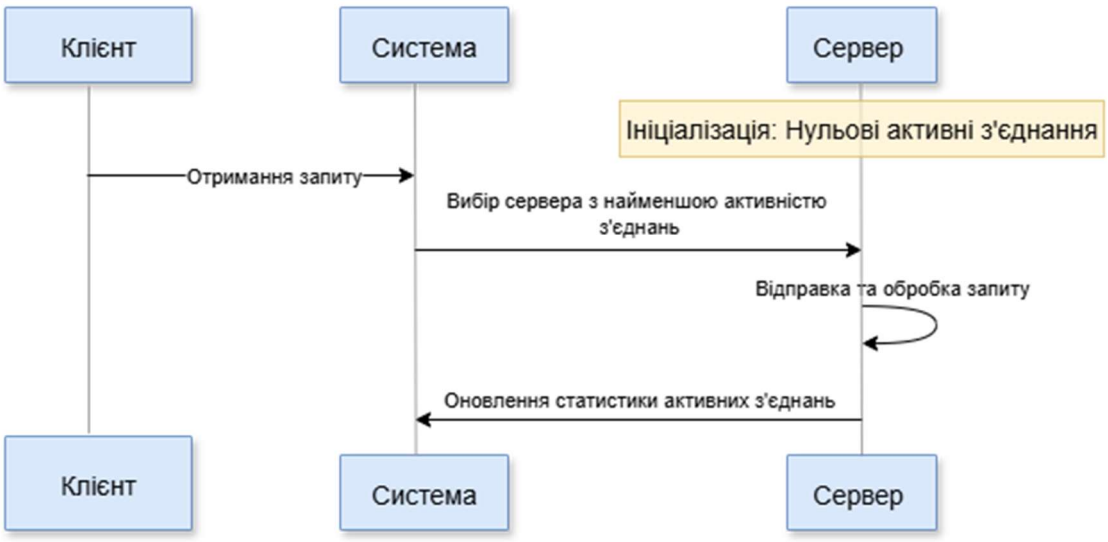


Рисунок 1.8 – Алгоритм роботи методу LC

3. Метод Weighted Round-Robin (WRR) є розширенням базового алгоритму Round-Robin, який використовується для розподілу навантаження між серверами в мережі. Головною відмінністю WRR є те, що кожному серверу призначається вага, яка відображає його потужність або пропускну здатність, дозволяючи краще врахувати відмінності між серверами (рис. 1.9).

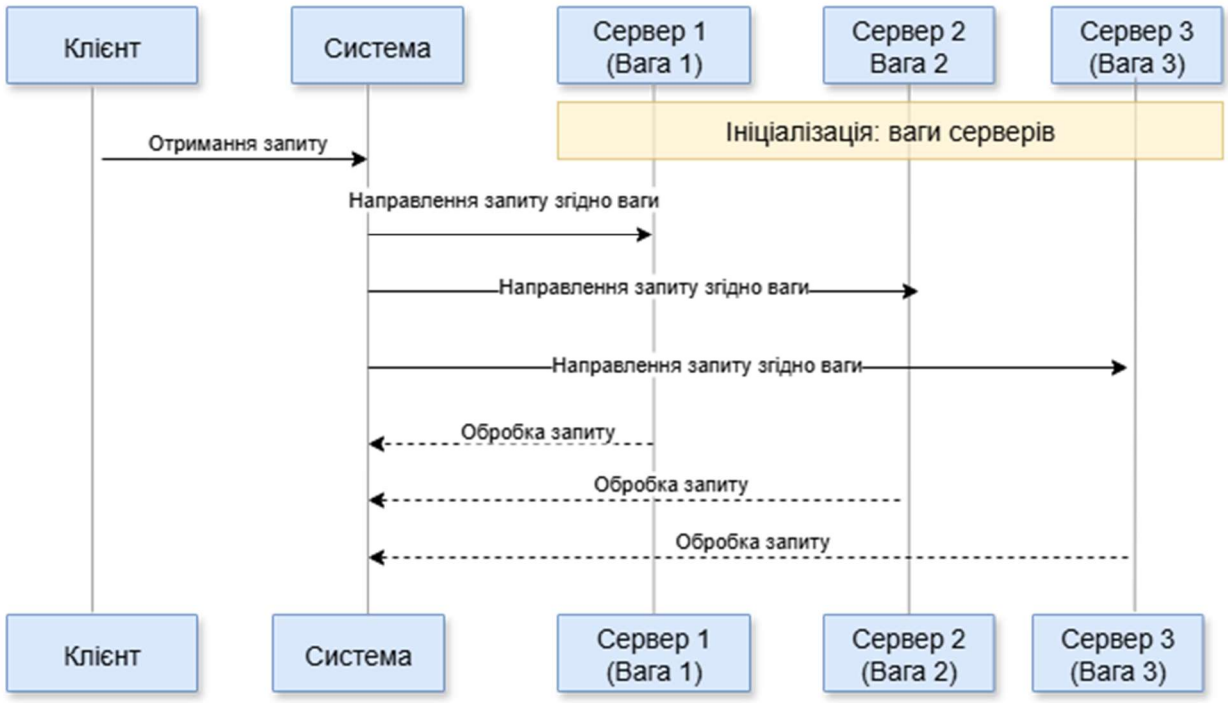


Рисунок 1.9 – Алгоритм роботи методу Weighted Round-Robin (WRR)

У WRR, завдяки ваги, сервери з вищою потужністю обслуговують більше запитів, ніж сервери з меншою потужністю, а саме:

– всім серверам присвоюються ваги відповідно до їх потужності або пропускної здатності. Вага визначає, скільки разів сервер буде обрано для обробки запиту перед тим, як перейти до наступного сервера в черзі;

– коли приходить запит на обслуговування, алгоритм вибирає наступний сервер у черзі на основі ваги;

– після того як сервер обробив запит відповідно до своєї ваги, алгоритм переходить до наступного сервера у черзі, враховуючи його вагу;

– процес продовжується циклічно.

Недолік – WRR не реагує на перевантаження або вихід з ладу серверів в реальному часі. Ваги серверів встановлюються статично і не адаптуються до змін. Порівняльна характеристика алгоритмів в РТС та в хмарних обчисленнях представлена в табл. 1.2.

Таблиця 1.2 – Статичні методи роботи в РТС та в хмарних обчисленнях

	Статичні методи в РТС		
	Round-Robin (RR)	Least Connections (LC)	Weighted Round-robin (WRR)
Переваги	Простота реалізації, не потребує координації між серверами.	Кращий розподіл навантаження при тривалих сесіях.	Гнучке управління ресурсами.
Недоліки	Низька ефективність із серверами різної потужності.	Надмірне навантаження на потужні сервери.	Потребує налаштування ваги серверів і додаткового управління.
	Статичні методи в хмарних обчисленнях		
	Round-Robin	Central Manager Algorithm	Threshold Algorithm
Переваги	Рівномірний розподіл навантаження. Інтеграція з Round-Robin DNS.	Централізований вибір найменш завантаженого сервера.	Адаптивність до змін, мінімізація частоти перерозподілу завдань, підвищення ефективності.
Недоліки	Не враховує поточне навантаження. Можливість застарілих записів у DNS.	«Пляшкова шийка», єдина точка відмови. Обмежена масштабованість	Складність налаштування порогів, потреба в постійному моніторингу, ризик затримок.

Таким чином, статичні методи розподілу навантаження не здатні забезпечити необхідну адаптивність та продуктивність у динамічних умовах сучасних РТС, що обґрунтовує необхідність впровадження більш гнучких підходів.

1.3.2 Дослідження динамічних методів розподілу навантаження

Динамічні методи розподілу навантаження (Dynamic Load Balancing) – це стратегії управління робочими завданнями і ресурсами в комп'ютерних системах, спрямовані на розподіл обчислювального навантаження між різними обчислювальними вузлами або серверами для досягнення більшої рівноваги, використання ресурсів та підвищення надійності і продуктивності системи.

Динамічні методи розподілу навантаження можуть бути реалізовані за допомогою різних методів та технологій, а саме.

1. Адміністратори комп'ютерних систем можуть вручну налаштовувати і керувати розподілом навантаження, визначаючи, який сервер або вузол буде обслуговувати конкретні запити чи завдання. Цей метод може бути ефективним для невеликих систем, але для систем великих масштабів або систем зі змінним навантаженням його не використовують.

2. Dynamic Power Management (DPM) – динамічне управління енергоспоживанням. Використовується для регулювання споживання енергії в обчислювальних системах. Вона може бути використана для розподілу навантаження шляхом вимкнення або усування неактивних ресурсів для зменшення споживання енергії.

Алгоритм DPM містить наступні етапи.

2.1 Оцінка навантаження. Це початковий крок, на якому система моніторингу оцінює поточне навантаження на CPU.

2.2 Моніторинг завантаження CPU. Система регулярно моніторить рівень завантаження CPU на всіх хостах.

2.4 Аналіз поточного стану. Перевірка, чи завантаження CPU на будь-якому з хостів перевищує заданий поріг протягом визначеного інтервалу часу.

2.5 Рішення про перерозподіл. Якщо виявлено, що завантаження CPU перевищує допустимі межі, система розглядає варіанти для перерозподілу ресурсів або навантаження між хостами.

2.6 Виконання перерозподілу. Після прийняття рішення про оптимальний варіант перерозподілу, система виконує необхідні дії для балансування навантаження (рис. 1.10).

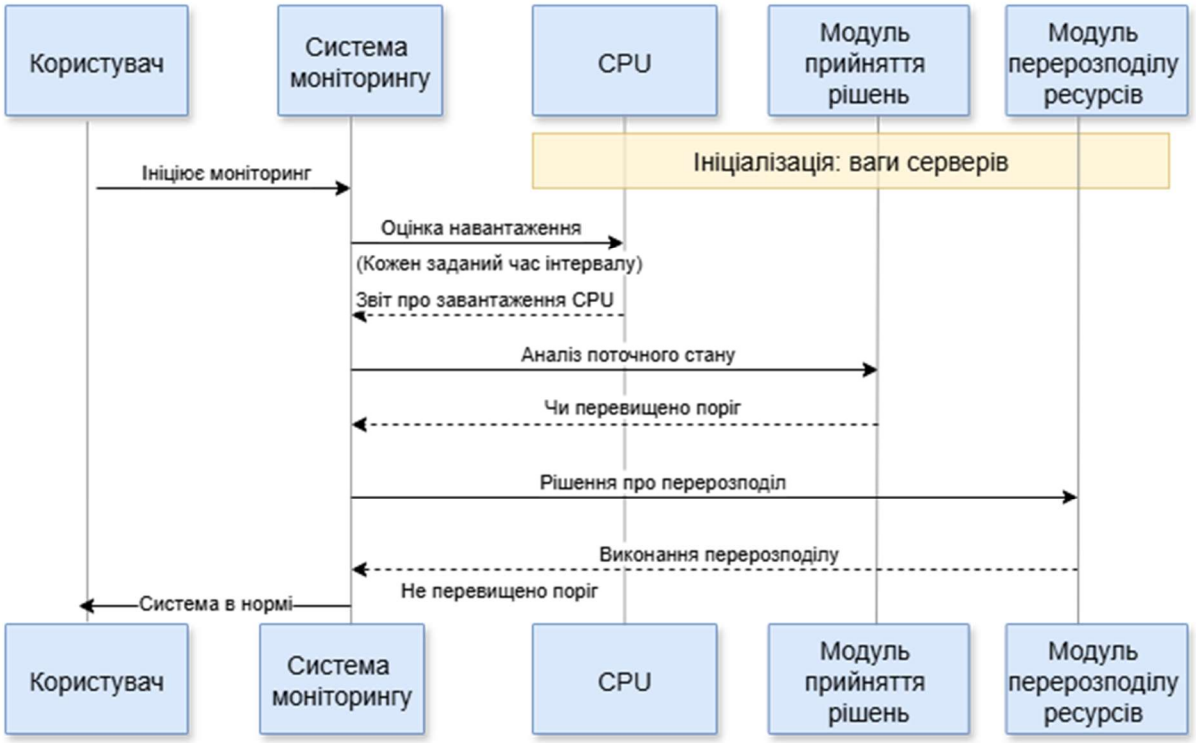


Рисунок 1.10 – Алгоритм динамічного управління споживанням DPM

3. Distributed Resource Scheduling (DRS) – динамічне планування ресурсів. Використовується в віртуальних середовищах (VMware vSphere). Автоматично розподіляє ресурси (процесор, пам’ять, мережу тощо) між віртуальними машинами в залежності від їх потреб і навантаження [113]. Етапи алгоритму DRS представлені на рис. 1.11. До етапів належать.

3.1 Користувач ініціює моніторинг на системі DRS.

3.2 DRS періодично збирає дані про навантаження сервера та використання ресурсів VMs (віртуальної машини).

3.3 Дані аналізуються для визначення потреби в перерозподілі ресурсів.

3.4 Якщо ресурси потребують перерозподілу, система DRS визначає VMs, яким потрібно більше ресурсів, визначає сервери з доступними ресурсами,

приймає рішення про перерозподіл та виконує міграцію VMs або коригування ресурсів.

3.5 Ресурси оптимальні – система DRS повідомляє, що зміни не потрібні.

3.6 Система продовжує моніторинг.

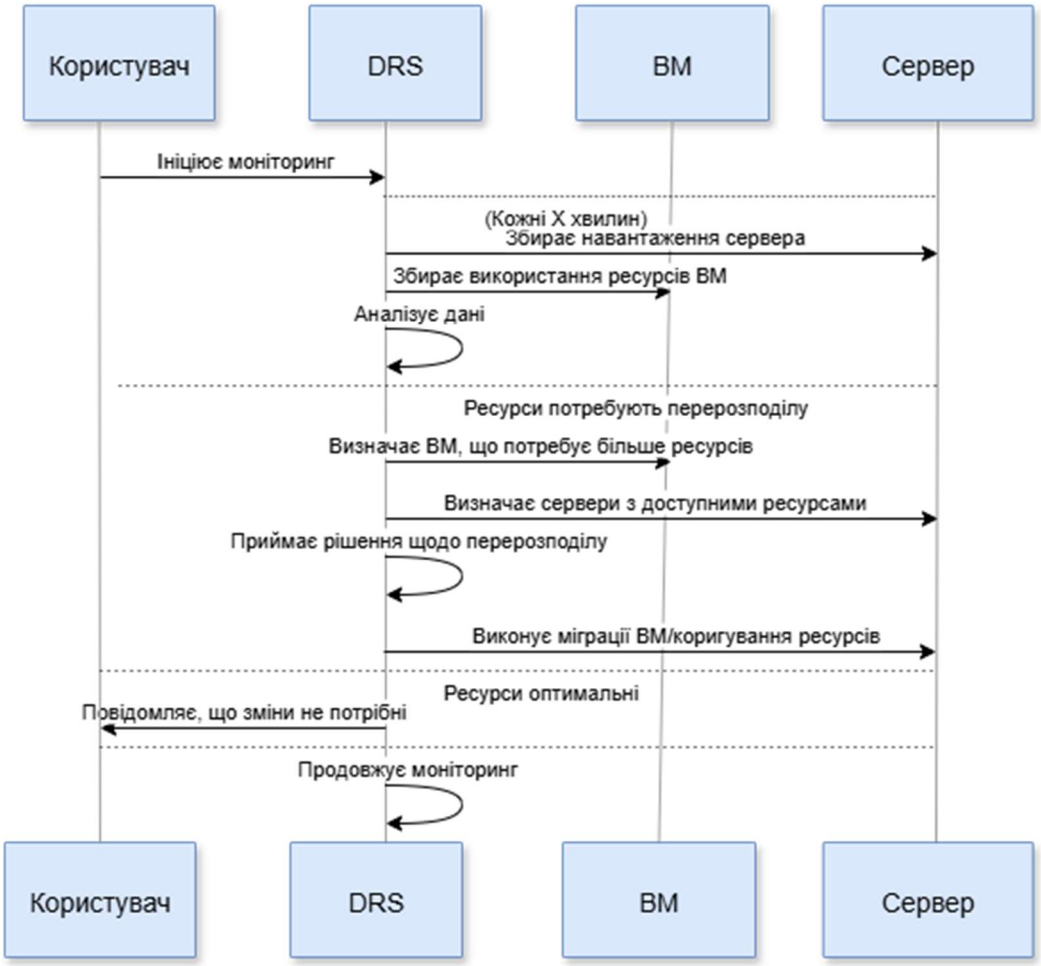


Рисунок 1.11 – Алгоритм динамічного планування ресурсів DRS

4. Алгоритми розподілу навантаження на основі аналізу даних (Data-Driven Load Balancing Algorithms) використовують аналіз даних про навантаження і використання ресурсів для прийняття рішень щодо розподілу навантаження [113]. Збирають історичні та реальні дані про навантаження і використання ресурсів серверів, аналізують для виявлення тенденцій і шаблонів, і на цій основі приймають рішення про оптимальне розподілення

вхідних запитів між серверами. Схематично алгоритм Data-Driven Load Balancing Algorithms представлений на рис. 1.12.

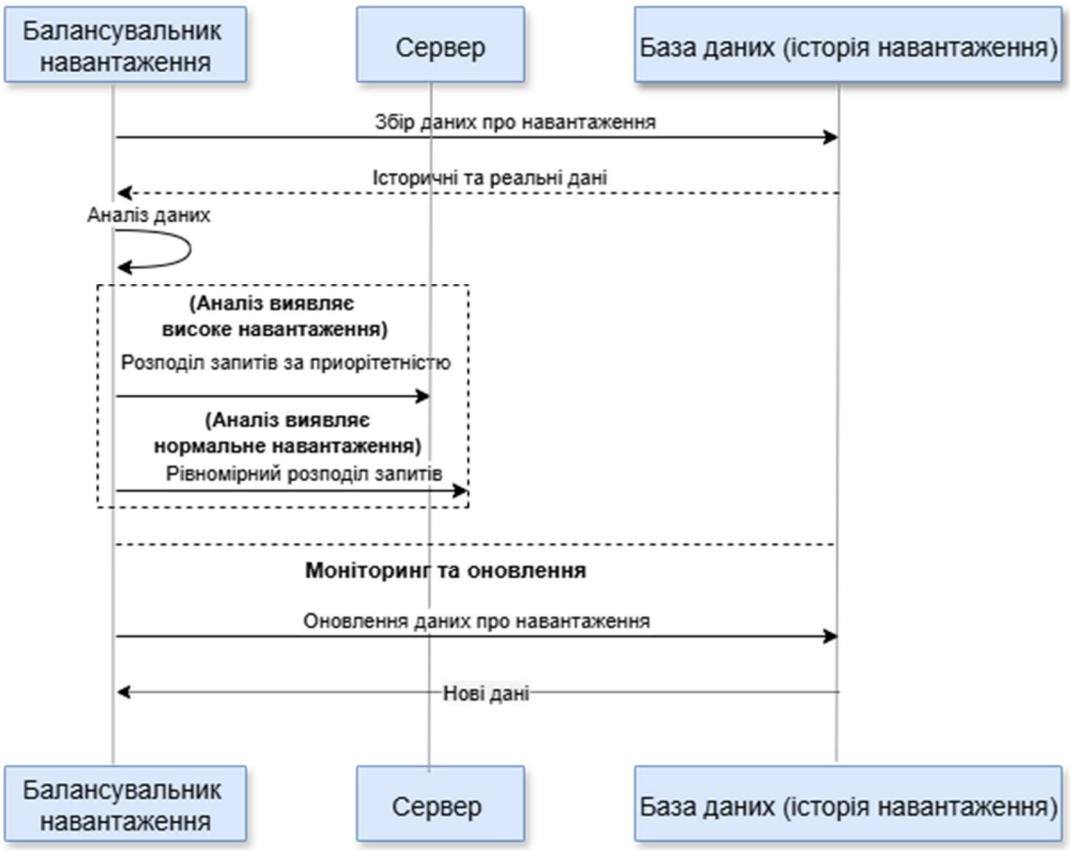


Рисунок 1.12 – Алгоритм Data-Driven Load Balancing Algorithms

5. Динамічні алгоритми призначення ресурсів (Dynamic Resource Allocation) реалізують ефективне управління ресурсами в РТС. Вони допомагають вирішити оптимізаційну задачу розподілу ресурсів, таких як процесорний час, обсяг пам’яті та пропускна здатність мережі, між різними завданнями або процесами на основі їх поточних потреб та доступності ресурсів [113]. Процес роботи алгоритму виглядає наступним чином:

– моніторинг поточних потреб у ресурсах;

– аналіз доступності ресурсів, включаючи процесорний час, обсяг пам’яті та пропускну здатність мережі,

– призначення або перерозподіл ресурсів,

– моніторинг змін в навантаженні та доступності ресурсів (рис. 1.13).

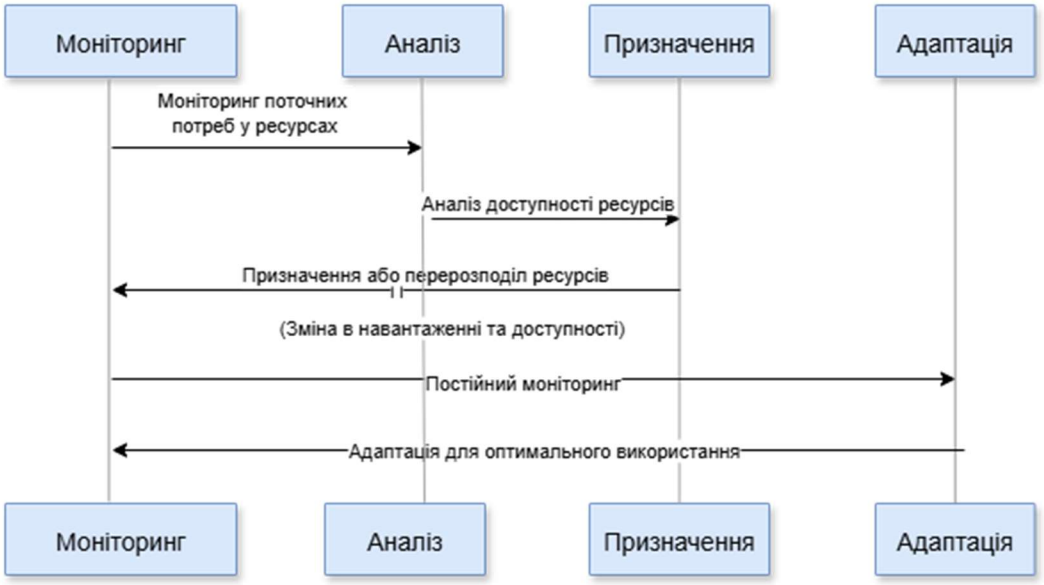


Рисунок 1.13 – Алгоритм Dynamic Resource Allocation

Порівняльний аналіз динамічних методів розподілу навантаження представлений в табл. 1.3.

Таблиця 1.3 – Порівняльний аналіз динамічних методів розподілу

Параметр порівняння	Динамічні методи
Адаптивність до змін	Висока (реагують на зміни в реальному часі)
Реалізація, підтримка	Висока (складні алгоритми, вимагають моніторингу та аналітики)
Стан серверів	Враховується поточний стан вузлів (завантаженість, доступність, ресурси)
Ресурсомісткість	Висока, досягається рівномірний та оптимальний розподіл ресурсів
Чутливість до відмов	Низька (автоматична реакція на відмови, перерозподіл навантаження)
Сфера застосування	Динамічне середовище з коливанням трафіку, великі та високонавантажені системи
Приклади використання	Високонавантажені корпоративні системи, хмарні сервіси, телекомунікаційні провайдери

Як видно з табл. 1.3, вибір динамічного методу розподілу навантаження в РТС визначається особливостями мережевої інфраструктури, інтенсивністю та типом трафіку, критичністю завдань та допустимим рівнем ресурсних витрат на впровадження і підтримку.

1.4 Обґрунтування науково-технічного завдання підвищення ефективності обробки даних у РТС

Проведений аналіз сучасних архітектур та методів розподілу навантаження у РТС показав, що наявні рішення здебільшого орієнтовані на централізовані або малогнучкі структури, які не враховують у повній мірі динамічний характер навантаження, структурну складність мережі та неоднорідність її компонентів. Виявлено, що існуючі методи кластеризації, вибору координаторів і розподілу обчислювальних задач не забезпечують достатнього рівня адаптивності до змін параметрів середовища, що призводить до затримок, перевантажень вузлів та зниження відмовостійкості системи.

Актуальним є впровадження механізмів динамічної самоорганізації, які дають змогу перебудовувати структуру взаємодії вузлів відповідно до змін навантаження та доступних ресурсів. Недостатня ефективність поточних підходів також обумовлена обмеженим урахуванням параметрів мережевих з’єднань під час прийняття рішень щодо маршрутизації та розподілу потоків, це відноситься до: затримки, пропускна здатність і надійність. Крім того, застосування класичних алгоритмів оптимізації в умовах багатокритеріальності та обмежених обчислювальних ресурсів є малоефективним, що зумовлює необхідність використання еволюційних підходів.

Отже, доцільно вирішити актуальне науково-технічне завдання підвищення ефективності обробки даних у РТС шляхом розробки нових методів, моделей та алгоритмів, здатних адаптивно формувати кластери обчислювальних вузлів, здійснювати стійкий вибір головних вузлів (координаторів) і забезпечувати оптимальний розподіл навантаження. Для

цього необхідно в дослідженні розглянути та запропонувати підходи до розв’язання таких часткових задач:

– розробити комплексний метод самоорганізації РТС з урахуванням продуктивності, топології мережі та параметрів каналів зв’язку;

– удосконалити алгоритм рекурсивної кластеризації на основі модифікованого підходу Лувена з динамічним регулюванням роздільної здатності;

– удосконалити метод вибору головного вузла на основі модифікованого алгоритму Gossip з асинхронно-узгодженим вибором координаторів;

– розробити метод формування обчислювальних конвеєрів для обробки потокових даних у кластеризованому середовищі з підтримкою динамічної самоорганізації;

– удосконалити метод багатокритеріального розподілу потоків на основі модифікації генетичного алгоритму NSGA-III з гібридним ранжуванням і адаптивним керуванням мутаціями;

– провести моделювання запропонованих методів, розробити програмну реалізацію, провести верифікацію та аналіз ефективності функціонування РТС на основі експериментальних результатів.

Таким чином, вирішення зазначеного завдання дозволить сформувати науково обґрунтовану базу для створення інтелектуально адаптивних розподілених телекомунікаційних систем, здатних ефективно функціонувати в умовах високої динаміки мережевого трафіку, обмежених обчислювальних ресурсів та зростаючих вимог до надійності передачі та обробки даних.

Висновки за розділом 1

1. Проведено аналіз сучасних архітектур, що використовуються для підвищення ефективності РТС. Досліджено особливості побудови дворівневих хмарних моделей, мультишарових Fog-архітектур, а також підходів, заснованих на SDN/NFV і Cloudlet-інфраструктурі. Показано, що традиційні централізовані

рішення не забезпечують необхідного рівня адаптивності, низької затримки та ефективності використання ресурсів у задачах обробки даних в реальному часі.

2. Обґрунтовано, що найбільш перспективними для подальшого вдосконалення РТС є архітектури Fog та Edge Computing, які забезпечують децентралізовану обробку даних, гнучке масштабування і стійкість до пікових навантажень. Окрему увагу приділено архітектурам, що реалізують територіальну розподіленість обчислень, адаптивний розподіл задач між рівнями і використання інтелектуальних механізмів керування ресурсами.

3. Проведено аналіз статичних методів розподілу навантаження, таких як Round-Robin, Least Connections і Weighted Round-Robin. Обґрунтовано, що вони мають перевагу у вигляді простоти реалізації та низьких вимог до обчислювальних ресурсів. Проте, їх обмеження у врахуванні поточного стану вузлів та неспроможність адаптуватися до змін у навантаженні роблять їх менш ефективними для складних і динамічних РТС.

4. Досліджено динамічні методи розподілу навантаження. Доведено, що вони забезпечують адаптивне балансування навантаження на основі поточного стану системи. Це дозволяє досягти рівномірного використання ресурсів, мінімізувати затримки та підвищити надійність. Водночас, динамічні методи потребують складної інфраструктури моніторингу і аналітики, що може підвищити вартість впровадження в реальні умови РТС, але виправдано у середовищах, де критичною є стабільність і продуктивність.

5. На основі аналізу сучасних архітектур та методів розподілу навантаження обґрунтовано, що сучасні рішення для РТС недостатньо адаптивні до динамічних змін мережі та обмежених ресурсів. Централізовані підходи не забезпечують ефективного кластеризаційного розподілу та управління потоками. Це обумовлює необхідність розробки нових методів самоорганізації, вибору координаторів і багатокритеріального розподілу навантаження.

РОЗДІЛ 2
МЕТОД БАГАТОКРОКОВОЇ ІЄРАРХІЧНОЇ КЛАСТЕРИЗАЦІЇ ВУЗЛІВ У РОЗПОДІЛЕНИХ ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ НА ОСНОВІ ГРАФОВИХ МОДЕЛЕЙ

2.1 Проектування багаторівневої архітектури та принципів взаємодії вузлів

Аналіз сучасних підходів до організації РТС свідчить про актуальність застосування графових моделей для кластеризації обчислювальних вузлів і управління мережевою топологією [8, 13, 17, 18, 50, 51, 73, 75, 76, 85, 98, 116, 125, 126, 135, 148]. Зокрема, в роботах [5, 51, 76, 85, 125, 126] доведено доцільність використання структурованих моделей для розподілених обчислень, однак більшість із них орієнтовані на статичні архітектури. Методи мінімізації затримок [14, 90] зосереджені переважно на топології передачі, не враховуючи змінність навантаження або динамічне переназначення вузлів. Алгоритми балансування [1, 12], що орієнтовані на мобільні обчислювальні середовища, недостатньо адаптовані до ієрархічної кластеризації.

Серед кластеризаційних підходів широке поширення отримав алгоритм Louvain [55, 73], який ефективний при роботі з великими графами, але має обмеження при формуванні дрібних кластерів через фіксований параметр модульності. Модифікація цього підходу в алгоритмі Leiden [148] дозволяє частково усунути проблему роздільної здатності, проте супроводжується зростанням обчислювальної складності. Інші підходи, пов’язані з розміщенням сервісів [76, 85, 98, 125] або управлінням трафіком у SDN [116], також не охоплюють задачі адаптивного групування вузлів з динамічним урахуванням топології та навантаження.

Тобто існуючі методи не забезпечують комплексного вирішення задач ієрархічної кластеризації вузлів у РТС з урахуванням обмежень на розмір кластерів, затримки між вузлами та змінність параметрів мережі. Це обумовлює

необхідність розробки нового методу, що поєднує механізми графової оптимізації, динамічне управління параметрами кластеризації та адаптацію до навантаження в реальному часі.

Проектування багаторівневої архітектури РТС повинно починатися з визначення обмежень наявних мережевих протоколів та технологій, що забезпечують масштабованість, адаптивність та ефективний розподіл навантаження між вузлами мережі [10, 36, 110].

Однією з головних проблем при проектуванні є організація ефективної взаємодії вузлів, що мають різну продуктивність, мережеві характеристики та рівень доступності. У традиційних підходах, таких як IEEE 802.1Q [152], існують обмеження масштабованості, які можуть негативно впливати на гнучкість системи. До них належать:

1. Робота тільки на другому рівні моделі OSI, що робить неможливим об’єднання мереж, розділених шлюзами (глобальними мережами), без використання додаткових тунельних протоколів (Layer 2 over Layer 3).

2. Ліміт у 4094 ідентифікаторів VLAN через 12-бітне адресування, де значення 0 і 0xFFF зарезервовані.

Для усунення обмежень традиційних VLAN-рішень було розроблено стандарт IEEE 802.1ad (Q-in-Q) [152], який дозволяє додавати кілька тегів VLAN у пакет, що розширює можливості створення вкладених віртуальних мереж. Це забезпечує мінімум 16 млн ідентифікаторів за умови використання двох VLAN-тегів. Однак навіть цей стандарт не забезпечує достатньої гнучкості у глобальних розподілених системах, що потребують автоматичного балансування трафіку та адаптивного управління ресурсами вузлів. Для вирішення цієї проблеми використовуються сучасні тунельні протоколи третього рівня OSI, такі як VXLAN, GRE та Geneve [10, 36, 141, 152], які підтримують масштабування та адаптацію до змінних умов:

– VXLAN усуває обмеження VLAN, використовуючи 24-бітний ідентифікатор VNI (до 16 млн можливих ідентифікаторів), що дозволяє масштабувати кластеризовані обчислювальні системи;

Крім того, ці протоколи підтримують шифрування, що є критичним для захисту міжкластерного трафіку та керуючих обмінів між вузлами. Інтеграція тунельних технологій у багаторівневу архітектуру РТС дозволяє: GRE є простим у налаштуванні, підтримує різні типи трафіку, що важливо для взаємодії між вузлами у динамічних мережевих структурах;

– Geneve – оптимізований для хмарних обчислень, дозволяє інтеграцію з мережевими функціями, що особливо корисно при гнучкому розподілі обчислювальних задач між кластерами [36, 110].

Крім того, ці протоколи підтримують шифрування, що є критичним для захисту міжкластерного трафіку та керуючих обмінів між вузлами. Інтеграція тунельних технологій у багаторівневу архітектуру РТС дозволяє:

– створювати ієрархічну кластерну структуру, де обчислювальні вузли згруповані за продуктивністю та типами задач;

– ефективно розподіляти навантаження між кластерами, враховуючи затримки передачі та пропускну здатність каналів;

– мінімізувати міжкластерні затримки, використовуючи динамічне переназначення вузлів, що дозволяє системі адаптуватися до змін у мережевому середовищі.

Окрім ефективного управління обчислювальними ресурсами та мінімізації затримок у міжкластерній взаємодії, важливим викликом для сучасних розподілених телекомунікаційних систем є обробка та передача великих обсягів даних, що генеруються пристроями IoT [83, 110].

Зростання кількості IoT-пристроїв у мережах РТС значно ускладнює балансування навантаження, адже трафік від таких пристроїв нерівномірний, може змінюватися в реальному часі та суттєво впливати на пропускну здатність каналів. Зокрема, IoT-датчики та камери відеоспостереження створюють потоки даних із високою частотою оновлення, що може спричинити локальні перевантаження мережевих сегментів [75, 89].

Для розв’язання цих проблем у багаторівневій архітектурі використовуються вузли Edge та Fog, що забезпечують адаптивну фільтрацію,

попередню обробку та агрегацію даних, зменшуючи навантаження на магістральні канали та обчислювальні ресурси кластерів.

IoT-пристрої є одним із головних джерел навантаження в РТС. Наприклад, камери відеоспостереження генерують великі обсяги відеоданих, що можуть перевантажувати канали зв'язку. Це особливо критично для нестабільних або обмежених мережевих середовищ, таких як мобільний зв'язок (3G, 4G, 5G) чи супутниковий інтернет (Starlink, Viasat) [141].

Щоб уникнути перевантажень і знизити вимоги до пропускної здатності каналів, застосовуються вузли рівня Edge, які:

– виконують локальну обробку даних перед передачею в основну мережу;

– фільтрують і зменшують необхідний обсяг переданих даних (наприклад, знижуючи частоту кадрів у відеопотоці до 1–10 кадрів за секунду);

– забезпечують попередню аналітику для підвищення ефективності кластеризованих систем.

Доповненням до вузлів Edge є вузли Fog, що виконують агрегацію трафіку, балансування навантаження між кластерами та підготовку даних для хмарних обчислень. Fog-вузли також стабілізують роботу системи під час пікових навантажень або динамічних змін у мережевій топології. Окрім управління трафіком, одним із ключових викликів у розподілених системах є оптимізація енергоспоживання, особливо для вузлів Fog і Edge, які працюють у складних або віддалених умовах [75, 83, 89, 110].

Сучасні процесори (наприклад, ARM) здатні динамічно змінювати частоту для економії енергії, але навіть у стані простою (idle) енергоспоживання може досягати 50% пікового значення через підтримку базових компонентів (ОЗУ, процесор, диски тощо). Це вимагає:

– рівномірного балансування навантаження між усіма доступними ресурсами;

– застосування адаптивних алгоритмів управління енергоспоживанням у кластеризованих системах;

– вибору оптимальних вузлів-зв’язків (Bridge Nodes) для мінімізації зайвих передач даних між кластерами.

Таким чином, інтеграція сучасних тунельних протоколів дозволяє мінімізувати затримки, підвищити продуктивність та забезпечити масштабованість системи.

Основні вимоги до сучасних РТС, сформовані на основі проведеного аналізу наукових джерел [50, 76, 90, 123, 126, 148], представлені в таблиці 2.1.

Таблиця 2.1 – Основні вимоги до архітектури РТС

Вимога	Приклади реалізації	Переваги	Обмеження
Масштабо-ваність	Багаторівнева кластеризація, балансування навантаження.	Гнучкість, підтримка великих мереж.	Складність керування.
Відмово-стійкість	Резервування вузлів, Bridge Nodes.	Надійність, швидке відновлення.	Додаткові витрати.
Мінімізація затримок	Локалізація трафіку, MST, Flow-Based Clustering.	Висока швидкодія, менші затримки.	Можливе перевантаження вузлів.
Енерго-ефективність	ARM-процесори, балансування навантаження.	Оптимізація ресурсів, зниження споживання.	Складність алгоритмів.
Гнучкість адаптації	Динамічне переназначення вузлів.	Адаптація до змін, швидке реагування.	Високі обчислювальні витрати.
Безпека	Шифрування, тунелювання, ізоляція кластерів.	Захист від атак, сегментація трафіку.	Зростання витрат на обчислення.
Гнучке управління ресурсами	Децентралізоване управління, SDN-контролери.	Оптимальний розподіл ресурсів.	Складність координації вузлів.

Для задоволення виявлених вимог архітектура РТС має будуватися як багаторівневий процес. Послідовність етапів представлена на рис. 2.1.

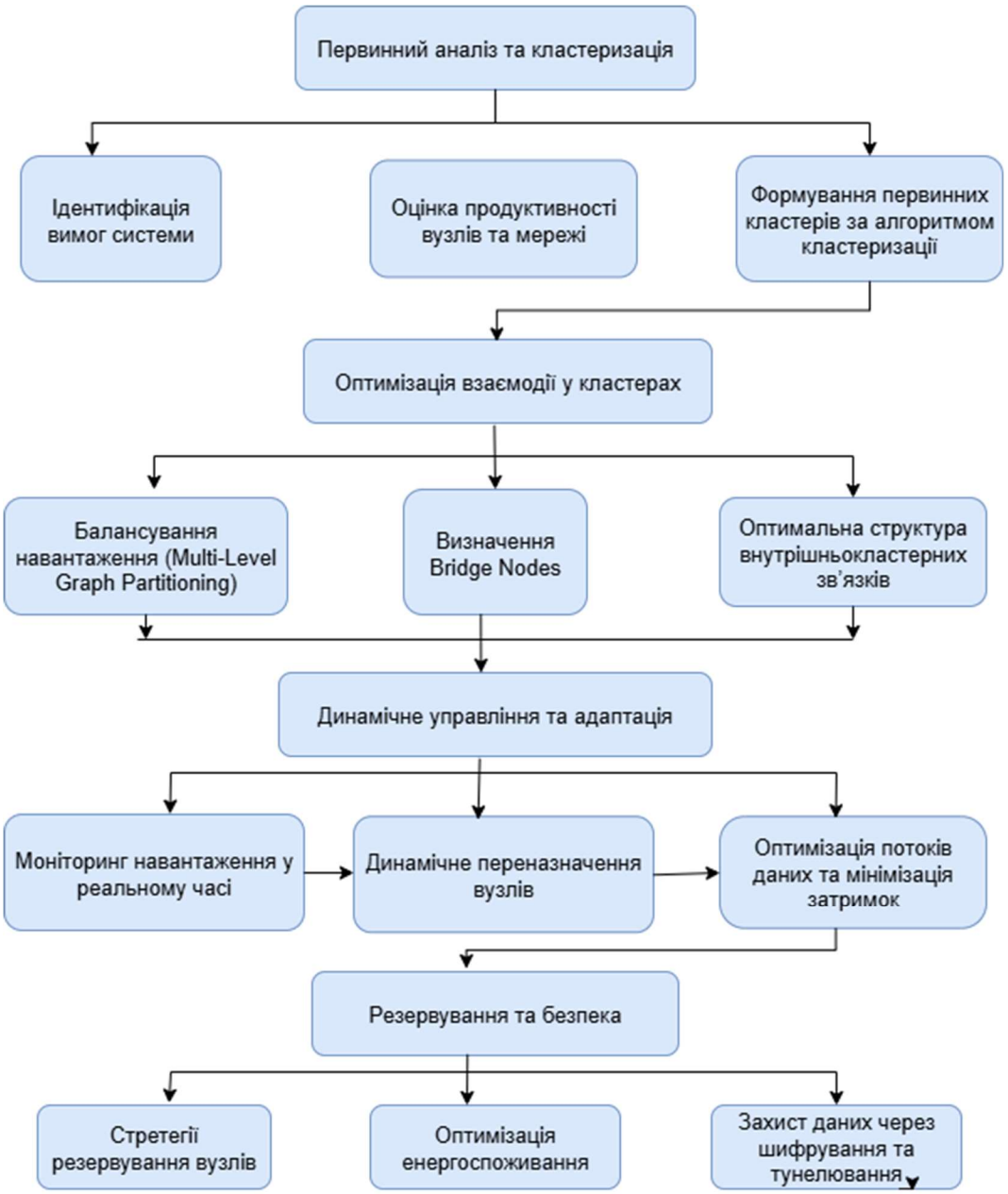


Рисунок 2.1 – Етапи побудови архітектури РТС

Розглянемо ці етапи більш докладно.

І Етап. Ідентифікація характеристик системи та моделі передачі даних

На першому етапі визначаються основні характеристики інфраструктури мережі, обчислювальних вузлів та типів задач, які вони повинні виконувати. Основні параметри, що треба проаналізувати на цьому етапі:

– типи обчислювальних задач (потокова аналітика, обробка IoT–даних, машинне навчання тощо);

– кількість та продуктивність вузлів (IoT–пристрої, Edge, Fog, Cloud);

– топологія мережі (затримки, пропускна здатність, доступність з’єднань між вузлами).

Для передачі даних використовується дві основні моделі.

1.Push-модель – дані надсилаються від джерела до вузла вищого рівня автоматично. Використовується для потокових задач, наприклад, відеоспостереження або сенсорних мереж IoT.

2. Pull-модель – вузол вищого рівня запитує дані у підлеглих вузлів за потреби. Використовується в аналітичних задачах, де не всі дані потрібні в реальному часі.

Оптимальний вибір моделі залежить від характеристик мережі та вимог до часу обробки.

II Етап. Формування кластерної структури та розподіл ресурсів

Цей етап спрямований на ефективну кластеризацію обчислювальних вузлів, щоб мінімізувати затримки та оптимізувати використання ресурсів. В межах цього етапу реалізуються методи.

1. Формування базових кластерів: вузли групуються за продуктивністю, мережевими характеристиками та типом задач, які вони можуть виконувати.

2. Балансування навантаження: продуктивні вузли виконують складні завдання, тоді як менш потужні вузли можуть обробляти локальні задачі або виступати у ролі вузлів-проміжників.

3.Рольова організація вузлів:

– вузли Edge виконують попередню обробку IoT-даних, зменшуючи вимоги до магістральних каналів;

– вузли Fog агрегують дані та здійснюють оптимізацію потоків між рівнями Edge і Cloud;

– вузли Cloud забезпечують збереження великих масивів даних та глибоку аналітику.

Реалізація Етапу II дозволяє адаптувати систему до змін у навантаженні та топології мережі.

III Етап. Оптимізація управління затримками та енергоефективністю

Для зниження затримок та підвищення продуктивності застосовуються такі методи.

1. Локалізація трафіку в межах кластерів – мінімізація міжкластерних обмінів дозволяє знизити час передачі даних.

2. Динамічне регулювання частоти передачі – вузли Edge можуть адаптивно змінювати частоту переданих кадрів у відеопотоці для уникнення перевантаження (зменшення частоти з 30–60 кадрів до 1–10 кадрів за секунду).

3. Оптимізація енергоспоживання:

– балансування завантаження вузлів дозволяє уникати перевантаження окремих елементів системи;

– використання ARM-процесорів та алгоритмів енергозбереження дає змогу знижувати загальні витрати енергії;

– перехід вузлів у режим енергозбереження під час низького навантаження.

IV Етап. Резервування та забезпечення відмовостійкості

Для підвищення стійкості системи реалізується механізм резервування вузлів і децентралізованого управління. Основні принципи.

1. Резервні вузли – визначається мінімально необхідна кількість резервних вузлів для підтримки роботи системи у разі виходу з ладу основних компонентів.

2. Протоколи швидкого перемикання – забезпечують миттєве перенаправлення трафіку та задач на резервні вузли у разі відмови.

3. Децентралізоване управління – усуває ризик «єдиної точки відмови» шляхом рівномірного розподілу керуючих функцій між вузлами.

Таким чином, запропонована багаторівнева структура РТС дозволяє досягти високої ефективності у взаємодії вузлів, оптимізувати використання ресурсів та адаптуватися до змін у мережі в реальному часі.

2.2 Розробка комплексного методу самоорганізації розподілених телекомунікаційних систем з урахуванням продуктивності та параметрів мережевої інфраструктури

На основі принципів побудови архітектури РТС (див. розділ 2.1) та аналізу сучасних методів кластеризації [51, 75, 85, 90, 145] розроблено метод самоорганізації РТС, спрямований на оптимізацію структури мережі. Запропонований підхід враховує неоднорідність продуктивності вузлів, варіативність параметрів мережевих з'єднань (затримки, пропускна здатність) та динамічні зміни в топології мережі, що впливають на стабільність та ефективність розподілених обчислювальних процесів (рис. 2.2).

Рисунок 2.2 –Рівні реалізації методу самоорганізації РТС

Як видно з рис. 2.2., комплексний метод реалізується порівнево: початкова кластеризація з урахуванням характеристик вузлів і мережі, оптимізація взаємодії всередині кластерів, визначення ключових вузлів (Bridge Nodes) для міжкластерної маршрутизації, багаторівневе балансування, динамічне переназначення вузлів, резервування та підвищення енергоефективності.

Слід зазначити, що повна реалізація всіх рівнів методу сприяє досягненню найвищої ефективності РТС, але у реальних сценаріях можливе удосконалення лише окремих елементів. Послідовність виконання рівнів не є строгою і залежить від топології, ресурсних обмежень і динаміки навантаження.

До особливостей запропонованого методу самоорганізації належать.

1. Кластеризація за задачами (Task-Oriented Clustering). Вузли групуються відповідно до набору задач, які вони можуть виконувати [98, 126]. Оскільки кожен вузол може обробляти декілька типів задач, метод кластеризації дозволяє формувати групи вузлів, що можуть спільно виконувати одну або кілька задач, враховуючи їх продуктивність.

2. Ієрархічна (багаторівнева) структура кластерів (Hierarchical Cluster Organization). Метод кластеризації передбачає багаторівневу кластеризацію [145], за рахунок ітеративного розділення графової структури з адаптивним налаштуванням параметрів.

3. Оптимізація внутрішньокластерної (Intra-Cluster Optimization) та міжкластерної (Inter-Cluster Optimization) взаємодії. Спрямована на мінімізацію затримок всередині та ззовні кластерів, що актуально при обробці даних в системах реального часу [90].

4. Використання вагових параметрів зв’язків (Latency & Bandwidth-Aware Clustering). Оскільки географічне розташування вузлів не визначене, оптимізація взаємодії базується на затримках та пропускній здатності каналів зв’язку [90].

5. Динамічне переназначення вузлів (Dynamic Node Reallocation). Для підтримки стабільної продуктивності система передбачає механізм динамічного переназначення вузлів між кластерами залежно від поточного навантаження та доступних ресурсів [145]. А саме:

– вузли можуть одночасно належати до кількох кластерів і виконувати різні задачі паралельно, якщо їх обчислювальні можливості це дозволяють;

– у разі перевантаження окремого кластеру, частина його вузлів може бути тимчасово перенаправлена в інший кластер для перерозподілу завдань;

– автоматичний моніторинг продуктивності та навантаження дозволяє адаптивно змінювати конфігурацію кластерів у реальному часі, забезпечуючи високу стійкість системи до змін.

Оскільки повна реалізація всіх рівнів методу самоорганізації РТС виходить за межі одного дослідження, у роботі поетапно реалізовано його основні компоненти:

– метод і алгоритм формування кластерів із цільовими характеристиками на основі модифікованого алгоритму Лувена з параметром роздільної здатності (Розділ 2);

– методи і алгоритми вибору головних вузлів і побудови обчислювальних конвеєрів із урахуванням нестабільності мережі, пропускної здатності та затримок (Розділ 3);

– методи і алгоритми інтелектуального розподілу задач між кластерами з використанням модифікованого алгоритму NSGA-III, що враховує обмеження ресурсів, змінне навантаження та потребу в делегуванні (Розділ 4).

Результати порівняння запропонованого методу кластеризації з методом-референсом, обґрунтованим науковцями в роботі [98], представлені в табл. 2.2.

Таблиця 2.2 – Порівняння методів кластеризації

Критерій	Запропонований метод	Метод-референс [98]	Перевага
Тип кластериза-ції	Ієрархічна багаторівнева кластеризація	Двоступенева кластеризація спільнот	Швидша організація та ефективний розподіл ресурсів

Складність для пошуку спільнот	Алгоритм Лувена	Алгоритм Гірвана-Ньюмана	Менше обчислень, вища швидкість
Параметр ваги	Комбінована (затримка + пропускна здатність)	Кількість переходів (hop)	Краще врахування фізичних параметрів мережі
Масштабо-ваність	Адаптивна ієрархічна структура	Значна залежність від розміру громади	Ефективна у великих системах
Оптимізація затримок	Локальна кластеризація з гео-оптимізацією	Транзитивне замикання сервісів	Менші затримки передачі даних між вузлами
Пріоритети кластери-зації	Мінімізація затримок із збереженням розміру кластерів	Рівномірний розподіл послуг	Швидше управління ресурсами та адаптація до збоїв
Пріоритеза-ція ресурсів	Завдання – продуктивним вузлам, контроль – слабким	Рівномірний розподіл	Ефективне використання продуктивних вузлів
Відмово-стійкість	Децентралізація + динамічне балансування ресурсів	Мінімізація впливу відмов через спільноти	Гнучкіше відновлення роботи системи
Продуктив-ність	Мінімізація навантаження на вузли	Орієнтація на дедлайни сервісів	Ефективне використання ресурсів
Адаптив-ність	Динамічне переназначення вузлів	Жадібний алгоритм із фіксованими пріоритетами	Краща реакція на зміну навантаження

Як видно з табл. 2.2, запропонований метод показує кращі результати ніж метод-референс за критеріям: вища адаптивність до змін мережі, зниження затримки за рахунок локальної оптимізації, можливість більш ефективного використання ресурсів продуктивних вузлів.

2.3 Обґрунтування вибору алгоритмів кластеризації для багаторівневих телекомунікаційних систем

Оскільки реалізація запропонованого методу кластеризації розпочинається з первинного формування кластерів, що передбачає групування вузлів відповідно до їх обчислювальних можливостей та типів задач, необхідно обрати оптимальний алгоритм кластеризації. Він має забезпечувати ефективне знаходження кластерів у графах з урахуванням продуктивності вузлів, затримки передачі даних і пропускної здатності каналів зв’язку.

Традиційні алгоритми кластеризації не завжди враховують топологічні особливості РТС та динаміку зміни навантаження, що може призводити до неефективного використання ресурсів. Тому, з метою обґрунтування вибору оптимального алгоритму кластеризації, який є найбільш релевантним для реалізації запропонованого методу, проведено порівняльний аналіз сучасних підходів (табл. 2.3).

Найбільш відповідними для наукових задач даного дослідження є алгоритми Лувена та його модифікація – алгоритм Лейдена [50,73,148].

1. Алгоритм Лувена було обрано через його високу швидкість і масштабованість, що є особливо актуальним для розподілених обчислювальних систем. Він забезпечує оптимізацію модульності графа, що дозволяє ефективно виявляти спільноти та формувати кластери. Основний процес роботи алгоритму Лувена складається з таких етапів (рис. 2.3):

– ініціалізація – кожен вузол вважається окремою спільнотою;

– локальне покращення – вузли переміщуються між спільнотами так, щоб покращити функцію якості (модульність);

– агрегація спільнот – вузли в межах однієї спільноти об’єднуються в супер-вузол, що утворює новий граф;

– повторення – процес повторюється рекурсивно, поки не буде досягнута стабільність.

Таблиця 2.3 – Порівняння алгоритмів кластеризації

Метод	Принцип дії	Багато-задачність	Вагові параметри	Оптиміза ція за зв’язками	Переваги	Недоліки
Agglomerative Clustering	Послідовне об’єднання вузлів у кластери	-	+	-	Гнучкий вибір метрики, адаптація до структури даних	Потрібно заздалегідь визначати кількість кластерів
Алгоритм Лувена (Louvain)	Оптимізація модульності для виявлення спільнот	+	+	+	Висока швидкість і масштабованість	Не дуже ефективний для динамічних мереж
Алгоритм Лейдена (Leiden)	Оптимізація модульності з удосконаленим розбиттям	+	+	+	Формує більш стійкі та точні кластери, знімаючи обмеження масштабу кластеризації	Вимагає більше часу на обчислення у порівнянні з Louvain
K-Medoids (ваговий варіант)	Кластеризація з вибором представника на основі схожості	+	+	+	Враховує топологію мережі, стійкий до викидів	Високі витрати на обчислення
Multi-Level Graph Partitioning (METIS, KaHIP)	Поділ графа з мінімізацією переривань між кластерами	+	+	+	Швидке балансування навантаження	Не дуже ефективний для малих графів
Flow-Based Clustering	Максимальний потік для кластеризації	+	+	+	Враховує пропускну здатність каналів зв’язку	Погано адаптується до змінного навантаження

Основні етапи алгоритмів Лувена та Лейдена представлені на рис. 2.3 та рис. 2.4.

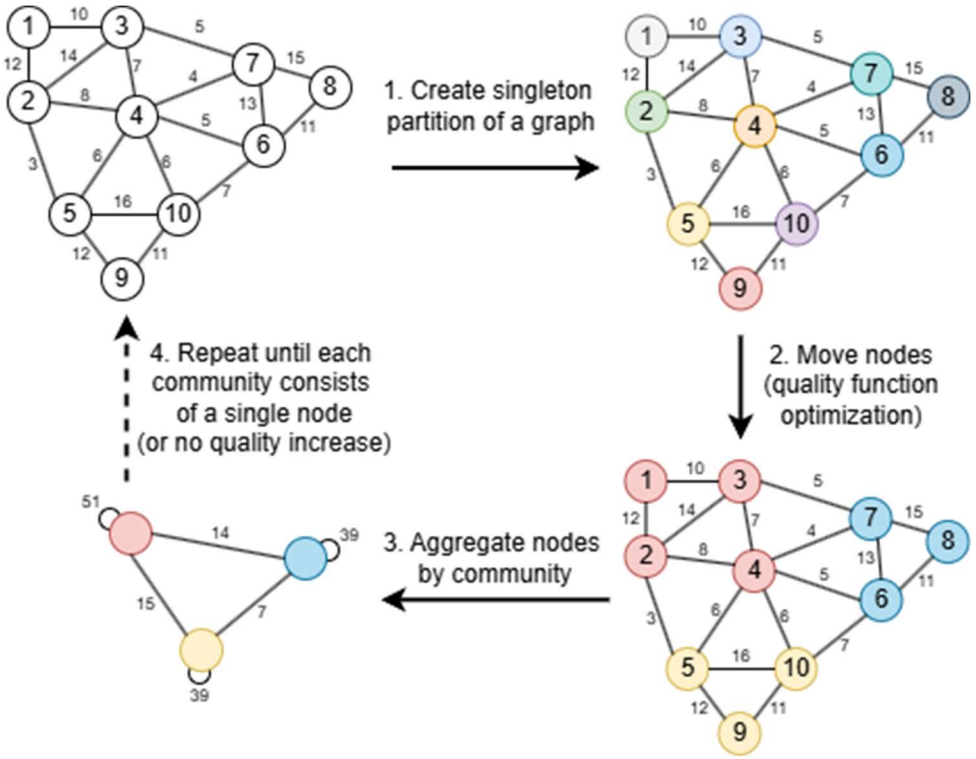


Рисунок 2.3 – Етапи роботи алгоритму Лувена

У початковому стані (чорно-біле зображення) усі вузли не належать жодному кластеру. Кольором на наступних етапах позначено належність вузлів до відповідних кластерів, сформованих під час кластеризації.

В результаті дослідження виявлено, що основним недоліком алгоритму Лувена є схильність до формування надмірно великих кластерів, що не відповідає задачам цього дослідження, оскільки необхідне детальніше розбиття вузлів із точнішим урахуванням їх внутрішньої структури [73]. Це пов’язано з тим, що на кожній ітерації нові спільноти формуються виключно на основі приросту модульності, що не завжди гарантує оптимальність їх розбиття.

2. Алгоритм Лейдена в даному дослідженні було обрано для порівняння ефективності роботи запропонованого методу ієрархічної кластеризації. Основною перевагою алгоритма Лейдена є усунення основних недоліків алгоритму Лувена, а саме: обмеження роздільної здатності (resolution limit) та постобробка кластерів (refinement) [148].

На відміну від Лувена, Лейден-кластеризація виконує не лише глобальне оптимізаційне оновлення, але також проміжне уточнення на рівні окремих вузлів, що забезпечує більш детальне групування.

Процес роботи алгоритму Лейдена складається з етапів:

– ініціалізація – кожен вузол стає окремою спільнотою;

– переміщення вузлів – відбувається покращення функції модульності шляхом локального пошуку;

– уточнення кластерів – застосовується додаткова стадія перевірки та коригування поділу;

– агрегація – вузли в межах спільноти об’єднуються в єдину групу;

– повторення – процедура триває доти, поки не буде досягнуто стабільного розбиття.

Лейден-алгоритм усуває проблему недостатньої деталізації кластерів в алгоритмі Лувена, оскільки дозволяє більш точно розподіляти вузли між кластерами, а також не дає слабко зв’язаним структурам залишатися в одному кластері (рис. 2.4).

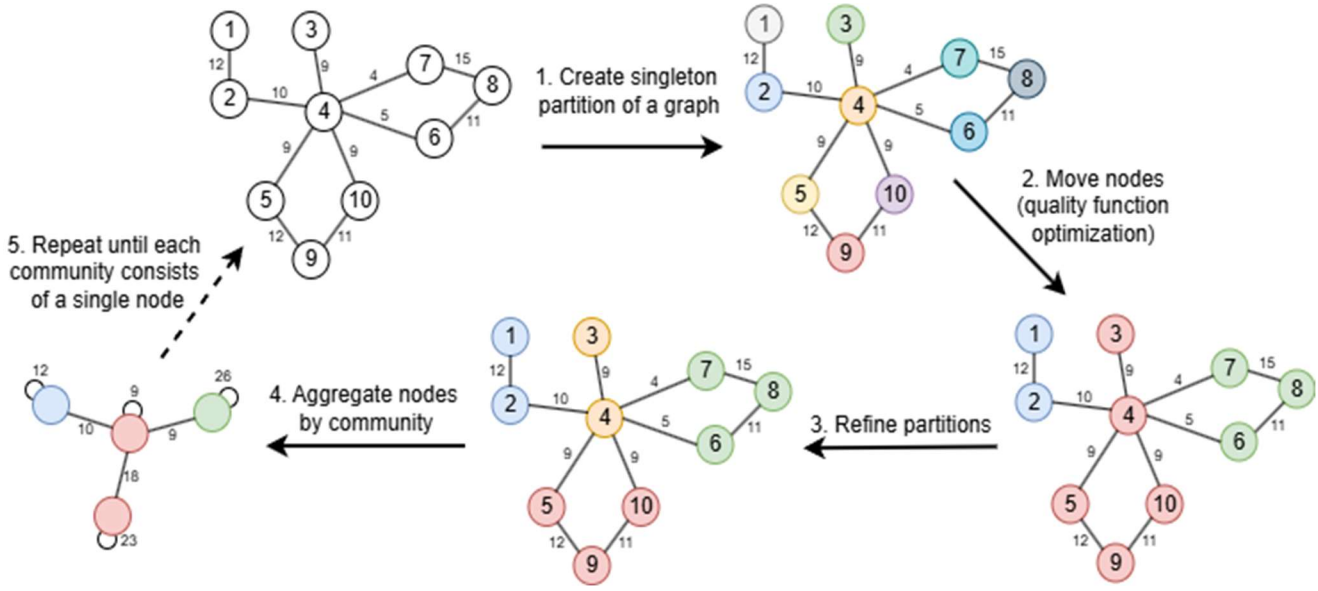


Рисунок 2.4 – Основні етапи роботи алгоритму Лейдена

Це досягається шляхом введення параметра роздільної здатності γ у функцію модульності, що дозволяє динамічно регулювати рівень деталізації кластерної структури в розподілених телекомунікаційних мережах. Такий підхід забезпечує оптимальний баланс між точністю виявлення кластерів і стабільністю їхньої структури, запобігаючи надмірній фрагментації або укрупненню, що є критично важливим для ефективного балансування навантаження та забезпечення сталої взаємодії між вузлами мережі.

Запропоноване удосконалення алгоритму Лувена дозволяє розробити метод ієрархічної кластеризації вузлів за рахунок підвищення якості кластеризації та оптимального розподілу ресурсів у розподілених телекомунікаційних системах.

Основними перевагами запропонованого методу є:

– підвищена точність виявлення кластерів завдяки адаптивному підходу до кластеризації;

– гнучке налаштування параметрів кластеризації, включаючи цільові обмеження розмірів кластерів та крок їх деталізації;

– зниження міжкластерних затримок за рахунок врахування зв’язності та пропускної здатності між вузлами;

Метод включає наступні етапи.

1 Етап. Представлення телекомунікаційної системи у вигляді зваженого графа $G = (V, E)$, де:

– V – множина вузлів, що відповідають обчислювальним елементам системи;

– E – множина ребер, зв’язків між вузлами, що характеризуються визначеними ваговими параметрами [76, 135].

Кожен вузол v_i має набір характеристик, а саме:

– мітка T_{ij}, що позначає тип задач, що може виконувати;

– обчислювальна продуктивність P_{ij} вузла.

Кожне ребро e_{ij} між вузлами v_i та v_j характеризується параметрами:

– w_{ij} – пропускна здатність каналу зв’язку між i та j;

– d_{ij} – затримка передачі даних.

Оскільки алгоритми Лувена та Лейдена оптимізують модульність графа, необхідно правильно визначити вагу ребер між вузлами, яка одночасно відображатиме затримку та пропускну здатність. Різні підходи до визначення такої ваги наведені в роботах [50,73,148].

1. Обернена пропускна здатність, зважена на затримку:

$$w(e) = \frac{d_{ij}}{w_{ij}} \tag{2.1}$$

Використовується у тих випадках, коли основною задачею є мінімізація затримки.

2. Гармонічне середнє пропускної здатності та оберненої затримки:

$$w(e) = \frac{2 \cdot w_{ij} \cdot {}^{1}/_{d_{ij}}}{w_{ij} + {}^{1}/_{d_{ij}}} \tag{2.2}$$

Дозволяє збалансувати вплив між швидкими та повільними зв’язками, придатний для гетерогенних мереж, але вразливий до дуже малих затримок – може переоцінювати зв’язки з низьким d_{ij}.

3. Нормалізована оцінка якості зв’язку:

$$w(e) = \alpha \cdot d_{ij} + \beta \cdot w_{ij} \tag{2.3}$$

Гнучко налаштовує ваги за допомогою коефіцієнтів α,β, що дає змогу адаптуватися в умовах змінної мережевої топології. Але сам вибір коефіцієнтів повинен бути коректним, щоб врахувати баланс між затримкою та пропускною здатністю, запобігти переоцінці зв’язків із високою пропускною здатністю, а також мінімізувати вплив аномальних затримок.

4. Ефективна швидкість передачі:

$$w(e) = \frac{w_{ij}}{1 + d_{ij}} \quad 2.4$$

Запобігає діленню на нуль і коректно обробляє зв'язки з дуже низькими затримками.

Для даного дослідження було обрано підхід 2 з ваговою функцією у вигляді лінійної комбінації затримки та пропускної здатності мереж. А вплив малих затримок знижено завдяки попередній нормалізації вхідних даних.

На основі цих параметрів будується початковий граф, який підлягає кластеризації.

2 Етап. Розбиття графа на шари

На цьому етапі вузли розподіленої системи класифікуються за типами завдань [76,135]. Це дозволяє створити багаторівневу структуру, де кожен рівень відповідає певному типу обчислювальних процесів. Такий підхід спрощує подальшу кластеризацію, забезпечує локалізацію обчислень і мінімізує міжкластерні витрати.

Для класифікації вузлів вводиться множина S рівнів обчислювального процесу, де кожен рівень $s_k \in S$ визначається множиною задач, що можуть виконувати вузли цього рівня. Визначається за критерієм сумісності задач:

$$s_k = \{v_i | T(v_i) \cap L(s_k) \neq \emptyset\} \quad 2.5$$

де L_{S_k} – множина задач, що належать рівню s_k.

Це рівняння означає, що вузол v_i належить рівню s_k, якщо його задача співпадає хоч з одною із задач, закріплених за цим рівнем.

3 Етап. Рекурсивна кластеризація кожного шару.

Кластеризація на кожному рівні здійснюється за модифікованим алгоритмом Лувена, який прагне досягти цільових показників розмірів кластерів шляхом рекурсивного їх виявлення. Формула модульності, яку використовує стандартний алгоритм Лувена, не містить параметрів, які б дали змогу регулювати зв'язність вихідних кластерів.

Для розв'язання цієї проблеми, в алгоритмі використовується модифікована формула модульності – Reichardt Bornholdt Potts Model, у якій визначено параметр роздільної здатності γ [50, 73, 123, 148].

Блок-схему алгоритму, що реалізує етапи наведеного методу, представлено на рис. 2.5.

Регулюючи показник модульності у процесі роботи алгоритму, можна досягти більш детального розбиття на кластери, оскільки застосовуються різні значення параметра роздільної здатності до різних частин графа. Модульність Q з врахуванням γ розраховується за формулою:

$$Q(v) = \frac{1}{2m}\sum_{i,j}\left[A_{i,j} - \gamma\frac{k_i k_j}{2m}\right]\delta(c_i, c_j), \qquad 2.6$$

де γ – параметр, що контролює рівень деталізації кластерів;

$\gamma > 1$ – сприяє створенню дрібніших кластерів;

$\gamma < 1$ – сприяє утворенню більших кластерів.

$A_{i,j}$ – матриці суміжності, що визначає наявність зв'язку між вузлами;

$k_i k_j$ – ступені вузлів i та j (загальна кількість зв'язків у графі);

m – загальна вага всіх зв'язків у графі;

c_i, c_j) – відповідні кластери для вузлів i , j;

$\delta(c_i, c_j)$ – індикатор того, чи вузол належить до кластера чи ні.

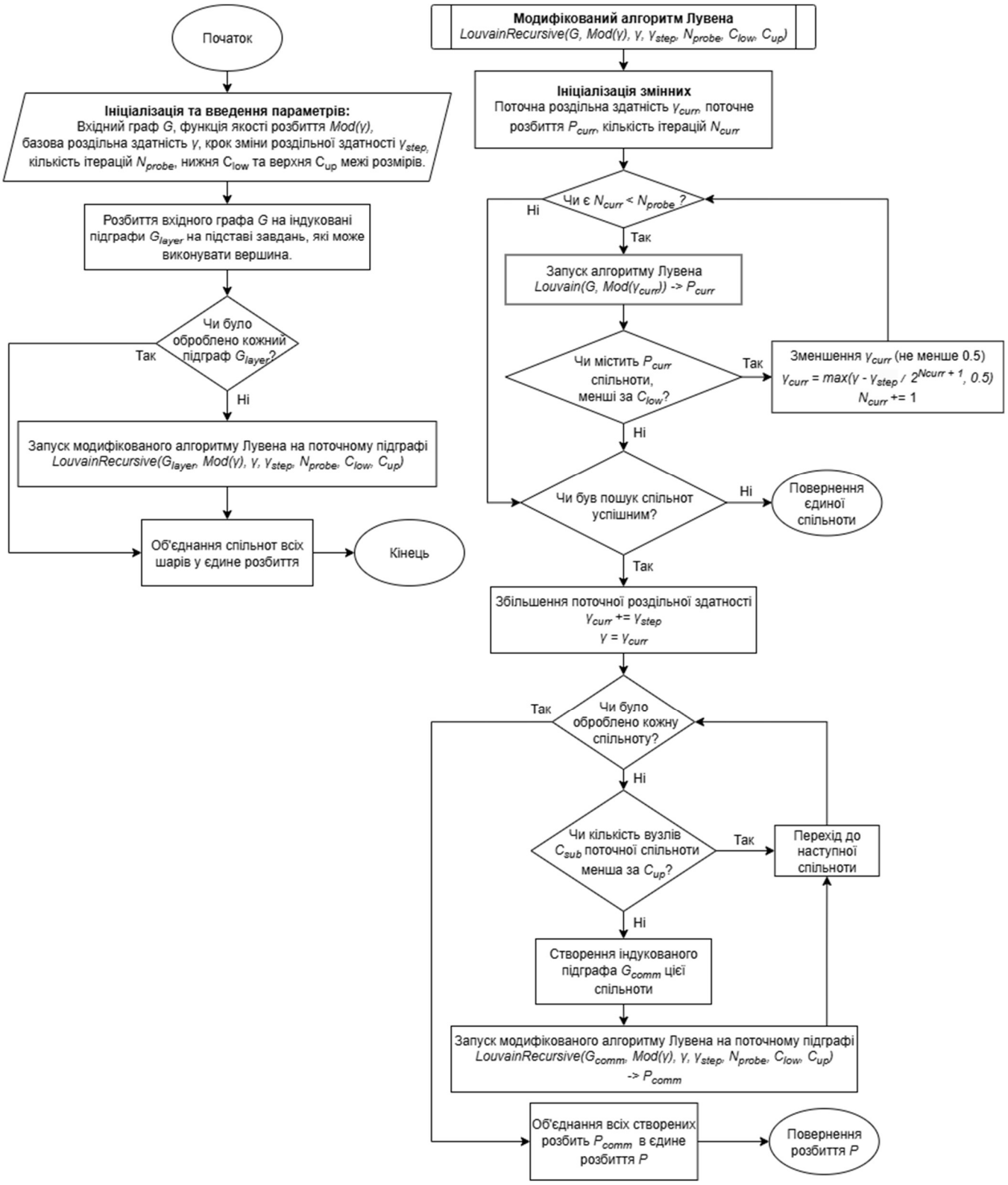


Рисунок 2.5 – Блок-схема модифікованого методу ієрархічної кластерізації

Рекурсивна кластеризація на основі модифікованого алгоритму Лувена включає наступні кроки.

3.1 Ініціалізація вхідних даних, параметрів та змінних.

Вхідні дані: граф G, функція якості розбиття $Mod(\gamma)$, базове значення параметра роздільної здатності γ, крок зміни роздільної здатності γ_{step}, кількість ітерацій пошуку кластерів N_{probe}, а також граничні обмеження розмірів кластерів: нижня межа C_{low} і верхня межа C_{up}.

Внутрішні змінні: поточне значення роздільної здатності Υ_{curr}, поточне розбиття графа P_{curr}, лічильник ітерацій пошуку кластерів N_{curr}.

3.2 Виконання стандартного алгоритму Louvain.

До поточного графа G застосовується алгоритм Louvain з використанням моделі Reichardt Bornholdt Potts Model для обчислення функції кластеризації, де параметр Υ_{curr} виконує роль роздільної здатності. Це формує початкове розбиття P_{curr}.

3.3 Перевірка мінімального розміру кластерів.

Якщо хоча б один отриманий кластер має розмір менший за C_{low}, то параметр Υ_{curr} зменшується за формулою:

$$\Upsilon_{curr} = \max(\Upsilon - \frac{Y_{step}}{2^{N_{curr}+1}}, 0{,}5), \qquad 2.7$$

Ця операція запобігає надмірній фрагментації кластерів і дає змогу підібрати оптимальне значення роздільної здатності для поточного графа. Значення роздільної здатності 0,5 – мінімальний поріг роботи алгоритму без прояву аномальної поведінки.

3.4 Перевірка успішності пошуку кластерів.

Якщо пошук кластерів був успішним, значення Υ_{curr} збільшується на Y_{step}, а якщо ні, то алгоритм повертає поточне розбиття P_{curr} як єдину спільноту.

3.5 Рекурсивна обробка великих кластерів.

Для кожного отриманого кластера виконується перевірка розміру:

– якщо розмір кластера C_{sub} менший за C_{up}, він залишається без змін;

– якщо розмір перевищує C_{up}, то створюється індукований підграф G_{comm}, що містить лише вузли цього кластера. Далі, на цьому підграфі знову виконується LouvainRecursive, що дозволяє деталізувати великі кластери.

3.6 Агрегація всіх отриманих кластерів у єдине розбиття.

Усі отримані часткові розбиття поточного рівня P_{comm} об'єднуються у єдине розбиття P, яке повертається на рівень вище, поки не дійде до остаточного виходу з функції.

3.7 Формування фінальної кластерної структури.

Отримані кластери кожного шару агрегуються у фінальну кластерну структуру. Так, результуюче розбиття враховує топологічні особливості мережі та її параметри, динамічно адаптує кластеризацію до заданих параметрів розмірності.

2.4 Експериментальна оцінка ефективності методу багатокрокової ієрархічної кластеризації на основі модифікованого алгоритму Лувена

Для оцінки ефективності запропонованого методу ієрархічної кластеризації було розроблено програмну реалізацію алгоритмів Лувена, Лейдена та модифікованого Лувена за допомогою мови програмування Python та бібліотеки NetworkX (Додаток В). Для збільшення достовірності експерименту було згенеровано граф розміром 2000 вузлів, структура якого максимально наближена до реальної розподіленої телекомунікаційної мережі [74].

Для візуалізації отриманого графа і спільнот було використано силовий алгоритм Фрухтермана-Рейнгольда, щоб підкреслити візуальне угруповання вузлів, якщо між ними визначено сильний зв'язок [80].

Проведене моделювання дозволило отримати графове представлення деякої РТС, на основі якої було виконано розбиття вузлів на кластери, відповідно до різних підходів (рис. 2.6-2.8).

На цих рисунках вершини графів (кольорові точки) відповідають обчислювальним або комунікаційним елементам телекомунікаційної системи (сервери, маршрутизатори, мережеві пристрої), а ребра (лінії відповідного кольору) – каналам передачі даних з урахуванням затримок та пропускної здатності. Кольори вузлів відображають кластерну структуру, отриману в результаті кластеризації: вузли одного кольору належать одному кластеру. Ребра, що мають колір відповідного кластера, ідентифікують внутрішньокластерні зв’язки, тоді як чорні лінії позначають міжкластерні з’єднання між вузлами з різних кластерів.

На рис. 2.6 представлено граф РТС з визначеними кластерами, що був отриманий в результаті кластеризації за алгоритмом Лувена.

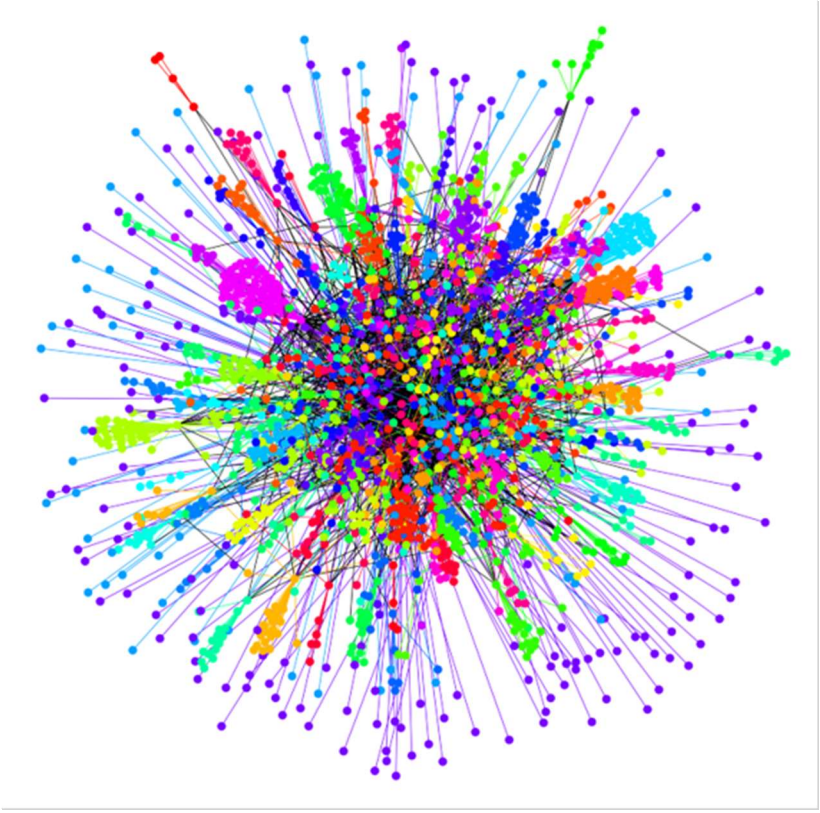


Рисунок 2.6 – Кластеризація вузлів за алгоритмом Лувена

У центрі графа розташовані кластери зі щільними зв’язками, а на периферії – менш пов’язані вузли, які формують довгі зв’язки з центральною частиною. Основною особливістю Лувен-кластеризації є схильність до утворення великих

кластерів, навіть якщо між ними немає достатньо сильного зв’язку. Це є наслідком жадібного підходу алгоритму. Введення параметра роздільної здатності дає змогу частково регулювати вимоги до зв’язності кластерів, проте не усуває проблему повністю: за низьких значень параметра кластери залишаються надто великими, а за високих – розбиваються нерівномірно.

На рис. 2.7 представлено граф PTC, кластеризований за алгоритмом Лейдена, який показує, що цей алгоритм створює більш рівномірно розподілені кластери, ніж за алгоритмом Лувена, завдяки чому структура системи стає менш централізованою.

На периферії графа так само залишаються вузли, з’єднані довгими зв’язками із деякою центральною частиною.

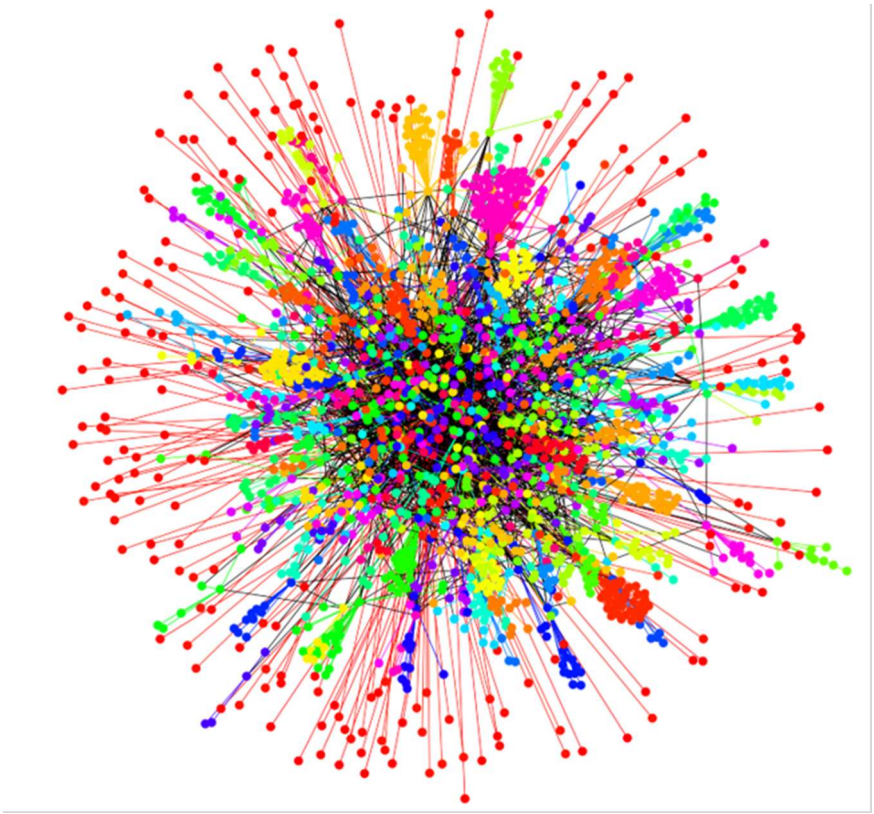


Рисунок 2.7 – Кластеризація вузлів за алгоритмом Лейдена

Однією з особливостей алгоритму є характер його роботи на графах з чітко вираженою кластерною структурою. Вже на малих значеннях роздільної здатності Leiden знаходить близьке до оптимального рішення, і подальше

підвищення цього параметра дає лише незначне збільшення фрагментації. Проте залишається одна з проблем алгоритму Louvain, а саме, кластери з високою щільністю зв’язків залишаються нерозділеними навіть за високих значень роздільної здатності, тоді як слабко зв’язані області продовжують фрагментуватися. Це призводить до нерівномірного розподілу кластерів, що може впливати на ефективність обчислень у РТС.

Одним із варіантів розв’язання цієї проблеми є використання функції модульності Constant Potts Model (CPM), яка робить поведінку алгоритму Лейдена подібною до Лувена, але з урахуванням запропонованих модифікацій [50, 73, 148].

На рис. 2.8 показано кластеризацію графа РТС, виконану за допомогою удосконаленого методу, що адаптивно змінює параметр роздільної здатності та мінімізує внутрішньокластерні та міжкластерні затримки.

Рисунок 2.8 – Кластеризація вузлів за модифікованим методом

На відміну від алгоритмів Лувена та Лейдена, запропонований метод формує збалансовані кластери, що запобігає як надмірному укрупненню

спільнот, так і їх зайвій фрагментації. Завдяки цьому досягається ефективний розподіл обчислювальних навантажень у мережі. Такий підхід до кластеризації враховує реальні характеристики телекомунікаційної системи та вимоги до неї, включаючи пропускну здатність каналів, продуктивність вузлів та цільові розміри кластеру. Це забезпечує стабільну взаємодію між кластерами, знижує затримки у міжкластерних обмінах даними та дозволяє знизити час прийняття рішень всередині кластеру.

Для оцінки ефективності методів кластеризації було проведено експериментальне моделювання при значенні роздільної здатності $\Upsilon = 1$ (результати представлені в табл. 2.4). На рис. 2.8–2.10 показано динаміку кількості виявлених кластерів, модульності та стандартного відхилення розміру кластерів відповідно до методу кластеризації та значення Υ.

Таблиця 2.4– Порівняльна оцінка результатів за різними методами кластеризації

Параметр	Лувен	Лейден	Запропонований метод
Кількість кластерів	50	215	137
Модульність	0,851892	0,811312	0,825021
Максимальний розмір кластера	184	66	64
Мінімальний розмір кластера	4	2	2
Середній розмір кластерів	40,000	9,302	12,664
Стандартне відхилення розміру кластерів	28,706	9,551	13.353
Середня внутрішньокластерна затримка	52,911	45,306	50,088
Середня міжкластерна затримка	111,927	95,644	102,523

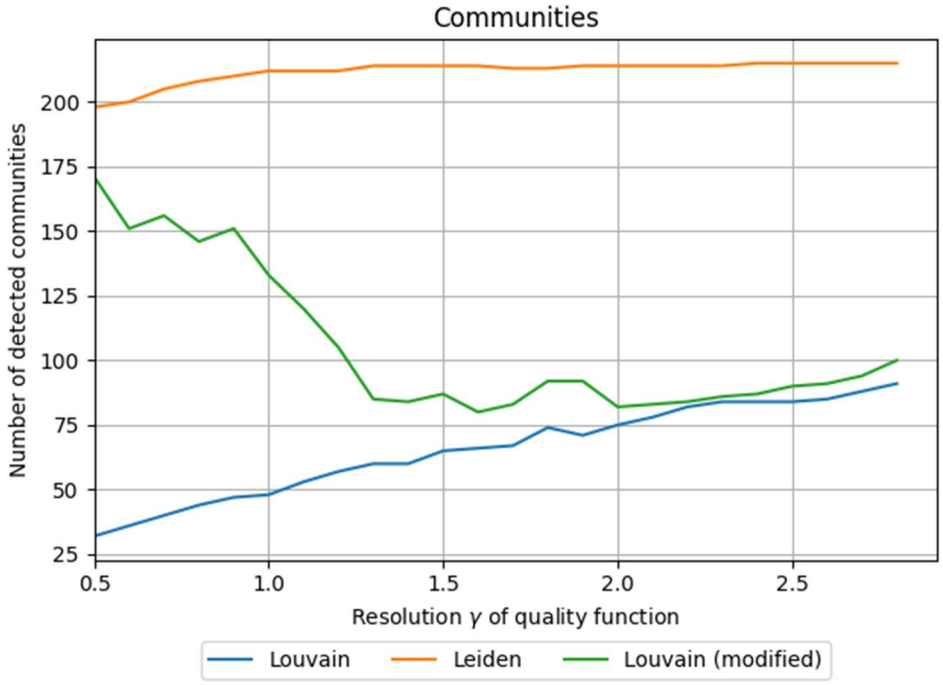


Рисунок 2.8 – Динаміка кількості кластерів за різними методами

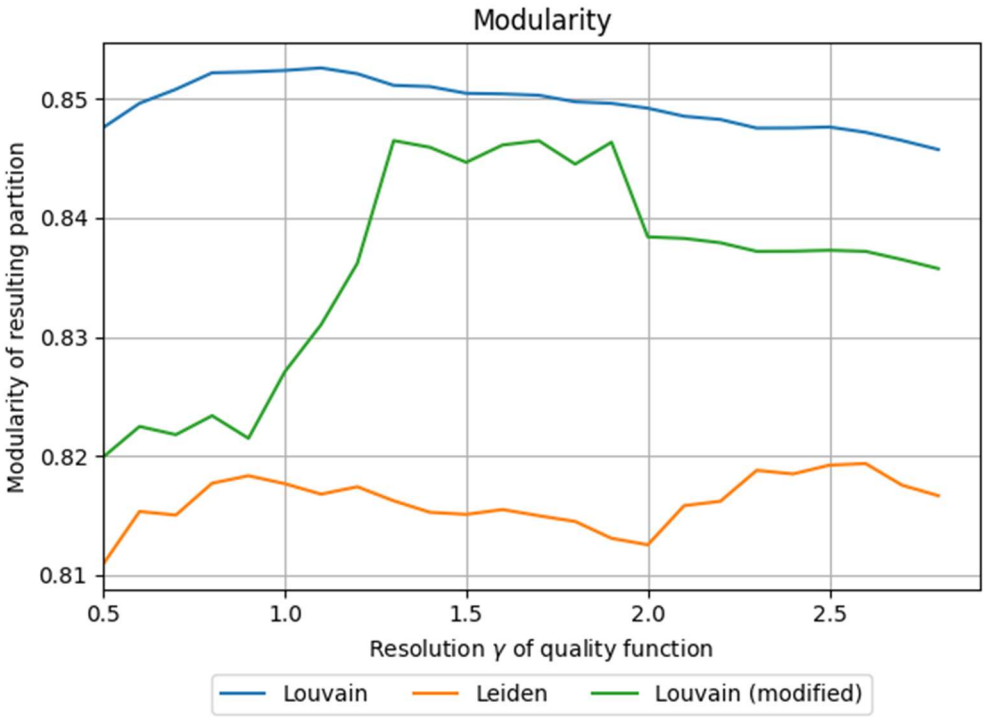


Рисунок 2.9 – Порівняння показника модульності за різними методами

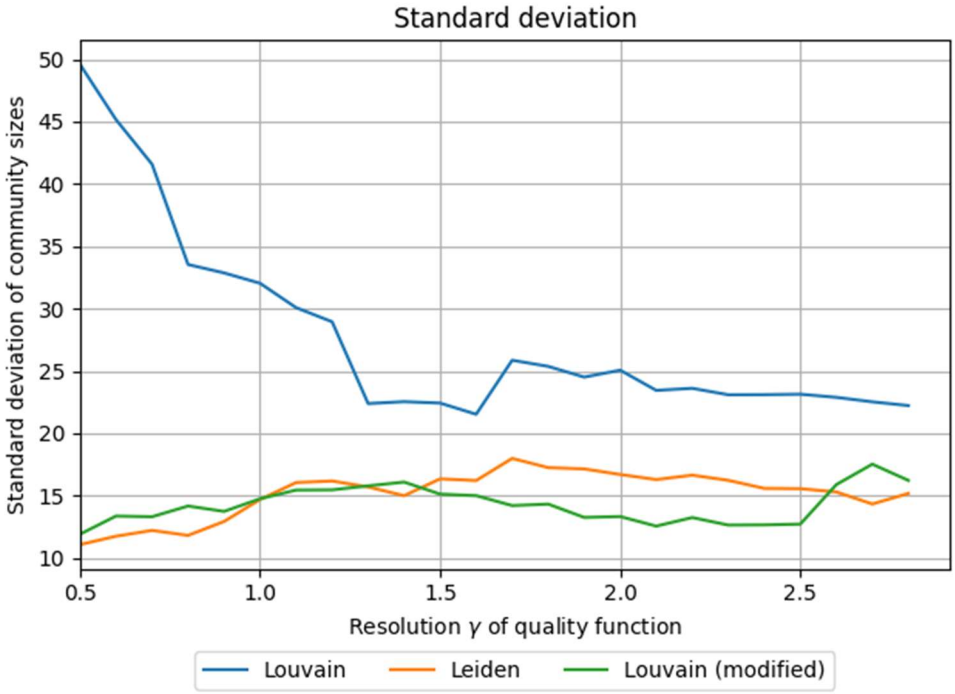


Рисунок 2.10 – Стандартне відхилення розмірів кластерів

Як видно з рис. 2.8–2.10 та результатів розрахунків табл. 2.4, запропонований модифікований метод має кращу збалансованість: стандартне відхилення розмірів кластерів у ньому на 53,5 % менше, ніж у класичного Лувена (13,353 проти 28,706), а модульність на 1,7 % вища, ніж у метода Лейдена (0,825021 проти 0,811312). За кількістю кластерів запропонований метод формує на 36,3 % менше, ніж у Лейдена, але на 174 % більше, ніж Лувена, що свідчить про його здатність уникати як надмірної фрагментації, так і надмірної агрегації.

На рис. 2.11–2.12 представлено динаміку середньої внутрішньокластерної та міжкластерної затримки за різними методами кластеризації та параметра роздільної здатності ϒ.

Як видно з рис. 2.11–2.12 та табл. 2.4, запропонований модифікований метод показує кращу збалансованість за затримками порівняно з класичним Louvain. Зокрема, середня внутрішньокластерна затримка в ньому на 5,3 % менша, ніж у Louvain (50,088 мс проти 52,911 мс), що свідчить про тіснішу логічну зв’язаність вузлів у межах кластерів.

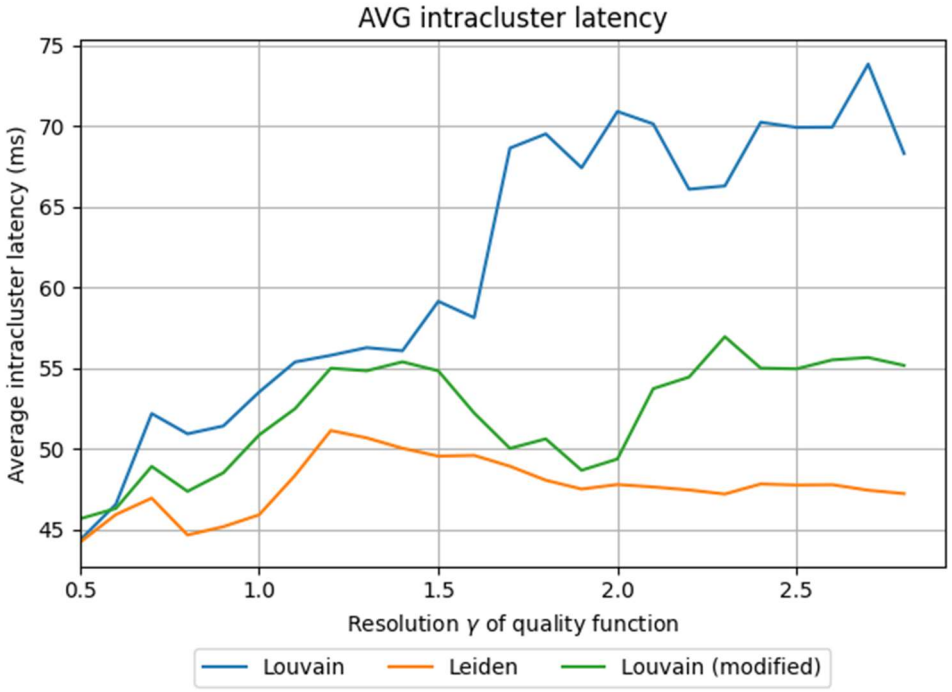


Рисунок 2.11 – Оцінка середньої внутрішньокластерної затримки

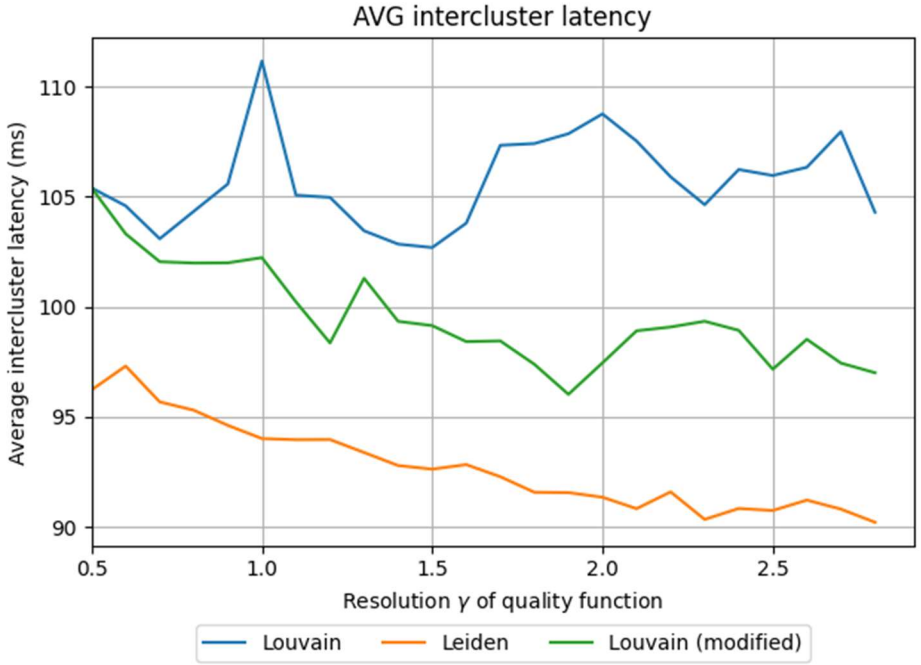


Рисунок 2.12 – Оцінка середньої міжкластерної затримки

Водночас, міжкластерна затримка зменшена на 8,4 % порівняно з Louvain (102,523 мс проти 111,927 мс), що вказує на зменшення загального навантаження на міжкластерну маршрутизацію.

За результатами експерименту можна зробити наступні загальні висновки.

1. Метод Лувен формує 50 кластерів, що призводить до великих спільнот із найвищою модульністю (0,851892). Проте показник міжкластерної затримки сягає 111,927 мс, що на 17,1 % більше, ніж у Лейден, та на 4,6 % більше, ніж у запропонованому методі. Модифікований підхід забезпечує кращий баланс між модульністю (0,825021), кількістю кластерів (137) та затримками, що робить його ефективним для РТС з обмеженими ресурсами, локальною взаємодією вузлів і потребою в стійких, але гнучко реконфігурованих кластерах.

2. Лейден створює 215 кластерів (у 4,3 раза більше, ніж Лувен), що дозволяє зменшити міжкластерні затримки на 14,5 %. Проте це супроводжується фрагментацією структури: максимальний розмір кластерів на 64,1 % менший, а стандартне відхилення – на 66,7 % нижче, ніж у Лувен. Така надмірна деталізація знижує модульність (0,811312) і може ускладнити управління у великих РТС, де важлива структурна цілісність та зменшення кількості міжкластерних з’єднань.

3. Модифікований метод забезпечує 137 кластерів, що на 174 % більше, ніж у Лувен, але на 36,3 % менше, ніж у Лейден. Це дозволяє знизити міжкластерні затримки на 8,4 % порівняно з Лувен, уникаючи надмірної деталізації Лейден. Крім того, внутрішньокластерні затримки на 5,3 % нижче, ніж у Лувен, що зберігає структурну згуртованість кластерів без втрати ефективності. Такий баланс показників робить запропонований метод доцільним для РТС з вимогами до гнучкої реконфігурації, помірної фрагментації та низьких міжкластерних затримок.

4. Експериментально доведено, що запропонований метод ієрархічної кластеризації та його алгоритм реалізації є універсальними і відкривають можливості побудови різних топологій, у тому числі, на основі георозподілу.

5. За рахунок підвищення показника роздільної здатності створюються спільноти з більшою кількістю кластерів (меншого розміру). Але, цей процес може відбуватися нерівномірно: у разі використання великого значення параметра роздільної здатності (10-20 і вище) – експериментально доведено, що формуються кластери з розміром 1, які не можна практично застосувати, тоді як

кластери з високою зв’язністю так і залишаються нерозділеними. Їх розділення відбувається за значно вищого параметра роздільної здатності (на рівні 50-100), але, при цьому, вже більша частина графа матиме кластери розміром 1-2.

6. Поєднання алгоритму Лувена з ієрархічною структурою дозволяє застосовувати кластеризацію не тільки на повному графі мережі системи, але і на окремих його сегментах. Це дає можливість оптимізовано перебудувати частину кластерів мережі без впливу на ті кластери, які вже досягли достатнього рівня оптимізації.

Висновки за розділом 2

1. Проаналізовано сучасні архітектурні підходи до побудови РТС із позицій масштабованості, адаптивності та мінімізації затримок. Показано, що класичні протоколи (VLAN, 802.1Q) мають обмеження у масштабуванні, тому доцільним є впровадження тунельних технологій (VXLAN, GRE, Geneve), які забезпечують ефективну взаємодію між вузлами у багаторівневих структурах із підтримкою шифрування та резервування.

2. Розроблено загальний метод ієрархічної кластеризації обчислювальних вузлів у РТС. Метод враховує неоднорідність продуктивності вузлів, параметри мережевих з’єднань (затримки, пропускна здатність), динамічну топологію та орієнтований на підвищення гнучкості, адаптивності й ефективності функціонування РТС. Метод реалізується поетапно: початкова кластеризація вузлів із урахуванням характеристик ресурсів і зв’язків; оптимізація внутрішньокластерної взаємодії; визначення головних вузлів для міжкластерної маршрутизації; багаторівневе балансування навантаження; динамічне переназначення вузлів; стратегії резервування, енергоефективності та захисту.

3. Обґрунтовано реалізацію в межах даного дослідження окремих етапів комплексного методу самоорганізації РТС: формування кластерів із цільовими характеристиками на основі модифікованого алгоритму Лувена (розділ 2), вибір головних вузлів і побудову обчислювальних конвеєрів (розділ 3),

інтелектуальний розподіл потоків між кластерами із використанням еволюційної багатокритеріальної оптимізації (розділ 4).

4. Етап первинного формування кластерів у межах комплексного методу самоорганізації реалізовано за допомогою розробки удосконаленого методу, який базується на алгоритмі Лувена, але доповнений механізмом динамічного регулювання параметра роздільної здатності і рекурсивним уточненням кластерної структури. На відміну від відомих підходів, зокрема алгоритму Лейдена, запропоноване рішення забезпечує більш керовану деталізацію кластерів, дозволяє уникнути надмірної агрегації вузлів та підтримує стабільну продуктивність за умов змін топології та навантаження.

5. Проведено експериментальне моделювання кластеризації на графах реалістичного розміру (2000 вузлів). Запропонований метод показав кращий баланс між кількістю кластерів (на 174 % більше, ніж у Лувен, і на 36 % менше, ніж у Лейден), модульністю (на 1,7 % вища за Лейден), розміром спільнот і затримками. Досягнуто зниження міжкластерної затримки на 8,4 % і внутрішньокластерної – на 5,3 % у порівнянні з Лувен, а стандартне відхилення розмірів кластерів зменшено на 53,5 %. Запропонована модифікація забезпечить стабільну та адаптивну роботу РТС в умовах нерівномірного навантаження.

6. Запропонований модифікований метод довів здатність підтримувати ефективну взаємодію між вузлами в умовах динамічних змін топології, нерівномірного трафіку та обмежених ресурсів. Це дозволяє забезпечувати стабільну продуктивність РТС, сприяє локальній реконфігурації кластерів без потреби в глобальному переформуванні мережі та знижує витрати на енергоспоживання та міжкластерну маршрутизацію, що є критично важливим для високонавантажених РТС.

РОЗДІЛ 3
РОЗРОБКА ІНТЕГРОВАНОГО МЕТОДУ КЕРУВАННЯ ІНФОРМАЦІЙНИМИ ПОТОКАМИ З ЗАБЕЗПЕЧЕННЯМ ВИБОРУ ГОЛОВНОГО ВУЗЛА ТА ФОРМУВАННЯМ КОНВЕЄРІВ ОБРОБКИ

Розроблений у розділі 2 метод багатокрокової ієрархічної кластеризації забезпечує динамічне групування обчислювальних вузлів оптимальне для вирішення конкретних задач, але для забезпечення ефективного управління отриманими кластерами необхідним наступним етапом дослідження є розробка інтегрованого методу керування інформаційними потоками, який включає вибір головного вузла та формування оптимальних конвеєрів обробки.

У сучасних розподілених телекомунікаційних системах (РТС) висуваються підвищені вимоги до швидкодії та надійності обробки даних у режимі реального часу. У зв’язку з цим особливої актуальності набувають питання раціонального управління обчислювальними ресурсами, ефективного вибору координаторів кластерів та побудови стійких обчислювальних конвеєрів. Аналіз сучасних підходів [39, 57, 58, 113] свідчить про доцільність використання алгоритмів вибуру лідера для забезпечення узгодженості в розподілених середовищах, однак більшість таких рішень орієнтовані на формальні моделі або класичні алгоритми (протоколи) без врахування специфіки кластеризованих РТС із затримками та обмеженими ресурсами. Низка досліджень [41, 67, 82, 87, 91, 93, 95, 96] описують класичні та ймовірнісні алгоритми вибору лідера, а саме: від кільцевої топології до самоорганізованих протоколів, але більшість із них вимагають глобальної координації, не забезпечують швидкого локального відновлення після відмови або супроводжуються значними витратами.

Попри окремі спроби удосконалити ці підходи, зокрема шляхом зменшення кількості повідомлень у кільцевих алгоритмах [40] чи використання гібридних моделей, таких як поєднання Raft з алгоритмом - консенсусу на основі PoW [149], проблема вибору координатора у динамічних умовах, із обмеженими ресурсами та затримками передачі, залишається відкритою.

Виходячи з вищезазначеного, набуває актуальності розробка асинхронно-узгодженого, масштабованого та ресурсоощадного методу вибору головного вузла, здатного до адаптивної роботи в умовах кластеризованої архітектури з потоковою обробкою даних.

Для реалізації науково-практичних завдань даного дослідження розглянуто розподілену обчислювальну систему, основною метою якої є забезпечення безперервної потокової обробки даних у покроковому режимі. Вихідні дані генеруються кінцевими пристроями, а результати обробки мають бути доставлені до хмарного середовища або дата-центру для подальшого зберігання чи аналізу.

Кожен етап обробки даних у системі має унікальний порядковий номер, а виконання етапів здійснюється у суворо визначеній послідовності, яка формує обчислювальний конвеєр. Такий конвеєр представляє собою впорядковану сукупність завдань (наприклад, послідовність із чотирьох оброблюваних операцій), що реалізуються поетапно.

Архітектура та функціональні особливості моделі розподіленої телекомунікаційної системи визначаються через сукупність положень, які розкривають логіку взаємодії її компонентів.

1. Кластерна структура. Система складається з кластерів обчислювальних вузлів. Вузли в межах одного кластера розташовані близько один до одного, хоча не обов’язково перебувають в одній мережі або географічній зоні. Кожен вузол кластера здатен виконувати лише одну конкретну задачу з визначеного конвеєра обробки.

2. Ресурсна модель вузла. Кожен вузол має обмежені обчислювальні ресурси (процесора, оперативної пам’яті, сховища) та встановлені канали зв’язку (тунелі) з іншими вузлами свого кластера. Передбачається, що в межах одного кластера відсутні ізольовані або недоступні вузли.

3. Модель інформаційного потоку. Передача даних здійснюється через тунелі, які визначаються як інформаційні потоки. Потік характеризується постійною інтенсивністю, яка у певні моменти може спонтанно зростати,

формуючи короткотривалі сплески. При цьому обробка інформації з потоку споживає частину ресурсів вузла, а ці витрати також можуть досягати пікових значень синхронно зі збільшенням інтенсивності потоку.

4. Роль головного вузла. У кожному кластері визначається головний вузол, який виконує функції координатора: збирає метрики з інших вузлів та здійснює управління кластером. Цей вузол не бере участі в передачі інформаційних потоків, проте має повну інформацію про структуру потоків вузлів свого кластера.

5. Супервізор у хмарі. У хмарній частині системи функціонує супервізорський вузол, що виконує функції остаточного координатора для міжкластерних рішень, хоча перевага надається локальному, внутрішньокластерному керуванню.

З врахуванням специфіки архітектури [75, 105, 110] та функціонального призначення моделі, запропоновано метод, спрямований на розв’язання двох науково-практичних задач:

– визначення головного вузла (координатора) в межах кожного кластера з передбаченням потенційних резервних варіантів, при цьому пріоритет надається вузлам з нижчим обсягом ресурсів та меншою середньою затримкою до інших вузлів;

– побудова обчислювальних конвеєрів із мінімізацією міжкластерних затримок за умови збереження послідовності виконання завдань та дотримання ресурсних обмежень кожного кластера.

Отже, наступним кроком у побудові ефективної кластерної архітектури розподілених телекомунікаційних систем є розробка методу вибору координаторів, який забезпечуватиме стабільне керування ресурсами всередині кластерів, формування обчислювальних конвеєрів та узгоджену маршрутизацію інформаційних потоків з урахуванням топології, затримок і обмежень продуктивності вузлів.

3.1 Обґрунтування методу вибору головного вузла в кластеризованому середовищі розподілених телекомунікаційних систем

Представлена на рис. 3.1 архітектура РТС відображає основні структурні та функціональні особливості, які було враховано під час розробки методу вибору головного вузла та формування конвеєрів обробки.

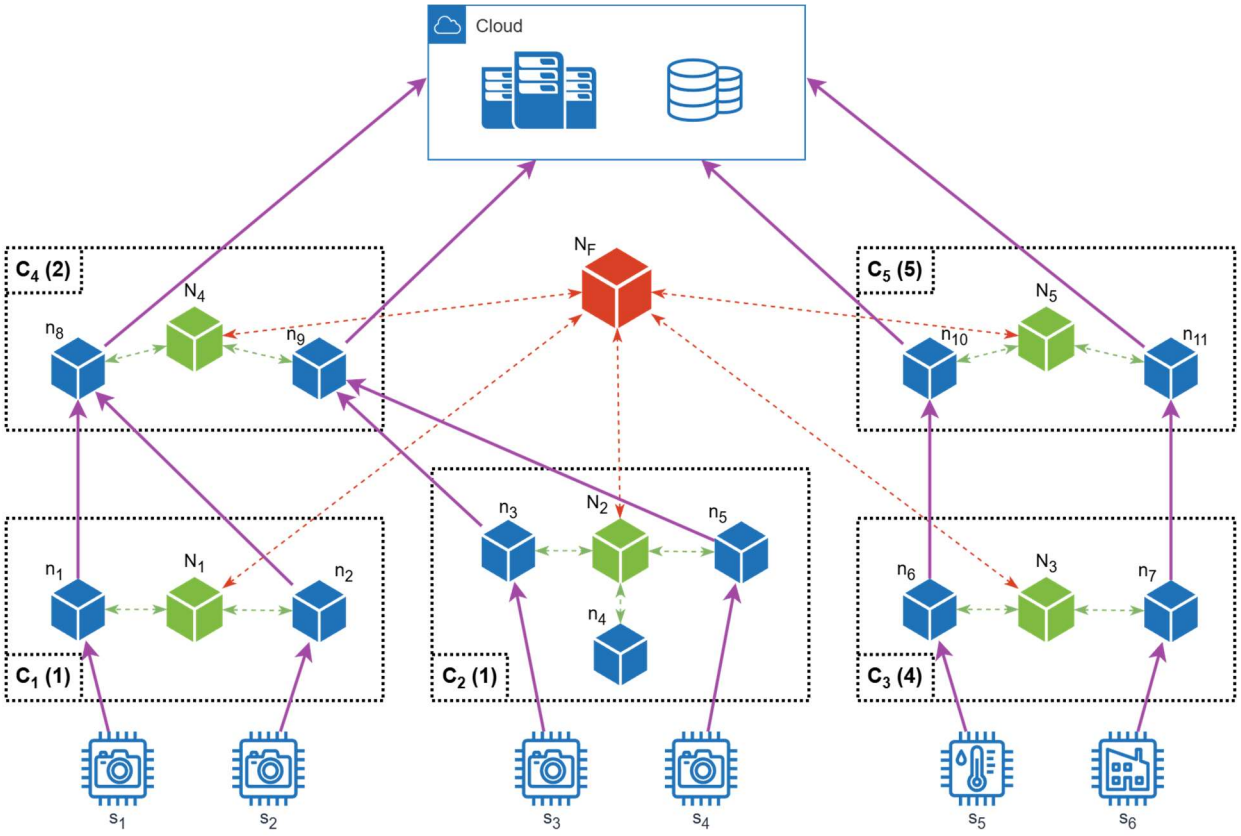


Рисунок 3.1 – Архітектура РТС з кластеризацією вузлів та централізованим супервізором

Позначення елементів на рис. 3.1.

– Сині квадрати з іконкою – сенсори $S_1 - S_6$. Вони є ініціаторами інформаційних потоків (безперервно генерують дані).

– Дрібний пунктир – межі кластерів обчислювальних вузлів. Позначення кластера вказує на його номер і, в дужках, номер задачі конвеєра, яку він вміє виконувати.

– Сині куби – обчислювальні вузли, які виконують обробку окремих етапів конвеєра.

– Зелені куби – локальні головні вузли (координатори кластерів $C_1 - C_5$).

– Червоний куб N_F – центральний супервізор, відповідальний за міжкластерну координацію.

– Фіолетові суцільні стрілки – напрямок інформаційних потоків (конвєєрів) від сенсорів $S_1 - S_6$

– Зелені пунктирні стрілки – з'єднання між обчислювальними вузлами та головним вузлом всередині кластеру.

– Червоні пунктирні стрілки – з'єднання між головними вузлами кластерів та центральним супервізором.

З урахуванням архітектури РТС в даному дослідженні сформульовано додаткові припущення:

1. Кластери є невеликими (5–60 вузлів), що забезпечується за рахунок попереднього групування на основі методу ієрархічної кластеризації обчислювальних вузлів у розподілених телекомунікаційних системах, який ґрунтується на модифікованому алгоритмі Louvain, що виконує багатокрокову кластеризацію графа з динамічним коригуванням параметрів [144].

2. У межах одного кластера забезпечується низька затримка, однак можлива часткова втрата пакетів унаслідок нестабільності з'єднань або розміщення вузлів у різних сегментах WAN.

3. Ключовою вимогою є мінімізація часу на відновлення керування кластером у разі відмови головного вузла.

Сформульовані в дослідженні припущення окреслюють обмеження та умови функціонування, в межах яких реалізується запропонований метод. У цьому контексті перша з визначених науково-практичних задач дослідження деталізується наступним чином.

1. Визначення критеріїв вибору головного вузла з множини кандидатів.

2. Розробка алгоритму вибору головного вузла.

Крім того, під час розроблення методу необхідно врахувати можливі варіанти роботи (функціонування) кластерів, що суттєво впливають на його ефективність, а саме.

1. Метод вибору головного вузла має забезпечити стабільну роботу як у випадку його відсутності (наприклад, під час ініціалізації або після відмови), так і у випадку наявності чинного координатора (для підтримання балансу при доєднанні нових вузлів).

2. Необхідно передбачити визначення «заступника» головного вузла для забезпечення швидкої та безперервної передачі повноважень у разі відмови головного вузла.

Для визначення пріоритетності вузла в процесі вибору головного вузла у даному дослідженні запропоновано показник – оціночна функція (Scoring Formula), яка дозволяє проводити ранжування кандидатів на роль координатора кластера. Ця функція розроблена з урахуванням трьох основних критеріїв, що враховують топологічні та ресурсні характеристики вузлів і дозволяють асинхронно-узгоджено розв’язувати ситуації з однаковими метриками.

Кожен критерій у запропонованій формулі нормалізується для приведення до уніфікованого масштабу, а також зважується відповідним коефіцієнтом значущості w_i в залежності від особливостей розподіленої архітектури. Загальний вигляд функції оцінки вузла i описується формулою:

$$Score(i) = w_1 \cdot AvgLatency(i) + w_2 \cdot NormalizedPower(i) + w_3 \cdot NormalizedID(i) \quad (3.1)$$

де – $w_1, w_2, w_3 \in [0,1]$, $w_1 + w_2 + w_3 = 1$.

1. Середня затримка до інших вузлів. Цей критерій відображає топологічну близькість вузла до інших учасників кластера, що важливо для зменшення затримок при координації. Визначається за формулою:

$$AvgLatency(i) = \frac{1}{|N| - 1} \sum_{j \neq i} Latency(i, j), \quad 3.2$$

де $|N|$ – кількість вузлів у кластері.

2. Нормалізована ресурсна потужність вузла. Оцінює загальну обчислювальну здатність вузла на основі сумарного обсягу основних ресурсів:

$$Power(i) = CPU_i + RAM_i + STG_i, \quad 3.3$$

Оцінене значення потужності вузла нормалізується відносно максимальної потужності серед усіх вузлів кластера:

$$NormalizedPower(i) = \frac{Power(i)}{max_k Power(k)} \quad 3.4$$

В запропонованому методі вибору головного вузла перевага надається слабшим вузлам, тобто таким, які мають нижчу обчислювальну потужність. Це дозволяє зберігати потужні вузли для інтенсивних обчислень, а координаторські функції передавати менш продуктивним учасникам.

3. Нормалізоване значення ідентифікатора вузла. Це додатковий критерій, який використовується для вирішення ситуацій з однаковими оцінками, як додатковий механізм розв’язання «нічиї»:

$$NormalizedID(i) = \frac{ID_i}{max_k ID_k} \quad 3.5$$

Чим менше значення $Score(i)$, тим вищий пріоритет вузла для призначення його головним. Запропонована оціночна функція дозволяє заздалегідь здійснити ранжування всіх кандидатів у кластері без необхідності запуску повномасштабної процедури виборів. Але для забезпечення стійкого функціонування системи в умовах динамічних змін необхідно, окрім попереднього ранжування, застосовувати ефективний алгоритм досягнення консенсусу, здатний оперативно і без надлишкових затримок визначати новий головний вузол у разі відмови поточного або при виникненні змін у топології.

У табл. 3.1 узагальнено результати аналізу відомих алгоритмів вибору координатора: класичних [87, 93], ймовірнісних [91, 95], кільцевих [40, 67], вдосконалених [41], а також протоколів консенсусу [101, 113, 149], що дозволяє порівняти їх з авторською модифікацією на основі алгоритму «Пліткар» (Gossip).

Таблиця 3.1 – Порівняльна оцінка алгоритмів вибору лідера

Алгоритм	Принцип роботи	Переваги	Недоліки
Алгоритм хулігана (Bully Algorithm) [87, 93]	У разі відмови головного вузла, лідером стає вузол з найвищим ідентифікатором. Ініціалізація відбувається шляхом розсилки повідомлень «Вибори», всім активними вузлами.	Простий, швидкий в умовах стабільної мережі	Великий трафік при частому використанні. Чутливий до втрати повідомлень.
Швидкий алгоритм хулігана (Fast Bully Algorithm) [41]	Удосконалений алгоритм з меншою кількістю повідомлень під час виборів, але з кешуванням вузлами більшої кількості інформації.	Більш швидкий завдяки скороченню повідомлень; оптимальний для малих кластерів.	Вимагає гарантії правильного порядку передачі повідомлень.
Raft (тільки на етапі вибору лідера) [113]	Вузли «запускають» голосування з випадковими таймерами, перемагає той, хто першим набирає більшість голосів.	Дуже стійкий навіть до втрати повідомлень, розділів.	Складніший за алгоритм хулігана та надлишковий для вибору лише головного вузла.

Вибір на основі реплікації з відміткою про перегляд (VSR) [101, 149]	Полегшений аналог Raft: узгодження нового лідера після зміни представлення	Добре підходить для кластерів з високими вимогами до доступності	Складніший у реалізації; вимагає постійної активності усіх реплік
Вибір на основі кільця (Чанг і Робертс) [58,67]	Вузли, розташовані у логічному кільці, передають повідомлення «Вибори» по колу. Перемагає вузол з найкращою оцінкою	Прості, гарантовані вибори.	Повільний у великих кластерах, не підходить для оптимізації затримок.
«Пліткар» з асинхронно-узгодженим відбором *(*власна розробка)*	Вузли обмінюються метриками, а потім самостійно та асинхронно-узгоджено обирають найкращий вузол за єдиними критеріями.	Легкий трафік, без виборчого шторму, швидке відновлення завдяки наперед визначеним замінам	Потребує ефективного початкового розподілу метрики. Не обробляє довготривалі розділи самостійно.

Аналіз даних табл. 3.1 дозволяє зробити висновок, що з огляду на сформульовану мету дослідження – забезпечення ефективного вибору головного вузла та стійкого функціонування кластерів у розподілених телекомунікаційних системах, доцільним є використання наступних алгоритмів.

1. Швидкий алгоритм хулігана.

– ефективний для малих і середніх кластерів із відносно стабільними умовами з’єднання;

– забезпечує швидку збіжність при використанні ідентифікаторів вузлів;

– дозволяє легко впровадити механізм вузлів-заступників, що підвищує надійність у разі збоїв;

– працює краще за класичний алгоритм хулігана в умовах наявності тимчасових проблем у мережі;

Недоліком цього алгоритму є те, що для безперебійної роботи потрібні ідентифікатори/оцінки інших вузлів., що у фазі ініціалізації може призводити до трансляційного перевантаження («виборчого шторму»).

2. Алгоритм «Пліткар» (Gossip) з асинхронно-узгодженим відбором координатора. До переваг алгоритму належать:

– вибори не запускаються взагалі. У разі відмови головного вузла, інші вузли або заступник це виявляють і асинхронно-узгоджено визначають новий головний вузол;

– постійний, але дуже легкий трафік;

– вузли знають повний список заступників. Відповідає потребі в миттєвому призначенні заступників;

– швидке відновлення після збою;

– працює в кластерах, топологія яких не є повним графом або при наявності циклів. Завдяки цьому максимальний розмір кластера значно збільшується і алгоритм здатен обробляти більше практичних випадків.

З огляду на переваги вибору головного вузла без запуску класичних виборчих процедур було удосконалено класичний алгоритм «Пліткар» шляхом поєднання ресурсно-затримкової оцінки вузлів з урахуванням штрафів за перевантаження для асинхронно-узгодженого вибору координатора.

3.2 Удосконалення алгоритму «Пліткар» (Gossip) для асинхронно-узгодженого вибору головного вузла (координатора кластера)

З метою удосконалення алгоритму «Пліткар» розглянемо більш докладно його основні характеристики та етапи реалізації.

«Пліткар» – це клас протоколів (алгоритмів), заснованих на поступовому поширенні інформації між сусідніми вузлами за аналогією з епідемією [39, 41, 101]. «Пліткар» широко використовується в децентралізованих мережах, де відсутній постійний координатор, і є ефективним способом обміну даними без централізованого контролю [113, 147].

В дослідженні на основі підходу «Пліткар» запропоновано удосконалений алгоритм вибору головного вузла, який дозволяє уникнути широкомовного обміну повідомленнями та реалізувати визначення координатора без явної фази виборів. Основними принципами реалізації алгоритму є:

1. Періодичний обмін між вузлами базовими метриками або попередньо обчисленими оцінками.

2. Самостійне обчислення кожним вузлом оптимального кандидата на роль головного вузла на основі зібраних метрик.

3. Автоматичне призначення попередньо визначеного заступника у випадку відмови поточного лідера, що забезпечує практично миттєве відновлення керування кластером.

4. Обмеження розміру повідомлень до мінімально необхідних параметрів ({ID, оцінка або метрики, мітка часу}).

5. Обмін так званими heartbeat-повідомленнями (це повідомленнями про стан, або «серцебиття») між вузлами після визначення головного для підтвердження актуальності статусу.

Основні позначення показників алгоритму представлені в табл. 3.2.

Таблиця 3.2 – Показники (метрики) удосконаленого алгоритму Gossip

Символ	Значення
ID_n	Унікальний ідентифікатор вузла n. Визначається при ініціалізації і не змінюється
M_n	Метрики вузла n: затримка, загальна кількість ресурсів (або їх використання)
$Score(n)$	Обчислена оцінка вузла n на основі його метрик (формула 1)
C_n	Локальний список відомих вузлів кластера з актуальними метриками, отриманими через механізм поширення Gossip
R_n	Ранжований список вузлів на основі значень $Score(k)$, впорядкований у зростаючому порядку
$Master_n$	Вузол, який вважається головним для вузла n
$Substitutes_n$	Список альтернатив (заступників) вузла n, що можуть замінити координатора у разі його відмови

Блок-схема удосконаленого алгоритму «Пліткар» при нормальній роботі та при виявленні збою головного вузла представлена на рис. 3.2.

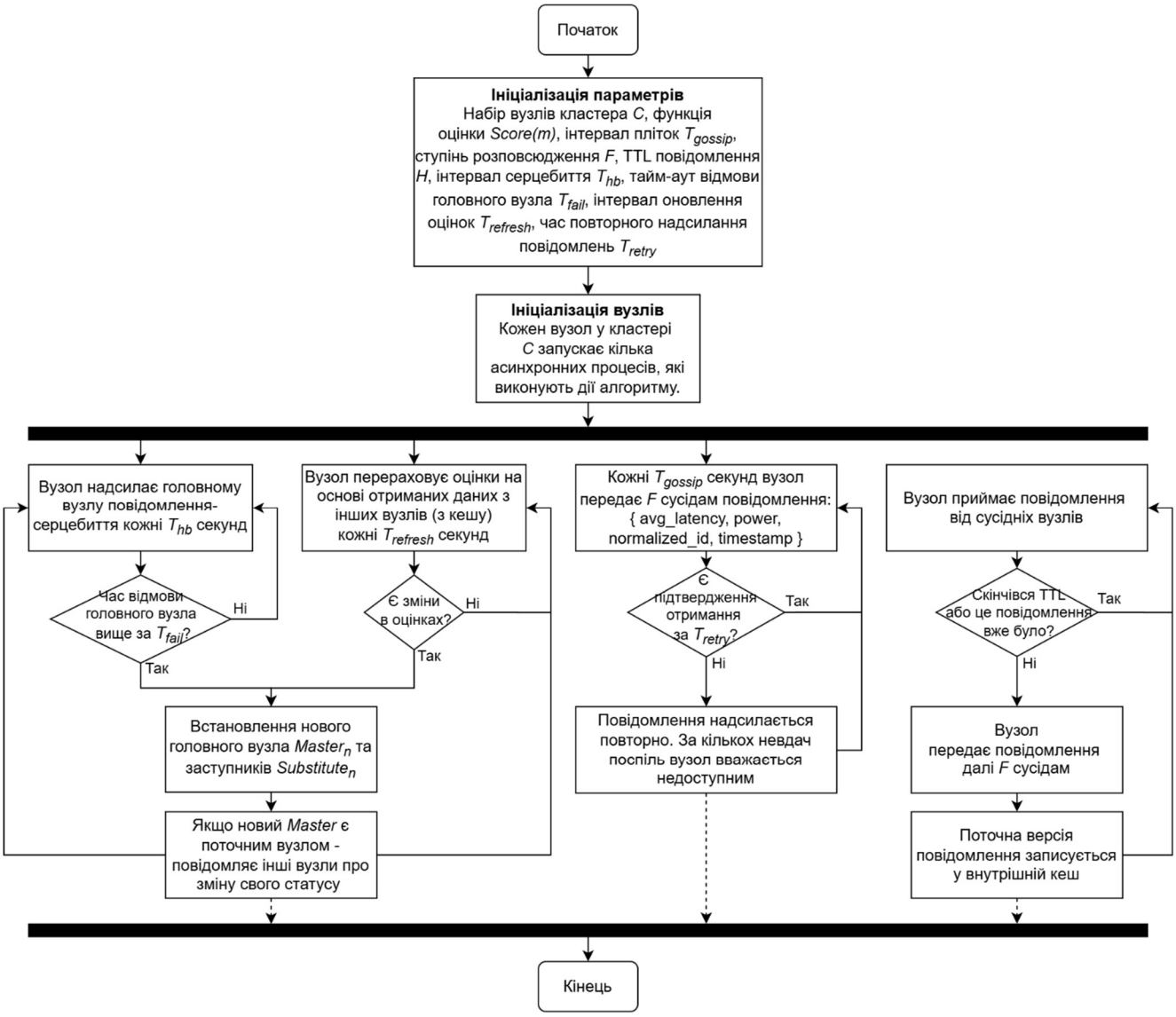


Рисунок 3.2 – Блок-схема удосконаленого алгоритму «Пліткар»

Як видно з рис. 3.2 процедура роботи алгоритму в нормальному режимі функціонування включає послідовність наступних дій. Розглянемо 2 сценарії: нормальний режим і при відмові головного вузла.

Кроки удосконаленого алгоритму «Пліткар» в нормальному режимі

1. Кожен вузол періодично передає («пліткує») свої поточні метрики, наприклад, з інтервалом 2–5 секунд.

2. Отримані від сусідів метрики вузол передає далі іншим вузлам (всім або певній їх кількості).

3. Вузол локально формує множину C_n (набір актуальних метрик відомих вузлів.) і незалежно:

– обчислює оцінку $Score(n)$для всіх вершин у C_n;

– сортує вершини в R_n (від кращої до гіршої);

– встановлює головний вузол $Master_n = R_n[0]$ та заступників (резервних кандидатів) $Substitutes_n = R_n[1:]$ (решта списку).

Таким чином, кожен вузол локально володіє інформацією про поточного координатора, а в разі його відмови миттєво призначає нового лідера із заздалегідь обчисленого списку.

У разі потреби можлива заміна метрик оцінками, однак це звужує гнучкість та універсальність застосування алгоритму.

Кроки удосконаленого алгоритму «Пліткар» при виявленні відмови головного вузла

У разі фіксації відсутності ознаки активності координатора (зникнення періодичних повідомлень типу «серцебиття»), відповідно до рис. 2 реалізується наступна послідовність дій:

1. Вузол із найвищим пріоритетом у списку $Substitutes_n$ негайно переходить до виконання функцій головного вузла $Master_n$.

2. (Опціонально) У разі критичних вимог до часу реакції новопризначений координатор може явно повідомити інші вузли про зміну свого статусу.

3. Усі вузли кластера:

– оновлюють головний вузол: $Master_n = Substitute$;

– відповідно зсувають список заступників.

Таким чином, зміна головного вузла відбувається без явного запуску процедури виборів, з мінімальними затримками та без генерування надмірного обміну повідомленнями. Контроль за станом координатора може здійснюватися або всіма вузлами кластера, або лише його резервними кандидатами.

Аналогічно, при динамічному приєднанні нових вузлів або при втраті частини вузлів, повторне обрання координатора не потребує явного запуску виборчої процедури. Новий лідер та оновлений список заступників формуються автоматично на основі повідомлень (gossip-метрик) в межах кластера.

Проте для забезпечення стабільної роботи алгоритму в умовах динамічної топології важливо врахувати потенційні ризики надмірного розповсюдження повідомлень та порушення узгодженості даних. З цією метою в запропонованому алгоритмі реалізуються спеціальні механізми контролю, що забезпечують ефективну обробку циклів та уникнення дублювання.

Однією з можливих проблем алгоритмів «Пліткар» є їхня робота в мережах з внутрішніми циклами. Без механізму переривання циклічної передачі, повідомлення може циркулювати в системі необмежену кількість разів. Крім того, є підтримка узгодженості даних, що циркулюють в системі (антиентропія). Для вирішення цих проблем можуть бути застосовані наступні підходи.

1. Використання версій або міток часу. Кожен вузол зберігає кеш останніх повідомлень (походження, версія або час), які він уже отримав. Якщо він отримує дублікат, то відкидає повідомлення. Це гарантує, що кожне оновлення поширюється лише один раз на кожний вузол.

2. Обмеження глибини поширення через TTL (англ. Time-To-Live – час життя). Кожне повідомлення «плітки» містить TTL (наприклад, 6-12 хопів). Кожен вузол, який пересилає повідомлення, зменшує TTL на 1. Якщо TTL досягає 0, то видаляє повідомлення. Корисно, якщо необхідно обмежити сферу дії і уникнути широкого розповсюдження.

3. Контрольні зведення по анті-ентропії**.** Кожні кілька раундів (наприклад, кожна 10 – та плітка) вузли надсилають стан (хеш або карту версій) того, що вони бачили. Інші вузли:

3.1 Порівнюють свій стан з отриманим станом.

3.2 Надсилають лише ті оновлення, яких бракує.

Це дозволяє уникнути надлишкового обміну даними та синхронізувати вузли, які відстали.

4. Протокол передачі пліток «передавання-стягнення» (англ. push–pull gossip protocol). передбачає двоступеневу модель обміну даними, яка є більш гнучкою та ефективною в умовах динамічної топології.

4.1 Фаза передавання (push-phase): вузол надсилає останні наявні оновлення до вибраних сусідів.

4.2 Фаза запиту (pull-phase): вузол опитує сусідів щодо наявності оновлень, яких йому бракує, з метою отримання лише необхідної інформації.

Чим більше підходів застосовується у системі, тим вища її стійкість і надійність, однак це збільшує її складність, ресурсоємність та навантаження на мережу. В запропонованого алгоритму використовуються підходи 1 та 2. Для систем з підвищеними вимогами до точності та швидкодії ефективне застосування підходів 3 та 4 або їх комбінації.

На рис. 3.3, згідно підходу 1, зображено приклад поширення повідомлення з метриками по кластеру, з подальшим скасуванням передачі повідомлення, якщо воно вже було отримано вузлом.

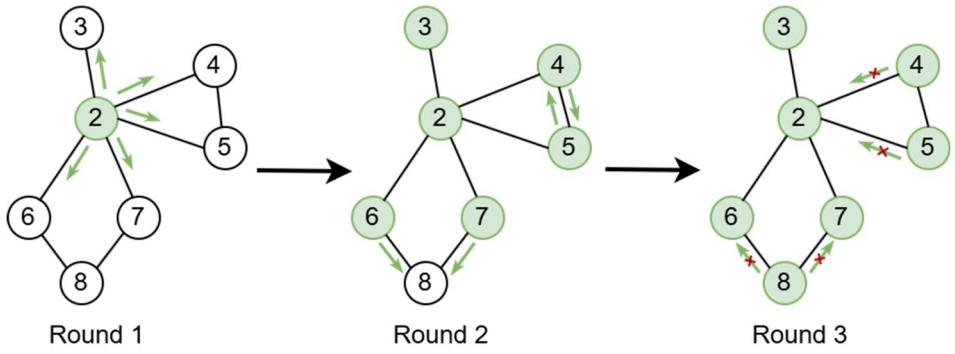


Рисунок 3.3 – Приклад поширення повідомлення з метриками по кластеру

На рис. 3.3 схематично позначено наступні елементи.

Білі кола – повідомлення, яке ще не отримано вузлом; зелені кола – повідомлення вже отримано вузлом; зелені стрілки – напрям розповсюдження повідомлення на поточному етапі; червоні хрестики – повідомлення не поширюється далі, бо воно вже отримано; чорні стрілки – перехід між етапами.

Для ефективного функціонування алгоритму необхідне коректне налаштування параметрів, приклади та рекомендовані діапазони значень яких наведено в табл. 3.3.

Таблиця 3.3 – Основні параметри алгоритму та їх рекомендовані значення

Параметр	Символ	Приклад значення	Призначення
Інтервал пліток	T_{gossip}	1,0–5,0 с	Як часто вузол повідомляє про свої метрики/стан
Розповсюдження пліток	F	2–6 вузлів або «всі сусіди» за раунд	Скільки вузлів комунікують (пліткують) з вузлом
TTL повідомлення	H	6-12 стрибків	Максимальна глибина поширення (якщо відстежується TTL)
Час збіжності	T_{conv}	$Depth_{max} \times T_{gossip}$	Час, за який дані досягають усіх вузлів
Інтервал серцебиття	T_{hb}	0,5–1,0 с	Інтервал серцебиття лідера/замінника
Тайм-аут відмови головного вузла	T_{fail}	$2 \times T_{hb}$ = 1,5–2,0 с	Таймаут перед тим, як головний вузол буде вважатися несправним
Інтервал оновлення оцінок	$T_{refresh}$	10–30 с	Перевизначення кандидатів на основі останніх повідомлень
Час повторного надсилання повідомлень	T_{retry}	100–1000 мс	Затримка перед повторним поширенням повідомлення, якщо немає підтвердження/розповсюдження

Для експериментального моделювання визначено набір умов за типовими сценаріями роботи РТС. Відповідні тестові дані узагальнено в табл. 3.4.

На основі параметрів з табл. 3.4 розраховано приклад моделювання для кластера з 50 вузлів за умови:

1. Інтервал поширення пліток $T_{gossip} = 1{,}0$ с.

2. Кількість вузлів, які отримують повідомлення за один раунд $F = 3$.

Таблиця 3.4 – Вхідні параметри для експериментального моделювання

Параметр	Припущене значення
Розмір кластера	5–100 вузлів
Затримка між вузлами	10–50 мс
Частота втрат повідомлень	Низька (≤ 1%)
Мета програми	Регулярна синхронізація метрик та швидке визначення лідера
Передача	TCP або UDP

Поширення повідомлень в межах алгоритму відбувається в геометричній прогресії:

$$Nodes\ reached\ in\ R\ rounds\ \approx\ F^R$$

Відповідно:

– Після 1 раунду: 3 вузли.

– Після 2 раунду: 9 вузлів.

– Після 3 раунду: 27 вузлів.

– Після 4 раунду: 81 вузол (весь кластер).

Висновок: повна збіжність (поширення інформації) в кластері з 50 вузлів відбувається за 4–6 секунд, що забезпечує швидке узгодження стану та оперативне визначення нового координатора у разі відмови головного вузла.

З огляду на наведені часові характеристики та вимоги до стійкості, проведемо порівняльний аналіз удосконаленого алгоритму Gossip з швидким алгоритмом хулігана, який використовує заступників [41, 87, 113, 147] (табл. 3.5)

Таблиця 3.5 – Порівняння удосконаленого «Пліткар» з Fast Bully Algorithm

Аспект	Швидкий алгоритм хулігана з заступниками	Удосконалений «Пліткар» з асинхронно-узгодженим відбором
Тригер обрання лідера	Явне виявлення збоїв, а потім активні вибори	Без виборів; вузли обчислюють головний вузол незалежно
Відновлення після збою	Попередньо вибраний заступник стає головним	Заступник стає головним. Інші дізнаються або після

	після відмови поточного лідера (шляхом надсилання повідомлення усім іншим)	повідомлення заступника, або з пліток.
Початкові витрати на проведення виборів	Значний сплеск (може викликати шторм)	Ніяких накладних витрат, дорівнює звичайній роботі
Накладні витрати на нормальну роботу	Майже нульові (лише серцебиття) до наступних виборів	Низькі (періодичні плітки про стан/показники), можна налаштовувати
Час збіжності	Швидкий (з використанням ідентифікаторів) або середній (з використанням оцінок).	Повільніший - збігається за кілька раундів пліток, в залежності від глибини кластера. Асинхронно-узгоджений, коли дані розподілені по всій системі
Детермінізм	Детермінований результат (однакові оцінки = однаковий результат)	Можуть короткочасно розходитися, якщо представлення ще не узгоджене
Придатність для нестабільних мереж	Дуже чутливий до втрати повідомлень під час виборів	Стійкий (плітки витримують часткову втрату повідомлень)
Масштабованість (розмір кластера)	Найкраще підходить для малих і середніх кластерів ($\leq$ 50 вузлів)	Легко масштабується до великих кластерів (100-500 вузлів)
Узгодженість даних у виборі головного вузла	Стале представлення після виборів	Може бути суперечливим під час узгодження
Налаштовуваність оцінки	Використовує ідентифікатор або оцінку	Може використовувати будь-яку оцінку
Ризик «розщеплення мозку» (split-brain, 2 головних вузла)	Дуже низький (узгоджене визначення)	Середній, можливий під час затримки/розбіжності повідомлень чуток
Сплеск трафіку при несправності головного вузла	Збір оцінок (тільки з оцінками) та передача результатів виборів	Немає сплеску, інформація поширюється поступово
Загальне використання пропускної здатності	Низьке (з значними сплесками)	Вище (періодичний обмін повідомленнями)

Як видно з табл. 3.5, обидва алгоритми мають свої переваги залежно від умов експлуатації розподіленої телекомунікаційної системи. За результатами порівняння обґрунтовано доцільність використання у межах даного дослідження удосконаленого алгоритму «Пліткар». Його адаптивність, низьке навантаження на канали зв'язку та здатність до стабільної роботи в умовах змінної топології мережі забезпечують кращу відповідність вимогам до масштабованості, відмовостійкості та QoS у системах класу Fog/Edge.

3.3 Метод формування обчислювальних конвеєрів на основі кластерної структури вузлів

Після визначення головних вузлів кластерів, наступним етапом побудови розподіленої телекомунікаційної системи є організація обчислювальних конвеєрів, що реалізують послідовну поетапну обробку даних потоків. Обчислювальний конвеєр в даному дослідженні – це структурований маршрут проходження потоку даних через послідовність кластерів, кожен з яких відповідає за конкретну операцію з обробки [13, 24, 51].

Формування таких маршрутів (конвеєрів) відбувається на основі вже створеної кластерної структури, де кожен кластер призначений для виконання визначеного підзавдання t_{i} в загальній схемі обробки. До формалізованих передумов ефективної організації обчислювальних конвеєрів належать.

1. $C_1, C_2, \dots, C_M$ – множина вже сформованих кластерів за ієрархічною структурою. При чому кожен кластер C_i виконує єдине фіксоване завдання t_i.

2. Завдання виконуються строго у визначеному порядку $t_1 \to t_2 \to t_3 \to \cdots$

3. Кожен кластер має обмежений обсяг обчислювальних ресурсів (CPU, RAM, STG).

4. Інтенсивність кожного потоку t_m – відома або прогнозована і навантаження може тимчасово зростати.

5. Обробка потоку на кожному етапі потребує визначеного обсягу ресурсів (CPU, RAM, STG) вузла, який також може зростати під час пікових навантажень.

6. Сумарна кількість використаних на обробку ресурсів (з урахуванням піків) може перевищувати загальну кількість ресурсів вузла, але цієї ситуації слід уникати, оскільки порушення штрафується відповідною функцією.

7. Передача даних між кластерами відбувається через тунелі з відомими затримками $d_{i,j}$, які впливають на загальний час обробки.

Метою етапу формування обчислювальних конвеєрів у розподіленій телекомунікаційній системі є побудова таких маршрутів конвеєрів між кластерами обчислювальних вузлів, що:

– завдання обробки виконуються у чітко визначеній послідовності;

– усі потоки, що генеруються користувацькими пристроями, проходять повний цикл обробки;

– мінімізується сумарна міжкластерна затримка передачі даних;

– мінімізується сумарний штраф за перевищення доступних обчислювальних ресурсів у кластерах.

Для реалізації мети можуть бути застосовані методи, які відрізняються складністю, принципами роботи, перевагами та недоліками (табл. 3.6).

Таблиця 3.6 – Порівняльний аналіз методів формування конвеєрів

Метод	Переваги	Недоліки
1. Паралельний алгоритм найкоротшого шляху Дейкстри з модифікаціями *(*авторське вдосконалення)*	Моделює кластери та завдання у вигляді спрямованого багатошарового графа. Добре масштабується до тисяч потоків і швидко працює, що дозволяє формувати конвеєри в реальному часі або за запитом. Завдяки штрафам за майже перевантажені вузли зменшує конфлікти без втрати швидкодії.	Потребує перенаправлення для деяких потоків після завершення обробки групи через можливі конфлікти ресурсів. Не є глобально оптимальним і може демонструвати непослідовну поведінку в умовах дуже високого перевантаження.
2. Змішане цілочисельне	Гарантує повністю оптимальне рішення з точки зору ресурсів, якщо таке	Не масштабується – тобто стає недоцільним для більш ніж ~1,000 потоків через

програмування (MIP) [42,55]	існує. Може спільно оптимізувати декілька ланцюжків, затримок та обмежень за один прохід. З модифікаціями пропонує надійний спосіб пошуку компромісів між використанням ресурсів та затримками.	надмірно велику кількість змінних. Повністю відмовляє, якщо не існує прийнятного рішення (немає запасного варіанту). Потребує ретельного налаштування вагових коефіцієнтів та допустимих меж .
3. Жадібний евристичний алгоритм [63,116]	Покроково будує конвеєр, обираючи кластери за затримкою та пропускною здатністю. Простий, прозорий, з мінімальними витратами – придатний для вбудованих систем.	Схильний до неоптимальних рішень через фіксацію ранніх виборів. Не підтримує відкат. Без механізму відновлення легко перевантажується при зростанні трафіку.
4. Генетичний алгоритм / Еволюційний пошук [84,119]	Кодує конвеєр як хромосому. Гнучкий до нелінійних функцій, змінної топології та додаткових критеріїв (надійність, балансування). Працює паралельно або асинхронно, придатний для розподіленої оптимізації.	Немає гарантії оптимальності і може знадобитися багато поколінь, щоб наблизитися до найкращих рішень. Складніше налагоджувати або перевіряти коректність виводу порівняно з моделями, заснованими на правилах.

Порівняльний аналіз в табл. 3.6 доводить, що найбільш оптимальне (точне) рішення дає метод «Змішане цілочисельне програмування (MIP)» [42,55] . Але, оскільки кожне можливе рішення (маршрути між кластерами) є окремою змінною в рамках методу, навіть найпродуктивніші розв’язувачі, такі як Gurobi, не зможуть вирішити дану задачу за короткий час вже при десятках тисяч потоків і парі сотень кластерів, що не дозволяє використовувати даний метод на практиці.

Більш компромісним є запропонований метод «Паралельний алгоритм найкоротшого шляху Дейкстри з модифікаціями», оскільки він містить збалансований алгоритм пошуку маршрутів і, за рахунок попередніх фільтрацій і паралелізму, має значну здатність до масштабування [125,151]. Крім того, метод

може застосовуватись як в однопотоковому, так і в багатопотоковому вигляді, що підвищує його практичну універсальність. Для реалізації методу використано формальні параметри та позначення, наведені в табл. 3.7.

Таблиця 3.7 – Параметри та позначення запропонованого методу

Символ	Значення
C	Множина всіх кластерів $\{C_1, C_2, \dots, C_i\}$
S	Множина усіх потоків вхідних даних $\{s_1, s_2, \dots, s_m\}$
T_m	Упорядковані ланцюжки завдань (етапів): $T_m = [t_1, t_2, \dots, t_n]$
C^t	Множина кластерів, які обробляють завдання типу t
R_{C_i}	Множина ресурсів кластера C_i: $(CPU_{C_i}, RAM_{C_i}, STG_{C_i})$
r	Тип ресурсу: $r \in \{CPU, RAM, STG\}$
SR_{d_m}	Витрати ресурсів (cpu_m, ram_m, stg_m) на обробку потоку даних d_m
$d_{i,j}$	Затримка від кластера i (завдання t) до кластера j (завдання $t+1$)
u_i	Поточна використана кількість ресурсів $u_i = (cpu_i, ram_i, stg_i)$ кластера i
$\omega_{i,j}$	Вага ребра (з'єднання) між двома вузлами (кластерами)
$p_{i,j}$	Штраф, що застосовується до ваги кластера для компенсації перевантажень
M	Запас для розрахунку накладних витрат на ресурси під час сплесків
ϕ_r	Вага штрафу для кожного ресурсу $r \in \{CPU, RAM, STG\}$
B	Розмір групи для паралельної реалізації
Δ_{max}	Максимальний поріг затримки

На основі цих параметрів будується алгоритм формування обчислювальних конвеєрів, результатом роботи якого є маршрути від пристроїв (або кластерів, що їх містять) до останніх у конвеєрах обробки кластерів, виражені у вигляді списку пар ідентифікаторів кластерів.

На основі цих параметрів будується алгоритм формування обчислювальних конвеєрів, результатом якого є маршрути проходження від пристроїв (або кластерів, що їх містять) до кластерів, що виконують фінальні

етапи обробки. Результати подаються у вигляді списку пар ідентифікаторів кластерів, які формують відповідні обчислювальні конвеєри.

З огляду на те, що послідовність завдань фіксована, а кожен кластер виконує лише одне з них, структура взаємозв’язків між кластерами відображається у вигляді спрямованого багатошарового графа.

У такому представленні зручно застосовувати алгоритм Дейкстри для побудови маршрутів, проте при цьому виникають низка специфічних проблем.

1. Алгоритм оперує тільки топологією графа та вагами його вершин (кластерів) і ребер (зв’язків). Дане представлення не враховує використання ресурсів та можливі перевантаження, які можуть виникнути при розподілі множини потоків на один шлях.

2. Навантаження на кластер є кумулятивним, що ускладнює впровадження паралелізму в метод, бо є можливість перевантаження (процеси розрахунку алгоритмів для різних потоків не знають результат виконання один одного під час роботи).

3. Стандартна реалізація алгоритму працює до тих пір, поки не відвідає всі вершини або позначить їх як «недосяжні». При наявності великої кількості кластерів, це може викликати зайві затримки при обчисленні, так як частина кластерів може знаходитися далеко від поточного і розрахунок шляхів щодо них є неефективним.

4. Перевантажені вузли можна також відразу виключити з роботи алгоритму.

5. Послідовне виконання алгоритму для кожного потоку може викликати проблеми з продуктивністю при їх великій кількості.

Для вирішення виявлених проблем проведено вдосконалення алгоритму.

1. Для врахування використаних ресурсів кластера (потоками) і можливих перевантажень, вводиться штраф, який застосовується на ребра, що ведуть до перевантаженого вузла. Штраф та вага ребра (зв’язка) розраховуються за формулами:

$$penalty_i = \sum_r \phi_r \cdot \max\left(0, \frac{u_i^r + M \cdot SR^r - R_i^r}{R_i^r}\right), \quad 3.6$$

$$\omega_{i,j} = d_{i,j} + penalty_j \quad 3.7$$

Правильно підібрані значення ваги ресурсів можуть зробити перевантажені вузли менш пріоритетними при прорахунку шляху.

2. Зменшення порядку вхідних даних алгоритму. Перед етапом розрахунку шляхів для потоку (або набору потоків) виконується формування підграфа за принципами:

2.1 Залишити тільки N кращих (за затримкою, наприклад) вузлів на кожному етапі.

2.2 Залишити тільки вузли з кумулятивною затримкою менше ніж M.

Ці принципи можна використовувати і окремо, в залежності від характеристик системи (числа кластерів).

3. (Опціонально) Суворий підхід до перевантажень. Полягає у формуванні підграфа після фільтрації вузлів за принципом їхньої навантаженості – тобто прибираються як вже перевантажені вузли, так і ті, що будуть перевантажені при додаванні конвеєру. На практиці цей принцип може призвести до роз’єднаного графа, коли всі вузли етапу вже мають потенційне перевантаження. Тому в разі відсутності рішення рекомендується запустити пошук шляху на графі до цього етапу.

Проведені удосконалення дозволяють оптимізувати роботу алгоритму Дейкстри, але загальний час роботи методу може залишитися значним. Така ситуація можлива за наявності великої кількості кластерів у системі, оскільки прорахунок маршрутів для кожного конвеєру виконується послідовно (кожне наступне обчислення вимагає результату попереднього через обмеження за задіяними ресурсами).

Рішенням даної проблеми є введення паралелізму в алгоритм. Пропонується впровадження наступних додаткових удосконалень.

1. Розбиття вхідної множини конвеєрів на піднабори з подальшим паралельним розрахунком маршрутів для кожного з них. Розмір піднабору може бути фіксованим (універсальний варіант) або визначатися адаптивно залежно від характеристик системи. Такий підхід істотно скорочує загальний час реалізації методу, проте створює ризик виникнення «перегонів» між потоками.

Ця проблема виникає, коли два конвеєри з однієї групи обирають спільний маршрут (або його частину), що призводить до перевантаження ресурсів одного чи кількох кластерів.

2. Для усунення «перегонів» між потоками необхідно впровадити механізм вирішення конфліктів.

Цю задачу можна реалізувати різними способами — зокрема, за допомогою жадібних або генетичних алгоритмів, методів цілочисельного програмування тощо. Найбільш збалансованим щодо продуктивності та витрат ресурсів є підхід, заснований на визначенні пріоритетів для потоків у місцях конфлікту:

2.1 У кожному вузлі, де зафіксовано перевантаження ресурсів, обчислюється пріоритет для кожного потоку, що претендує на ресурси цього вузла. Пріоритет може визначатися як лінійна комбінація доступних параметрів: інтенсивність потоку, витрати ресурсів на поточному етапі обробки тощо.

2.2 Потоки сортуються за отриманим пріоритетом, тобто обираються ті N, які вкладаються в наявні ресурсні обмеження.

2.3 Для вузла з конфліктом перераховується штраф, після чого запускається повторний пошук маршруту (алгоритм Дейкстри) для потоків, що залишилися без ресурсу.

2.4. Процедура повторюється до повного розв’язання всіх конфліктів у групі.

Такий підхід є прямолінійним і легко адаптується до різних умов, що робить його придатним до налаштувань і подальших удосконалень.

Для вирішення проблеми масштабованості та уникнення перевантаження ресурсів під час паралельної обробки потоків у розподіленій телекомунікаційній

системі запропоновано алгоритм з розподілом вхідних даних на піднабори та механізмом пріоритетного вирішення конфліктів (рис. 3.4).

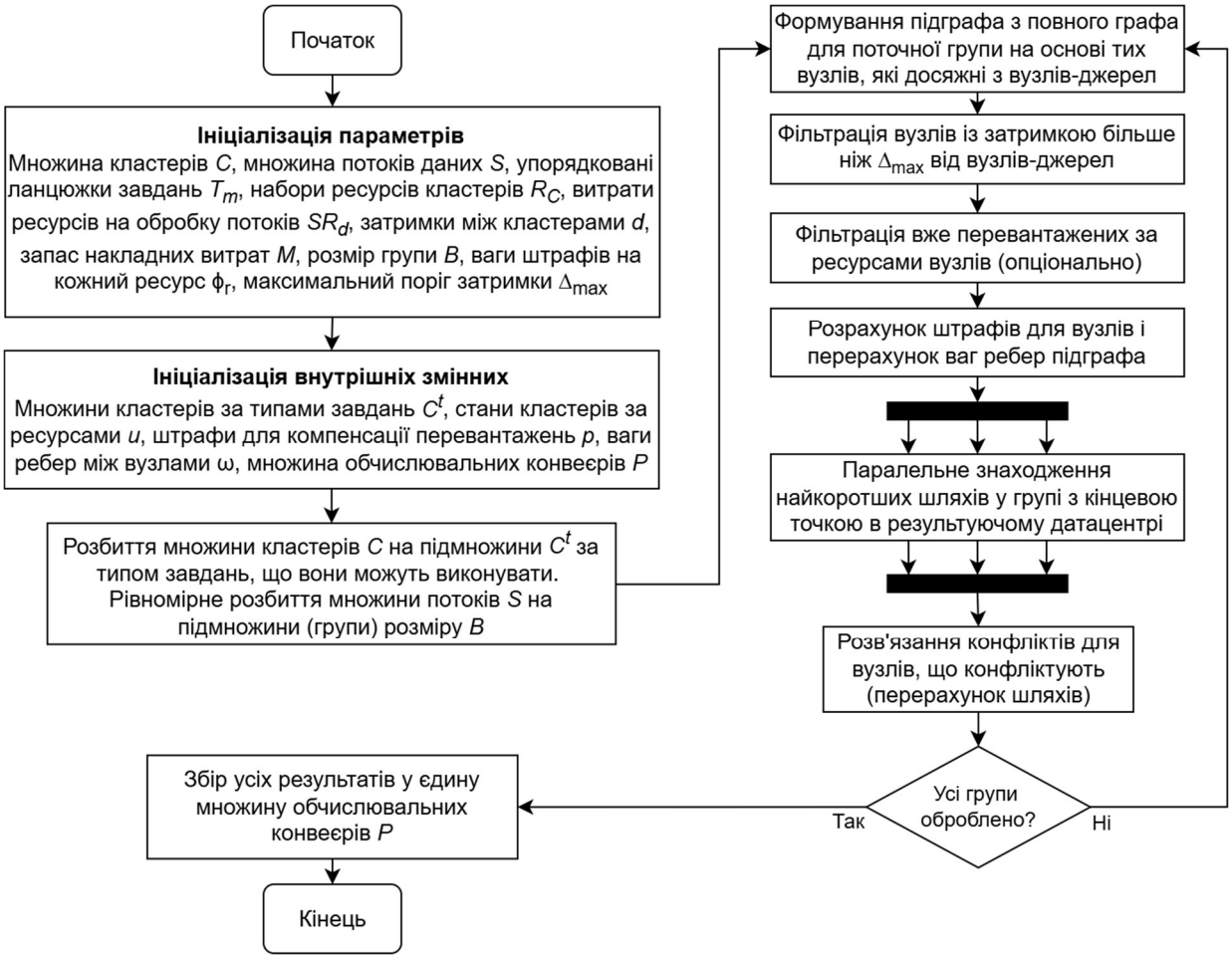


Рисунок 3.4 – Блок схема алгоритму паралельного формування обчислювальних конвеєрів

В результаті проведення всіх модифікацій, алгоритм побудови обчислювальних конвеєрів буде включати наступні кроки.

1. Ініціалізація вхідних даних, параметрів та змінних.

Вхідні дані: множина кластерів C, множина потоків даних S, упорядковані ланцюжки завдань T_m, набори ресурсів кластерів R_C, витрати ресурсів на обробку потоків SR_d, затримки між кластерами d, запас накладних витрат M,

розмір групи B, ваги штрафів на кожний ресурс ϕ_r, максимальний поріг затримки Δ_{max}.

Внутрішні змінні: множини кластерів C^t, що обробляють завдання типу t, стани кластерів за використанням ресурсів u, штрафи для компенсації перевантажень p, ваги ребер між вузлами ω, множина обчислювальних конвеєрів P (результат).

2. Розбиття потоків на групи. Потоки розбиваються на групи однакового розміру (випадковим чином, за локацією або пріоритетом) залежно від їхньої послідовності завдань.

3. Попередня обробка графа. Для кожної групи виконуються такі дії:

– фільтрація вже перевантажених вузлів.

– фільтрація вузлів із затримкою більше ніж Δ_{max}.

– створення підграфа з вузлів, що залишилися.

– розрахунок штрафів для вузлів, що залишилися, і перерахунок ваг усіх ребер з урахуванням штрафів.

4. Знаходження шляхів. Для кожного потоку в групі паралельно виконується алгоритм Дейкстри з кінцевою точкою у вигляді датацентру (місця, куди концентруються всі результати обробки). Якщо шлях не знайдено, конвеєр позначається як «непрохідний».

5. Розв’язання конфліктів. За раніше представленим способом (або альтернативним), перераховуються шляхи для всіх вузлів, що конфліктують.

6. Кроки 3–5 виконуються для всіх груп, поки не будуть оброблені всі потоки.

Послідовна реалізація алгоритму має подібні кроки. Замість розбиття потоків на групи, розрахунок шляхів відбувається послідовно. Відповідно, оскільки штрафи перераховуються після кожного обчислення шляху, вирішення конфліктів також не потрібно виконувати.

Додатково слід зазначити, що IoT-пристрої (камери, датчики) зазвичай згруповані на один мережевий або обчислювальний вузол, хоча кожен із них

генерує власний інформаційний потік. Для зменшення розмірності графа під час побудови конвеєрів приймається:

– один кластер розглядається як окрема вершина графа з ресурсами, що дорівнюють сумі ресурсів усіх його вузлів;

– потоки одного типу (з однаковою послідовністю етапів обробки) агрегуються в один потік з інтенсивністю, яка дорівнює сумі інтенсивностей у групі.

Такий підхід дозволяє знизити обчислювальні витрати при моделюванні. Водночас при невеликій чи/або середній кількості кластерів або потоків агрегацію можна не застосовувати, що підвищує точність побудованих конвеєрів за рахунок більшої деталізації вхідних даних.

Крім того, завдяки використанню методу ієрархічної кластеризації (розділ 2), кластери можуть бути попередньо згруповані за принципом «відстані». Це дає можливість виключити необхідність прорахунку шляхів і затримок між віддаленими кластерами, зосереджуючи увагу лише на реально можливих варіантах маршрутизації.

Узагальнення потоків та попередня кластеризація вузлів дозволяють знизити складність побудови маршрутів і підвищити ефективність методу, що далі підтверджується експериментальною оцінкою у підрозділі 3.4.

3.4 Експериментальна оцінка ефективності методу формування конвеєрів потокової обробки

Для перевірки ефективності запропонованого методу було проведено експериментальне моделювання з використанням графа, представленого у розділі 2, який було удосконалено відповідно до вимог етапу формування конвеєрів потокової обробки даних. Отримана модель є наближеною до реальних вимог РТС саме на даному етапі дослідження (рис. 3.5).

На графі представленому на рис. 3.5 вершини відповідають кластерам вузлів, які об’єднують гетерогенні обчислювальні ресурси. До кластерів першого

етапу приєднуються вхідні вузли–потоки, що моделюють джерела даних. Кожен кластер містить набір обчислювальних вузлів, які виконують завдання певного типу. Всередині одного етапу вершини утворюється повний граф, що відображає можливість гнучкої маршрутизації потоків між кластерами. Водночас між етапами існують міжкластерні з’єднання, кількість яких обмежена певною часткою від максимальної, що моделює реальні умови обмеженої пропускної здатності.

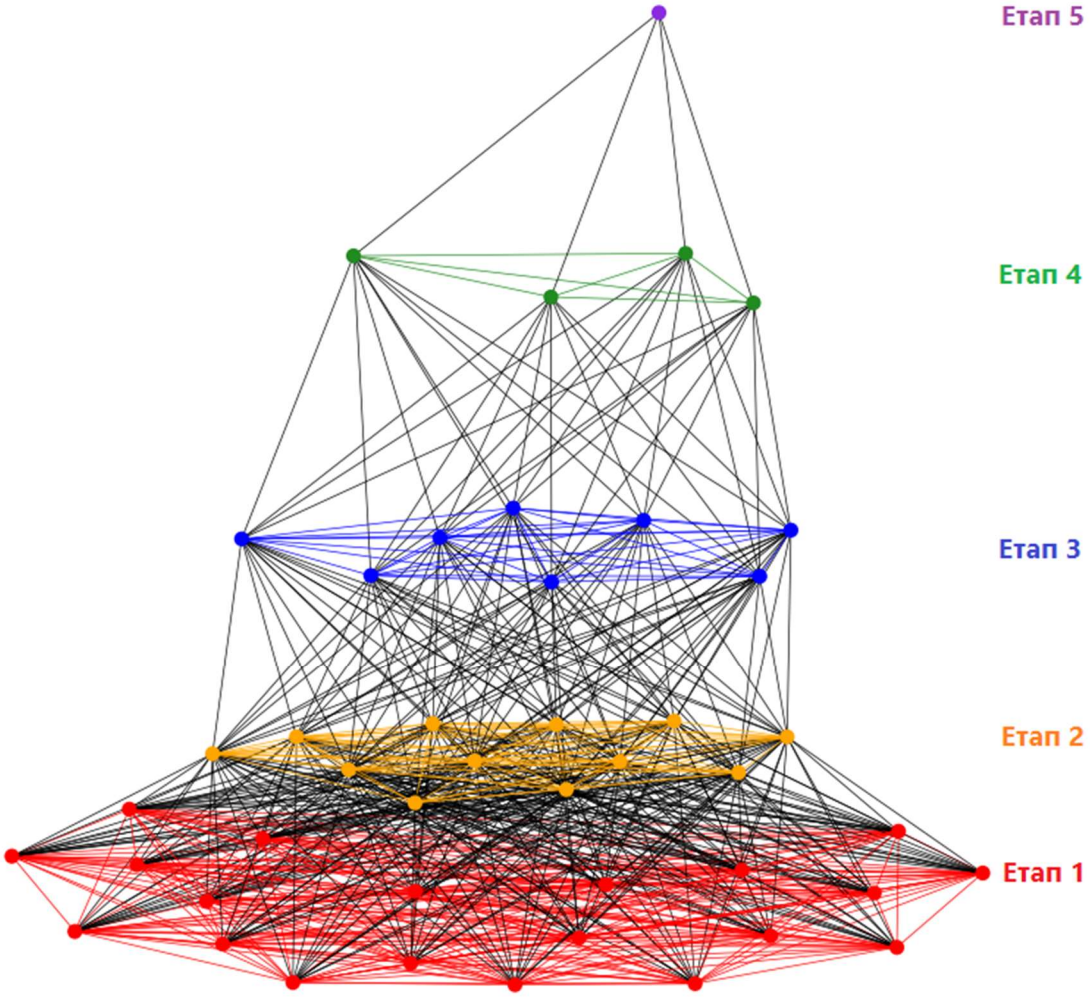


Рисунок 3.5 – Графове представлення системи

Різними кольорами на рис. 3.5 виділено кластери різних етапів конвеєра потокової обробки: червоний – етап 1 (вхідні потоки даних), помаранчевий – етап 2 (попередня обробка), синій – етап 3 (виділення та трансформація ознак), зелений – етап 4 (класифікація та агрегація результатів), фіолетовий – етап 5 (формування вихідних даних).

Для кількісної оцінки ефективності методу формування конвеєрів було проведено експериментальне моделювання на змодельованому графі (рис. 3.5). Система містить 28 кластерів, організованих за рівнями у співвідношенні 28 (10 -> 8 -> 6 -> 3 -> 1), з яких формується конвеєрна структура обробки даних. Загальна кількість потоків в експерименті становила 180. У кожному кластері знаходяться обчислювальні вузли різної потужності, кількість яких зменшується від 8 на першому рівні до 1 на вихідному рівні. Така конфігурація дозволяє оцінити поведінку алгоритмів при різній щільності та глибині конвеєрів.

У табл. 3.8 наведено результати порівняння роботи різних алгоритмів.

Таблиця 3.8 – Експериментальні результати роботи алгоритмів

Алгоритм	Жадібний	Дейкстра	Модифікований (авторський)
Затримка (мін., мс)	218.0	218.0	218.0
Затримка (макс., мс)	308.0	274.0	318.0
Затримка (середне, мс)	275.377778	243.944444	260.961111
Затримка (ст.відх., мс)	19.940458	11.242240	18.669835
Навантаження (мін.)	0.0	0.0	0.0
Навантаження % (макс.)	378.666667	594.126500	198.042167
Навантаження % (середне)	97.363195	97.363195	97.580444
Навантаження % (ст.відх.)	96.906852	130.722072	60.371745
Перевантаження % (мін.)	0.0	0.0	0.0
Перевантаження % (макс.)	278.666667	494.126500	98.042167
Перевантаження % (середне)	32.266744	43.116399	22.889553
Перевантаження % (ст.відх.)	72.067678	105.097815	28.058323

За результатами експерименту можна зробити наступні висновки.

1. Модифікований алгоритм усунув значні стрибки навантаження вузлів. Максимальне навантаження зменшилося на 47,7% у порівнянні з жадібним алгоритмом та на 66,7% у порівнянні з алгоритмом Дейкстри. На відміну від класичного алгоритму Дейкстри, який орієнтується лише на оптимальні маршрути, що приводить до перевантаження вузлів на останніх етапах, запропонований удосконалений алгоритм усуває цей недолік.

2. Стандартне відхилення навантаження та перевантаження зменшилося на 37,7% відносно жадібного алгоритму та 53,8% відносно алгоритму Дейкстри, що свідчить про більш рівномірне завантаження кластерів.

3. Середнє значення перевантаження вузлів також знизилося на 9,4% у порівнянні з жадібним алгоритмом та на 20,2% у порівнянні з алгоритмом Дейкстри, що підтверджує підвищення стабільності роботи системи.

4. Показники затримки залишаються на прийнятному рівні. Це свідчить про те, що запропонований алгоритм забезпечує краще балансування навантаження без істотного зростання часу обробки.

Візуалізація результатів експерименту представлена на рис. 3.6 – 3.10.

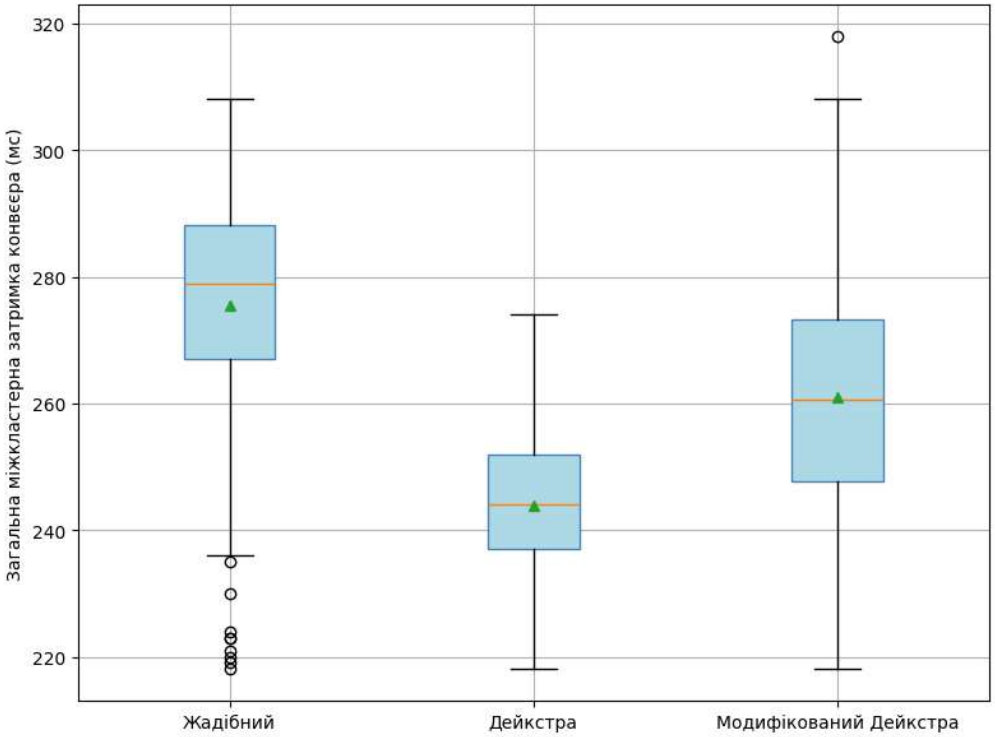
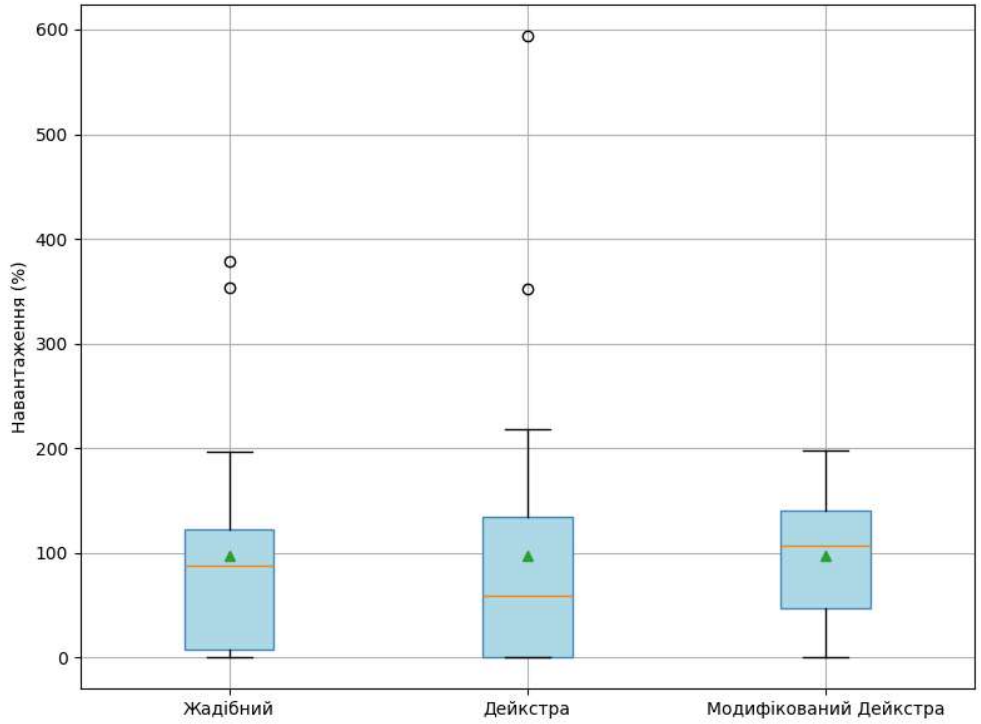


Рисунок 3.6 – Розподіл міжкластерних затримок за алгоритмами

Як видно з рис. 3.6, модифікований алгоритм забезпечує стабільніші значення міжкластерних затримок та менший розкид.



Рисунки 3.7 – Розподіл навантаження за алгоритмами

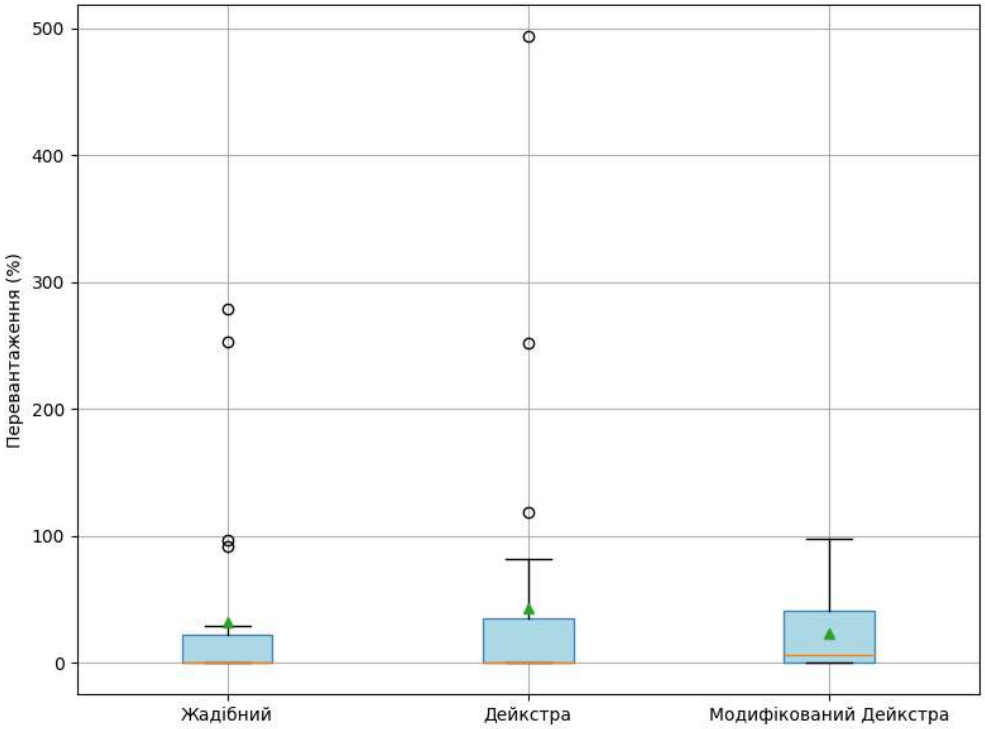


Рисунок 3.8 – Розподіл перевантаження за алгоритмами

На рис. 3.9 та 3.10 представлено розподіл навантаження та перевантаження між окремими вузлами системи для різних алгоритмів.

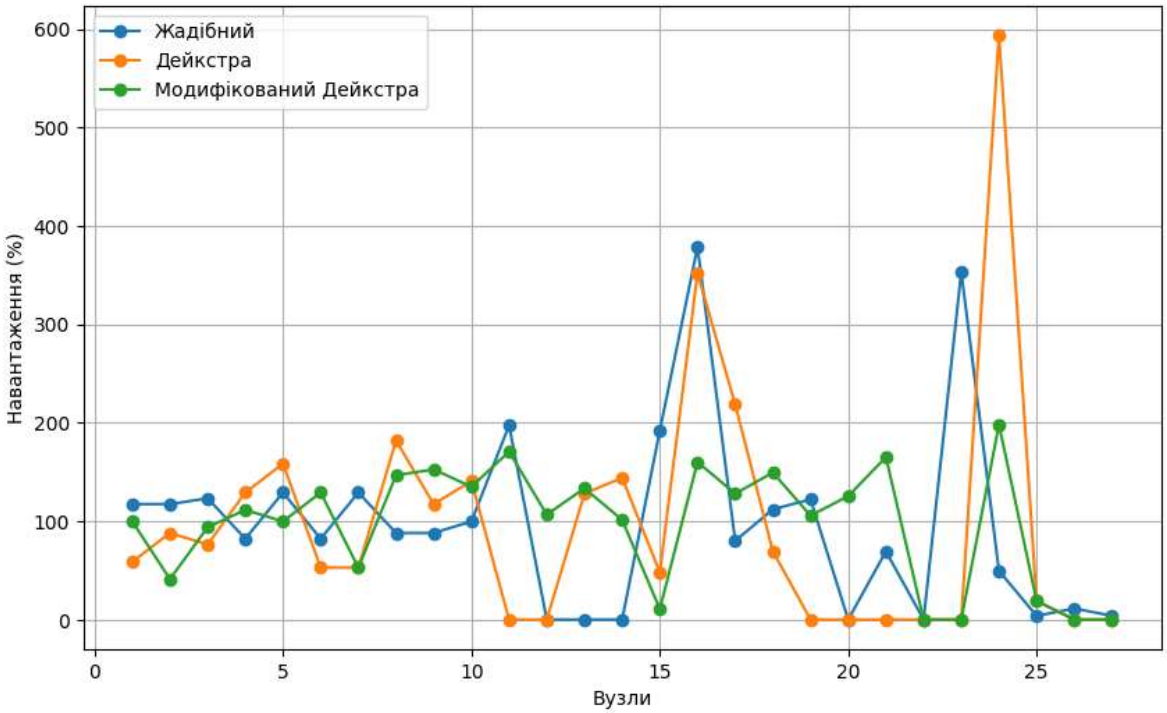


Рисунок 3.9 – Розподіл навантаження за вузлами для різних алгоритмів

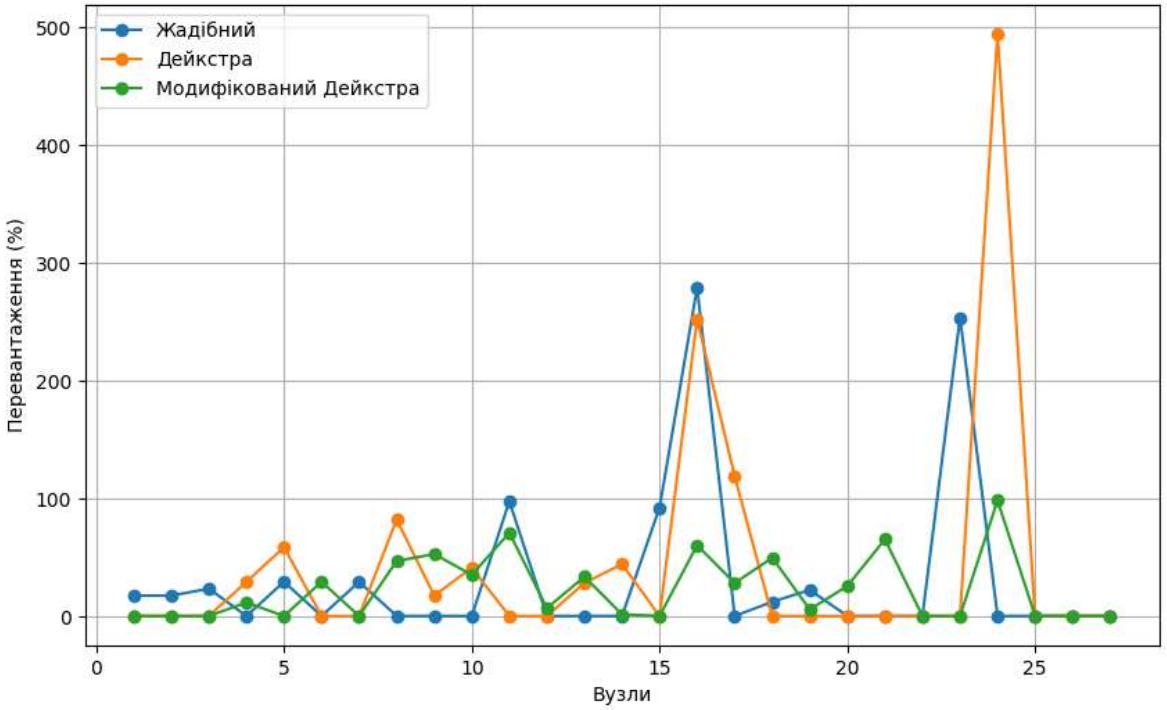


Рисунок 3.10 – Розподіл перевантаження за вузлами

Як видно з рис. 3.9 та 3.10, модифікований алгоритм забезпечує більш рівномірний розподіл навантаження та перевантаження між вузлами. На відміну від класичного алгоритму Дейкстри, де спостерігаються різкі пікові значення навантаженості окремих вузлів, удосконалений підхід знижує амплітуду коливань та мінімізує ризик надмірного перевантаження на критичних етапах обчислень. Це підтверджує його здатність стабілізувати роботу системи за рахунок балансування ресурсів.

Таким чином, для роботи в РТС модифікований алгоритм є доцільним вибором, оскільки поєднує прийнятні затримки з більш рівномірним використанням ресурсів та зниженням ризику перевантажень, що підвищує загальну надійність і ефективність потокової обробки даних.

Висновки за розділом 3

1. Досліджено проблему вибору головного вузла та формування обчислювальних конвеєрів у РТС з кластерною архітектурою. На основі порівняльного аналізу класичних та ймовірнісних алгоритмів лідерства, а також консенсус-протоколів, обґрунтовано недоліки традиційних підходів щодо високих накладних витрат, нестабільності при втраті зв’язку та обмеженої масштабованості. Проведено порівняльну оцінку таких алгоритмів, як Bully, Raft, VSR та Fast Bully, що дозволило виокремити їх сильні та слабкі сторони з урахуванням вимог до кластеризованих РТС.

2. Запропоновано метод вибору головного вузла, який базується на модифікованому алгоритмі «Пліткар» (Gossip) з удосконаленими показниками під цілі дослідження. Розроблена оціночна функція дозволяє ранжувати вузли за трьома критеріями: середньою затримкою, ресурсною потужністю та ідентифікатором. Завдяки цьому забезпечено стабільне керування без запуску явної процедури виборів. Метод підтримує автоматичне призначення заступника координатора, що мінімізує час відновлення після збоїв, знижує навантаження на мережу та підвищує стійкість до часткових втрат повідомлень.

3. Математично обґрунтовано ефективність запропонованого підходу шляхом аналізу параметрів швидкості збіжності, трафіку, TTL, частоти серцебиття та тайм-аутів. Проведено симуляційне моделювання, яке показало, що обмін метриками в межах кластерів дозволяє досягати повної синхронізації за 4–6 секунд. У підсумку встановлено, що вдосконалений алгоритм Gossip демонструє кращу масштабованість і стійкість порівняно з Fast Bully у динамічних умовах, що підтверджує доцільність його впровадження для самокерованих кластерів РТС з потоковою обробкою даних.

5. Запропоновано метод формування обчислювальних конвеєрів у кластеризованому середовищі РТС, що забезпечує послідовну маршрутизацію потоків між кластерами з урахуванням продуктивності вузлів, ресурсних обмежень та пропускної здатності. За допомогою методу можна уникати перевантаження, зменшувати затримки та ефективно розподіляти завдання між етапами обробки.

6. Розроблено алгоритм реалізації на основі спрямованого багатошарового графа з урахуванням міжкластерних затримок, штрафів за перевантаження та паралельного прорахунку маршрутів. Запропоновано механізм фільтрації непрохідних вузлів і механізм пріоритетного вирішення конфліктів, що забезпечує масштабованість до тисяч потоків і стабільність роботи при пікових навантаженнях. Експериментально доведено ефективність запропонованого методу до умов РТС.

7. Експериментальна оцінка ефективності методу формування конвеєрів потокової обробки довела, що удосконалений алгоритм забезпечує скорочення середніх затримок на 5–7%, зменшення кількості перевантажень до 20%, а також підвищення рівномірності завантаження ресурсів вузлів на 38–54%.

РОЗДІЛ 4
МЕТОД ІНТЕЛЕКТУАЛЬНОГО РОЗПОДІЛУ ОБЧИСЛЕНЬ У ПОТОКОВИХ ДАНИХ НА ОСНОВІ ГЕНЕТИЧНОГО АЛГОРИТМУ З ПРОГНОЗУВАННЯМ НАВАНТАЖЕННЯ

4.1 Обґрунтування методу багатокритеріального розподілу інформаційних потоків у РТС на основі еволюційного підходу

У РТС ефективна обробка інформаційних потоків ускладнюється обмеженістю ресурсів, нерівномірним завантаженням кластерів, сплесками трафіку та потребою в оперативному делегуванні навантаження. Такі системи характеризуються динамікою та наявністю множини конфліктних критеріїв, що потребує пошуку компромісних рішень у реальному часі.

Для розв’язання поставленої науково-практичної задачі, з врахуванням сучасних концептуальних підходів до багатоцільової еволюційної оптимізації [33, 48, 49, 60, 66, 84-85, 114, 122, 119, 127, 139, 158], запропоновано метод багатокритеріального прийняття рішень для розподілу інформаційних потоків, який передбачає:

– представлення рішень у вигляді хромосом з призначенням потоків до вузлів;

– оцінювання рішень за функцією пристосованості, яка враховує кількість делегацій, баланс навантаження та ризик перевантажень;

– адаптивне реагування на зміни у трафіку шляхом динамічного оновлення параметрів.

Метод є універсальним і може бути реалізований на основі будь-якого багатокритеріального генетичного алгоритму. У межах даного дослідження обрано NSGA-III, оскільки він забезпечує ефективне покриття Парето-простору при великій кількості критеріїв і дозволяє легко інтегрувати адаптивні механізми.

Запропонований метод формує стійке рішення в умовах нестабільного навантаження, забезпечує гнучкий контроль за ресурсами системи та може бути використаний для динамічного управління потоками в реальному часі.

Аналіз сучасних досліджень у галузі багатокритеріальної еволюційної оптимізації та застосування генетичних алгоритмів у розподілених обчислювальних середовищах доводить, що проблема ефективного управління інформаційними потоками за умов динамічного навантаження залишається актуальною. У роботах [33, 84, 119, 131, 153] розглянуто використання генетичних алгоритмів для задач маршрутизації, покриття та розміщення сервісів у ресурсно обмежених середовищах, однак без врахування багатокритеріальності чи адаптивності до змін параметрів трафіку.

Наукові роботи [49, 60, 66, 104, 146, 150, 127] зосереджені на розвитку багатоцільових еволюційних методів, зокрема NSGA-III, де досліджено механізми побудови референсних точок, інкрементального розширення векторів ваг, сортування рішень та генерації добре розподілених Парето-фронтів. Проте, як засвідчують результати [99, 114, 122], більшість з існуючих підходів потребують подальшої адаптації до умов нестабільного навантаження, сплесків ресурсоємності та потреби в гнучкому делегуванні задач у реальному часі.

В дослідженнях [158, 139] запропоновано моделі сервісного розміщення та оптимізації у хмарно-телекомунікаційних архітектурах, але вони не враховують специфіку потокової обробки даних з високою варіативністю параметрів, що характерна для телекомунікаційних кластерів з неоднорідними ресурсами.

Отже, узагальнені результати сучасних досліджень формують теоретичну основу для розробки адаптивного методу розподілу інформаційних потоків у розподілених телекомунікаційних мережах, який поєднує переваги багатокритеріальної оптимізації, механізмів делегування та здатність до реакції на раптові зміни в навантаженні.

Модель системи, у якій реалізується запропонований метод, описує розподілене середовище обробки інформаційних потоків [33,119], і ґрунтується

на архітектурі, спеціально розробленій у межах даного дослідження, що складається з кластерів обчислювальних вузлів, що можуть виконувати задачу одного типу. Кожен кластер містить гетерогенні обчислювальні ресурси та має координатора, що виконує централізоване управління призначенням задач на вузли кластера. У разі нестачі ресурсів передбачено механізм делегування потоків до сусідніх кластерів або глобального супервізора. Інформаційні потоки надходять до системи з різною інтенсивністю, можуть відрізнятись за типом, ресурсоємністю та мати властивість раптових сплесків (збільшення ресурсоємності на деякий проміжок часу).

Узагальнена архітектура запропонованої системи наведена на рис.4.1.

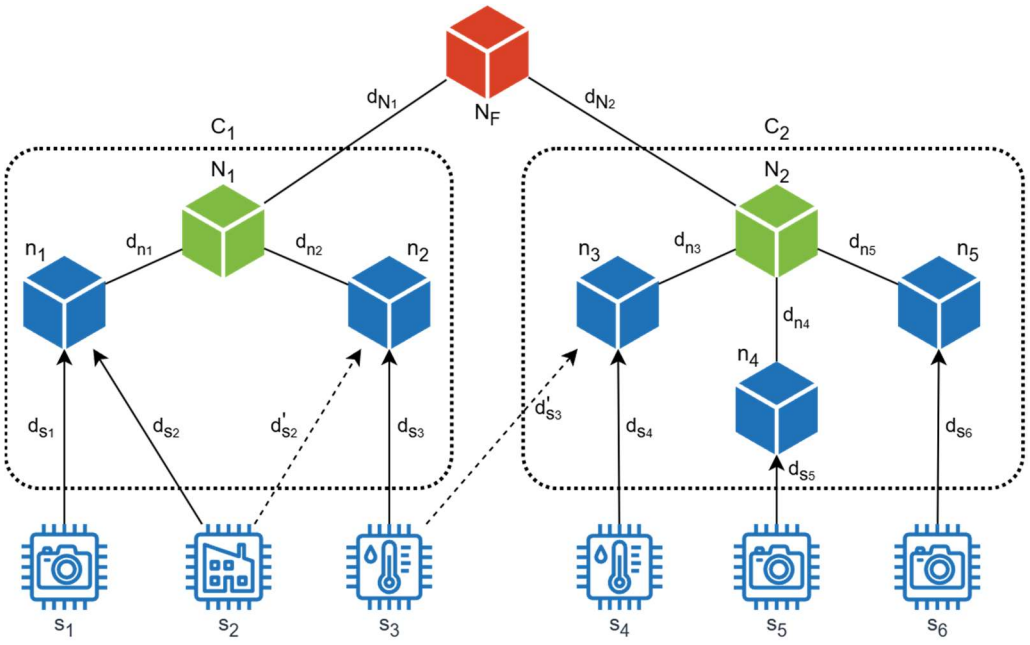


Рисунок 4.1 – Архітектура системи з координаторами кластерів та механізмами делегування

З рис. 4.1 видно, що архітектура запропонованої системи представлена кластерами (C_1, C_2), кожен з яких складається з обчислювальних вузлів ($n_1 - n_S$), що виконують задачі одного типу, але відрізняються за обчислювальними можливостями (CPU, RAM, Storage). Кожен кластер має координатора (N_1, N_2), що здійснює централізоване керування розподілом вхідних інформаційних

потоків ($d_{S1} - d_{S6}$). У разі нестачі ресурсів кластер може делегувати інформаційний потік іншому вузлу кластера ($d_{S2} \rightarrow d'_{S2}$), іншому кластеру ($d_{S3} \rightarrow d'_{S3}$) або глобальному супервізору (N_F). Така архітектура формує адаптивну розподілену систему, здатну гнучко реагувати на зміну навантаження, зокрема на сплески ресурсоємності потоків.

Синім кольором на схемі позначено обчислювальні вузли, які фізично виконують обробку потоків, зеленим кольором – координатори кластерів, які приймають рішення про розподіл потоків, а червоним кольором позначено глобального супервізора (N_F), до якого можуть передаватись потоки при неможливості обробки у кластері.

У нижній частині рисунка розміщено іконки, які умовно позначають типові джерела інформаційних потоків ($s_1 - s_6$). Кожна з них ілюструє приклад пристрою або тип задачі, що генерує відповідний потік:

– іконка будинку – це потоки з пристроїв «розумного дому» або з екосистеми промислового підприємства (industrial IoT);

– іконка камери – відеопотоки або окремі задачі з обробки зображень (image processing);

– іконка термометра – потоки з сенсорів температури або подібних пристроїв моніторингу навколишнього середовища.

Запропонована архітектура формує адаптивну розподілену систему, здатну гнучко реагувати на зміни навантаження, зокрема на раптові сплески ресурсоємності, що моделюються як статистичні події з імовірністю, що зростає з часом.

З врахуванням задач даного дослідження, а також, логіки розподілу інформаційних потоків між вузлами в описаному середовищі, модель може бути представлена як різновид багатокритеріальної задачі розкладу [33,84,119]. Вона включає елементи задачі розміщення (англ. placement problem), але розширена специфічними обмеженнями, що відображають особливості та вимоги даного дослідження:

– обробка задач лише фіксованого типу в межах кожного кластера;

– обмеженість обчислювальних ресурсів (CPU, RAM, Storage);

– ризик перевантаження при раптових сплесках навантаження;

– необхідність підтримки гнучкого делегування при дефіциті ресурсів.

Основні характеристики системи, які враховуються під час моделювання та вирішення задачі оптимального розподілу, наведено в табл. 4.1.

Таблиця 4.1 – Основні характеристики системи

Елемент	Описання
Типи задач	Є кілька фіксованих типів задач (наприклад, 1–4); кожен кластер обробляє задачі лише одного типу
Інформаційні потоки	Кожен потік має: тип задачі, ресурсоємність (CPU, RAM, Storage), можливість короткочасних сплесків ресурсоємності
Вузли обробки	У межах кластера; гетерогенні за ресурсами; обробляють лише потоки одного типу задач
Кластери	Об’єднують вузли одного типу; кожен має координатора, що виконує централізоване планування
Координатори	Виконують призначення потоків на локальні вузли на основі доступних ресурсів; не мають інформації про детальний стан інших кластерів
Делегування	Можливе лише у разі перевантаження: до сусідніх кластерів або до глобального супервізора
Сплески навантаження	Виникають з імовірністю, що зростає за експоненційним законом; тривають протягом інтервалу τ та призводять до тимчасового зростання вимог до ресурсів
Обмеження потоку	Призначення кожного потоку: лише один вузол або делегування; контроль ресурсного перевантаження; мінімізація делегування
Критерії оптимізації	Використання ресурсів, кількість делегацій, ризик перевантаження під час сплесків, збалансованість навантаження

В табл. 4.2 представлено авторську формалізацію основних параметрів, структурних компонентів та змінних, які використовуються у запропонованому методі багатокритеріального розподілу інформаційних потоків. Вказані параметри узгоджуються з базовими підходами [33, 52, 84, 119].

Таблиця 4.2 – Параметри та позначення показників

Позна-чення	Значення
C	Набір всіх кластерів $\{C_1, C_2, \dots, C_i\}$, які обробляють задачі типу T_k
n_{C_i}	Вузол у кластері C_i, що обробляє задачі типу T_k
$S_{C_i}^n$	Набір всіх обчислювальних вузлів всередині кластеру C_i: $\{n_1, n_2, \dots, n_j\}$
N_{C_i}	Головний вузол кластера (координатор), що розподіляє навантаження
N_F	Вузол, що виконує роль глобального супервізора системи
D	Набір всіх вхідних інформаційних потоків $\{d_1, d_2, \dots, d_m\}$
NB_C	Набір сусідніх кластерів: $NB_C = \{C_1', \dots, C_k'\}$
EX_C	Зовнішні ресурси кластеру C: $EX_C = \{NB_C, N_F\}$
$R_{n_{C_i}}$	Сукупність ресурсів вузла у кластері C_i: $\left(CPU_{n_{C_i}}, RAM_{n_{C_i}}, STG_{n_{C_i}}\right)$
R_{d_m}	Витрати ресурсів (cpu_m, ram_m, stg_m) на обробку потоку даних d_m
τ	Тривалість можливого сплеску навантаження на потоці
λ	Параметр експоненційного розподілу для оцінки ймовірності сплеску
σ	Мультиплікатор навантаження під час сплеску. Показує скільки додаткового навантаження споживає потік під час сплеску $\left(\sigma_{cpu}, \sigma_{ram}, \sigma_{stg}\right)$
$x_{m,n_{C_i}}$	Змінна прийняття рішення. Має значення 1 при присвоєнні вузлу n_{C_i} всередині кластера – потоку d_m
y_{m,EX_C}	Змінна прийняття рішення. Має значення 1 при присвоєнні ресурсу ззовні EX_C (іншому кластеру або супервізору) – потоку d_m
X_m	Узагальнена змінна прийняття рішення – присвоєння потоку d_m ресурсу (вузлу, іншому кластеру або супервізору)
F	Вектор цільової функції $F = [f_1, f_2, f_3, f_4]$, що включає критерії оптимізації: мінімізацію використання ресурсів f_1, кількість делегацій f_2, перевантаження під час сплесків f_3 та дисбаланс навантаження f_4
w_{f_a}	Вагові коефіцієнти для кожного критерію у функції F

Для реалізації запропонованого методу багатокритеріального розподілу інформаційних потоків при оцінюванні кожного варіанта призначення

використовується система цільових функцій f_1, f_2, f_3, f_4. В даному дослідженні, кожен можливий розподіл (потенційне рішення) оцінюється за допомогою векторної функції мети $F = f_1, f_2, f_3, f_4$, яка враховує як поточний стан обчислювального середовища, так і потенційні ризики його перевантаження у майбутньому.

Структура запропонованих функцій базується на принципах еволюційної багатокритеріальної оптимізації [33, 49, 93], з адаптацією до специфіки телекомунікаційних систем [84, 105]. Кожен компонент вектора *F* має власну вагу w_{f_a}, що дозволяє налаштувати вплив кожного критерію на результат вибору.

Розроблена система критеріїв у межах запропонованого методу дозволяє досягти наступних наукових цілей.

1. Оцінка ефективності використання ресурсів. Цей аспект описує критерій f_1 – мінімізація загального використання локальних ресурсів (процесор, оперативна пам’ять, сховище). Розраховується за формулою (позначення показників див. табл. 4. 2):

$$\min f_1 = \sum_{n \in S_C^n} \left(\frac{1}{CPU_n} \sum_{d \in D} x_{d,n} \cdot cpu_d + \frac{1}{RAM_n} \sum_{d \in D} x_{d,n} \cdot ram \right. \\ \left. + \frac{1}{STG_n} \sum_{d \in D} x_{d,n} \cdot stg_d \right) \quad (4.1)$$

2. Оцінка потреби в делегуванні потоків. Відповідає критерію f_2 – це мінімізація кількості потоків, що були делеговані зовнішнім ресурсам (сусіднім кластерам або супервізору):

$$\min f_2 = \sum_{i=1}^{m} y_{i,EX} \quad (4.2)$$

3. Передбачення ризику перевантаження ресурсів у разі сплесків. Критерій f_3 – мінімізація очікуваного перевантаження ресурсів під час сплеску навантаження τ, з урахуванням імовірності його виникнення:

$$\min f_3 = \sum_{n \in S_C^n} (CPU_n^{over} + RAM_n^{over} + STO_n^{over}) \quad 4.3$$

Очікувана перевантаженість ресурсів для кожного вузла n розраховується за формулами:

$$CPU_n^{over} = max\left(0, \sum_{d \in D} \left(P_d^{surge} \cdot x_{d,n} \cdot \sigma_{cpu} \cdot cpu_i\right) - CPU_n\right) \quad 4.4$$

$$RAM_n^{over} = max\left(0, \sum_{d \in D} \left(P_d^{surge} \cdot x_{d,n} \cdot \sigma_{ram} \cdot ram_i\right) - RAM_n\right) \quad 4.5$$

$$STO_n^{over} = max\left(0, \sum_{d \in D} \left(P_d^{surge} \cdot x_{d,n} \cdot \sigma_{stg} \cdot sto_i\right) - STO_n\right) \quad 4.6$$

Де імовірність сплеску навантаження розраховується як:

$$P^{surge} = 1 - e^{-\lambda\tau} \quad 4.7$$

4. Забезпечення збалансованості розподілу потоків між вузлами кластера. Оцінюється критерієм f_4 – це мінімізація дисперсії (Var) навантаження на вузол (для процесора, оперативної пам’яті, сховища), що забезпечує збалансований розподіл потоків:

$$\min f_4 = Var^{CPU} + Var^{RAM} + Var^{STG} \quad 4.8$$

$$\begin{cases} Var^{CPU} = Var(CPU_1^{load}, CPU_2^{load}, \dots, CPU_j^{load}) \\ Var^{RAM} = Var(RAM_1^{load}, RAM_2^{load}, \dots, RAM_j^{load}) \\ Var^{STG} = Var(STG_1^{load}, STG_2^{load}, \dots, STG_j^{load}) \end{cases} \quad 4.9$$

$$\begin{cases} CPU_n^{load} = \dfrac{1}{CPU_n} \sum\limits_{d \in D} x_{d,n} \cdot cpu_d \\ RAM_n^{load} = \dfrac{1}{RAM_n} \sum\limits_{d \in D} x_{d,n} \cdot ram_d \\ STG_n^{load} = \dfrac{1}{STG_n} \sum\limits_{d \in D} x_{d,n} \cdot stg_d \end{cases} \quad 4.10$$

Описані критерії оптимізації застосовуються за наявності системи обмежень.

1. Обмеження однозначного призначення (кожен потік повинен бути або оброблений локально, або делегований):

$$X_d = \sum_{n \in S_C^n} x_{d,n} + y_{d,EX} = 1, \forall \mathrm{d} \in D \quad 4.11$$

2. Обмеження обсягу ресурсів вузлів (для кожного вузла не допускається перевищення доступних ресурсів):

$$\sum_{d \in D} x_{d,n} \cdot cpu_d \leq CPU_n, \forall n \in S_C^n, \quad 4.12$$

$$\sum_{d \in D} x_{d,n} \cdot ram_d \leq RAM_n, \forall n \in S_C^n, \quad 4.13$$

$$\sum_{d \in D} x_{d,n} \cdot sto_d \leq STO_n, \forall n \in S_C^n, \quad 4.14$$

3. Сумісність типів задач (кожен вузол може обробляти лише задачі певного типу — закладено у структурі кластерів, і забезпечується на етапі ініціалізації хромосоми).

З урахуванням зазначених обмежень кожне потенційне рішення оцінюється через адитивну функцію пристосованості:

$$F = w_1 \cdot f_1 + w_2 \cdot f_2 + w_3 \cdot f_3 + w_4 \cdot f_4, \qquad 4.15$$

Для реалізації запропонованого методу використовується генетичне представлення потенційних рішень у вигляді хромосоми, що відповідає загальноприйнятим підходам до еволюційного моделювання задач розміщення та маршрутизації у розподілених середовищах [33, 84, 93]. Кожна хромосома кодує один з можливих варіантів призначення інформаційних потоків до доступних ресурсів, враховуючи як локальні вузли, так і можливість делегування до зовнішніх об’єктів.

Індикація приналежності потоку до вузла в хромосомі може бути виконана двома способами: бінарним і цілочисельним [52, 84, 119].

1. Бінарне кодування передбачає створення набору бітів для кожного потоку. Розмір такого набору дорівнює кількості всіх можливих напрямків розміщення (всі вузли кластера, супервізор, сусідні кластери).

2. Цілочисельне кодування використовує лише один масив цілих чисел, де абсолютне значення числа є ідентифікатором ресурсу, а знак – його типом:

– якщо число > 0 – воно є ідентифікатором локального вузла у межах кластера;

– якщо число дорівнює 0 – потік необхідно розподілити вузлу-супервізору;

– якщо число менше за 0 – воно є ідентифікатором кластера, якому необхідно розподілити цей потік.

За потреби, якщо використовуються складні ідентифікатори кластерів і вузлів, цей підхід можна поєднувати з маппінг-таблицею, яка дозволяє відновити точну відповідність між ідентифікатором та ресурсом.

Проведемо оцінку обсягу пам’яті, необхідної для представлення хромосоми у кожному з методів кодування із врахуванням параметрів, що визначені для задач цього дослідження (табл. 4.3).

Таблиця 4.3 – Оцінка пам’яті для представлення хромосоми

Параметр	Значення
Кількість потоків S	1000
Кількість внутрішніх вузлів N	16
Кількість сусідніх кластерів N_b	7
Наявність супервізору N_F	1
Розмір цілочисельного ідентифікатора кластера	8 біт (1 байт)
Розмір у пам’яті (бінарне кодування) $Size_{bin}$	1000 × (16+7+1) = 24000 байт
Розмір у пам’яті (цілочисельне кодування) $Size_{int}$	1000 × 8 = 8000 байт

Порівняння підходів, наведене в табл. 4.4, дозволяє зробити висновок, що для задач даного дослідження ефективнішим є цілочисельний підхід до кодування хромосоми. Його застосування забезпечує значне зменшення обсягу пам’яті, вищу масштабованість та дозволяє уникнути обмежень щодо максимальної кількості вузлів, оскільки немає потреби додатково контролювати належність потоку до внутрішніх чи зовнішніх ресурсів.

Таблиця 4.4 – Оцінка ефективності бінарного та цілочисельного кодування

Характеристика	Бінарне кодування	Цілочисельне кодування
Обсяг пам’яті	Високий	Низький
Складність декодування	Простий	Потребує маппінгу
Гнучкість масштабування	Обмежена кількістю вузлів	Вища – не залежить від фіксованої N
Швидкість пошуку	Швидке розпізнавання бітів	Можливо повільніше, залежить від реалізації
Універсальність	Не підходить для гібридних структур	Ефективно працює з гібридними системами

Структуру хромосоми, що використовує цілочисельне кодування, представлено на рис. 4.2.

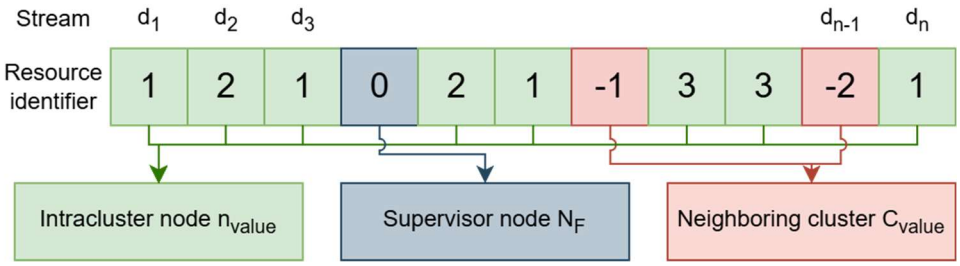


Рисунок 4.2 – Представлення хромосоми з цілочисельним підхідом

Таким чином, запропонована формалізація задачі, вибір цілочисельного способу кодування та обґрунтування структури хромосоми створюють підґрунтя для розробки відповідної модифікації алгоритму NSGA-III, що забезпечить ефективну реалізацію запропонованого методу багатокритеріального розподілу інформаційних потоків.

4.2 Розробка модифікації алгоритму NSGA-III для багатокритеріальної оптимізації розподілу потоків

Блок-схема модифікації ГА наведена на рис. 4.3. Зеленим кольором виділено етапи, які були удосконалені в межах даного наукового дослідження для забезпечення адаптивності алгоритму до змінного навантаження та врахування прогнозу сплесків.

Блок-схема показує основні кроки реалізації модифікованого алгоритму NSGA-III, що включає інтеграцію механізмів прогнозування, гібридну обробку обмежень, динамічне налаштування напрямів пошуку та адаптивну зміну частоти мутацій залежно від навантаження системи.

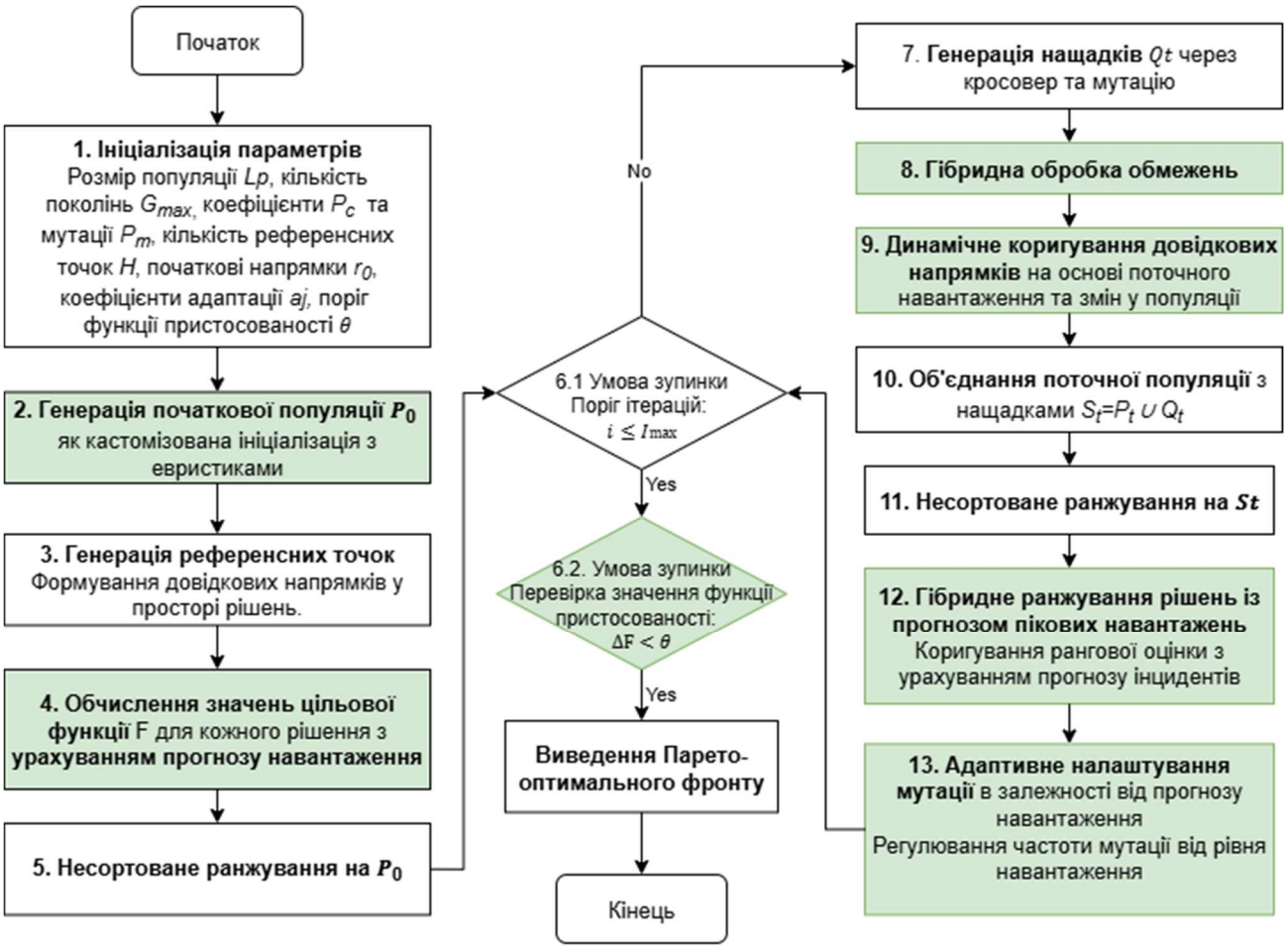


Рисунок 4.3 – Блок-схема удосконаленого алгоритму NSGA-III

Розглянемо основні етапи запропонованого модифікованого алгоритму більш докладно.

Етап 1. Ініціалізація параметрів.

На початковому етапі визначаються базові параметри генетичного алгоритму, необхідні для формування популяції та подальшого пошуку рішень. Зокрема, задаються:

– кількість рішень у популяції L_p;

– максимальна кількість поколінь G_{max};

– коефіцієнти кросоверу P_c та P_m;

– кількість референсних точок H, що визначають напрямки пошуку;

– початкові напрямки r_0;

– коефіцієнти адаптації a_j, які використовуються для гібридного налаштування параметрів на основі навантаження;

– поріг удосконалення значення функції пристосованості θ, що використовується як один із критеріїв завершення алгоритму.

На цьому етапі закладаються умови для забезпечення стабільності пошуку та чутливості алгоритму до динамічних змін середовища, оскільки правильний вибір початкових параметрів безпосередньо впливає на швидкість збіжності та якість отриманих рішень.

Етап 2. Генерація початкової популяції (удосконалення).

Початкова популяція P_0 формується не випадковим чином, а шляхом попереднього фільтрування множини допустимих рішень, що задовольняють умовам (формули – 4.12 – 4.14) з використанням евристичних стратегій (First-Fit, Best-Fit, Least-Loaded). Це дозволяє підвищити допустимість рішень і врахувати резервування ресурсів.

Додатково для початкової популяції запроваджується умова резервування критичних вузлів: мінімум один вузол у кожному кластері залишається вільним для обробки потенційних сплесків, що забезпечує початкову адаптивність системи.

Таким чином, кожне рішення $X_j \in P_0$ не лише задовольняє обмеження на ресурси, а також включає політику резервування:

$$\exists n \in S_C^n : \; CPU_n^{free} \geq \beta \cdot CPU_n \,, \qquad 4.16$$

де β – коефіцієнт зарезервованого ресурсу.

Етап генерації початкової популяції завершується формуванням множини рішень, придатних для подальшої еволюційної обробки.

Цей етап є модифікацією ГА. У класичному алгоритмі NSGA-III початкова популяція формується випадково, а в запропонованому алгоритмі застосовано евристичне формування початкової популяції з попереднім фільтруванням та

застосуванням політики резервування ресурсів. Це дозволяє вже на старті ГА підвищити допустимість рішень та забезпечити адаптивність системи до потенційних сплесків навантаження.

Етап 3. Генерація референсних точок.

На цьому етапі формується система референсних напрямків (reference directions) у просторі рішень, що забезпечує рівномірне покриття області Парето-оптимальних рішень та використовується для орієнтації алгоритму під час сортування.

Для їх генерації застосовується метод дискретизації симплексу з умовою:

$$\sum_{i=1}^{M} \lambda_i = 1, \qquad \lambda_i \geq 0, \tag{4.17}$$

де M – кількість цільових функцій, λ_i – координати напрямку.

Кількість референсних напрямків N визначається за формулою:

$$N = \binom{M + H - 1}{H}, \tag{4.18}$$

де H – параметр рівня дискретизації (задається користувачем),

Таким чином формується множина напрямків пошуку, що забезпечує рівномірний розподіл рішень уздовж фронту Парето.

Етап 4. Обчислення цільової функції (функції пристосованості) з урахуванням прогнозу навантаження (удосконалення).

На цьому етапі для кожного рішення (хромосоми) обчислюються значення цільових функцій $F = f_1, f_2, f_3, f_4$ з урахуванням не лише поточних навантажень, але й прогнозованих майбутніх пікових навантажень. Для прогнозування використовується показник P^{surge} імовірності сплеску навантаження (формула 4.7). З урахуванням імовірності обчислюється очікуване перевищення ресурсів (формули 4.4–4.6), що враховується при обчисленні f_3 – мінімізація очікуваного перевантаження при сплесках.

На відміну від стандартного NSGA-III, де цільові функції обчислюються тільки за поточними параметрами, у модифікованому алгоритмі до оцінки додано механізм прогнозування сплесків навантаження, що дозволяє враховувати ймовірні майбутні перевищення ресурсів при розрахунку f_3.

Етап 5. Несортоване ранжування початкової популяції.

На цьому етапі виконується багатокритеріальна оцінка популяції P_0 та формування множини фронтів Парето за принципом відсутності домінування (non-dominated sorting). Кожне рішення призначається до відповідного фронту залежно від рівня домінування. Тобто створюється структурована множина рішень для подальшого еволюційного відбору, що забезпечує збереження різноманіття та оптимальних властивостей популяції.

Етап 6. Перевірка умов завершення.

Перевіряються умови завершення алгоритму:

– досягнення максимальної кількості поколінь $i \leq I_{max}$, кількість ітерацій не перевищує допустиме;

– додаткова умова (модифікація алгоритму): приріст (удосконалення) значення функції пристосованості між останньою та поточною ітерацією є меншим за заданий поріг $\Delta F < \theta$.

Якщо будь-яка з умов виконується, алгоритм зупиняється і виводиться знайдений Парето-оптимальний фронт.

Етап 7. Генерація нащадків.

Створюються нові потенційні рішення (нащадки) за допомогою генетичних операторів Q_t – кросовера та мутації.

Етап 8. Гібридна обробка обмежень (удосконалення).

Виконується коригування недопустимих рішень (repair-оператор) шляхом заміни або перенаправлення потоків у межах доступних ресурсів. Для рішень, які все одно порушують обмеження, вводяться адаптивні штрафні коефіцієнти $penalty_{adapt}$, що враховують ступінь перевищення ресурсу та ймовірність сплесків навантаження.

Етап 9. Динамічне коригування довідкових напрямків (удосконалення).

Референсні напрямки оновлюються адаптивно відповідно до змін навантаження, що дозволяє алгоритму зміщувати пошук у ті області Парето-фронту, де рішення є більш стійкими до сплесків; позначаються як r_t^{adapt} – адаптивні референсні напрямки на ітерації t. Таке корегування дозволяє підвищити ефективність пошуку в умовах нестабільного навантаження.

Етап 10. Об'єднання поточної популяції з нащадками.

Поточна популяція та новозгенеровані нащадки об'єднуються для подальшого ранжування та відбору: $S_t = P_t \cup Q_t$. Це дозволяє забезпечити конкуренцію між попередніми та новими рішеннями під час формування наступного покоління.

Етап 11. Несортоване ранжування об'єднаної популяції.

Виконується ранжування об'єднаної популяції S_t, формується новий фронт рішень без домінування. Ранжування забезпечує розподіл рішень за рівнями Парето-оптимальності для подальшого відбору до наступної популяції.

Етап 12. Гібридне ранжування рішень із прогнозом пікових навантажень (удосконалення).

На цьому етапі додатково здійснюється уточнення ранжування рішень, враховуючи не лише поточний стан системи, але й очікуване пікове навантаження – показник P^{surge}. Результатом є гібридне ранжування $rank_t^{hy}$, що дозволяє точніше визначити пріоритетність рішень із урахуванням прогнозу ризиків майбутніх сплесків.

Етап 13. Адаптивне налаштування мутації з прогнозом навантаження (удосконалення).

На цьому етапі здійснюється адаптивне регулювання частоти мутацій P_m^{adapt} в залежності від прогнозованої імовірності сплеску навантаження P^{surge} Це дозволяє алгоритму гнучко адаптувати інтенсивність пошуку нових рішень:

– при високому рівні ризику перевантажень підвищується частота мутацій, що сприяє пошуку нестандартних варіантів;

– при низькому рівні ризику зберігається більш стабільна стратегія зниження випадковості.

Після виконання адаптивного налаштування мутації алгоритм завершує поточну ітерацію. Далі відбувається перехід до перевірки умов завершення (Етап 6), де аналізується досягнення максимального числа поколінь I_{max} або проводиться перевірка удосконалення значення функції пристосованості між останньою та поточною ітерацією ΔF. Якщо хоча б одна з цих умов виконується, алгоритм завершується та формується остаточний Парето-оптимальний фронт, який містить оптимальні рішення щодо розподілу потоків з урахуванням визначених критеріїв. Якщо умови завершення не виконано, запускається наступна ітерація (повернення на Етап 7) з подальшою генерацією нових нащадків, і весь цикл повторюється до моменту виконання умов завершення.

На основі описаних етапів алгоритму здійснено його експериментальну верифікацію, результати якої наведено в підрозділі 4.3.

4.3 Експериментальна оцінка ефективності методу багатокритеріального розподілу інформаційних потоків

Для верифікації ефективності методу багатокритеріального розподілу інформаційних потоків на основі модифікованого генетичного алгоритму NSGA-III було проведено експериментальне моделювання на основі графів і конвеєрів, сформованих за попередніми методами (розділи 2 та 3). Тестування проводилось у 100 ітерацій, чого достатньо для досягнення стаціонарної динаміки та статистично значущої оцінки результатів, оскільки після цього порогу показники алгоритмів стабілізуються.

В процесі моделювання порівнювалися три алгоритми.

1. «Без балансування» – використання лише побудованих конвеєрів без додаткової оптимізації.

2. «NSGA-III» – базовий генетичний алгоритм багатокритеріальної оптимізації.

3. «Удосконалений NSGA-III» – модифікація, що включає механізм гібридного ранжування та адаптивне регулювання частоти мутацій з урахуванням прогнозованих пікових навантажень.

Результати моделювання представлені в табл. 4.5 і на рис. 4.4.

Таблиця 4.5 – Експериментальна оцінка алгоритмів за навантаженням

Алгоритм	Без балансування	NSGA-III	Удосконалений NSGA-III
Перевантаження, % (max)	38,39	19,35	11,68
Перевантаження, % (min)	0	0	0
Середне перевантаження, %	17,097	7,0578	3,2712
Стандарт. відхилення перевантаження, % (stddev)	12,893695	6,776123	3,299384

З табл. 4.5 видно, що модифікований NSGA-III суттєво зменшує показники перевантаження, що підтверджує стійкість системи до перевантажень.

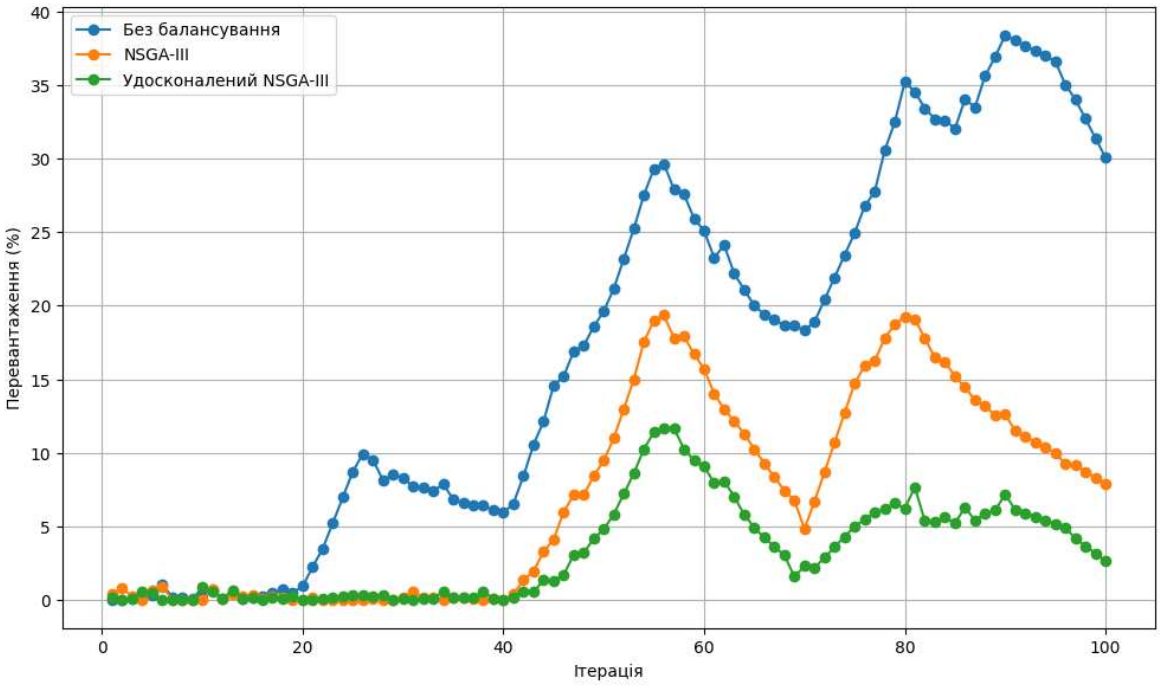


Рисунок 4.4 – Динаміка перевантаження вузлів за алгоритмами

На рис. 4.4 показано динаміку зміни відсотка перевантаження під час виконання 100 ітерацій. При відсутності балансування, перевантаження вузлів зростає і досягає 35%. Реалізація базового NSGA-III дозволяє знизити пікові перевантаження вдвічі, проте коливання навантаження залишаються суттєвими. Удосконалений NSGA-III показує стабільне зменшення перевантаження протягом усіх ітерацій, що підтверджує доцільність проведених модифікацій.

Експериментальне моделювання дозволило отримати наступні результати.

1. Без балансування: середнє перевантаження становило 17,1%, з високим стандартним відхиленням (12,9%) і піковими значеннями до 38%. Це доводить значну нестабільність і нерівномірність роботи.

2. Генетичний алгоритм NSGA-III: середнє перевантаження зменшилось до 7,1%, стандартне відхилення – до 6,8%, що свідчить про краще балансування ресурсів. Максимальне значення знизилось до 19%.

3. Удосконалений NSGA-III: експериментально доведено найкращі показники з трьох алгоритмів, а саме: середнє перевантаження – 3,3%, тобто на 80,9% нижче, ніж у випадку без балансування, і на 53,7% нижче, ніж у класичному NSGA-III. Стандартне відхилення знизилось у 2,1 рази в порівнянні з NSGA-III (3,3% проти 6,8%), що свідчить про найбільш рівномірний розподіл потоків. Максимальне перевантаження також скорочено – до 11,7%.

Таким чином, експериментальна оцінка підтвердила ефективність удосконаленого методу багатокритеріального розподілу потоків. Запропоновані модифікації забезпечили скорочення середнього перевантаження на 53,7% порівняно з класичним NSGA-III та більш ніж на 80% у порівнянні з варіантом без балансування, зменшення пікових значень до 11,7% та зниження стандартного відхилення удвічі. Це доводить здатність розробленого методу забезпечувати стійкість і рівномірність завантаження ресурсів у динамічних умовах функціонування розподіленої телекомунікаційної системи.

4.4 Обґрунтування використання баз даних для збереження результатів конвеєрної обробки

Результатом роботи алгоритмів, розроблених у розділах 2–4, є обчислювальні конвеєри, у яких обробка даних виконується поетапно, з формуванням структурованих результатів на кожному етапі. Вихідні дані кожного конвеєра підлягають збереженню у базах даних для подальшого аналізу, візуалізації чи передачі користувачам.

Для моделювання логіки зберігання складноструктурованих даних у межах потокових обчислювальних конвеєрів було використано підходи з [101, 103], адаптовані до архітектури розподіленої телекомунікаційної системи. Зокрема, застосовувалися методи нормалізації, денормалізації та графового моделювання, що дозволило отримати збалансовані рішення залежно від сценаріїв використання (табл. 4.6).

На основі аналізу табл. 4.6 було визначено особливості застосування кожного з методів [101, 103].

1. Нормалізація (MongoDB).

Для завдань, де важливою є узгодженість даних і потрібна можливість багаторазового їх оновлення, нормалізація забезпечує відсутність дублювань і чітку логічну структуру. Використання зовнішніх ключів дозволяє описати умовні зв’язки, компенсуючи відсутність механізмів цілісності у MongoDB. Недоліком є зростання кількості запитів, але це виправдано там, де пріоритет має коректність і відтворюваність даних.

2. Денормалізація (MongoDB).

Для етапів з інтенсивним читанням та записом (наприклад, підсумкові логи, масиви ознак чи сегментів), денормалізація забезпечує швидкий доступ до вкладених документів. Вона зменшує накладні витрати на запити й дає змогу працювати з великими масивами даних. Недоліком є дублювання та ускладнене оновлення, проте ці фактори не критичні для потоків, де дані переважно додаються, а не редагуються.

3. Графове моделювання (Neo4j).

Для представлення топології обчислювальних конвеєрів, взаємозв’язків між вузлами і маршрутів даних найбільш доцільним є графовий підхід. Neo4j дозволяє виконувати складні запити на знаходження шляхів та аналізувати залежності між даними. Попри складнішу підтримку 1:1 зв’язків, цей підхід особливо важливий для задач аналітики, оптимізації маршрутизації та соціальних мереж.

Таблиця 4.6 – Порівняльний аналіз методів логічного проектування

Метод	Особливості реалізації	Переваги	Недоліки	Придатність до РТС
Нормаліза-ція (MongoDB)	Використання модифікованої реляційної нотації; зовнішні ключі для відображення зв’язків	Висока цілісність даних; чітка структура	Зростання кількості запитів при складних операціях JOIN-подібного типу	Висока для критичних до консистент-ності завдань
Денормаліза-ція (MongoDB)	Використання вкладених документів; атрибутовані зв’язки	Швидкий доступ; менше операцій читання	Зайві дублікати; складність оновлень	Висока для даних, що мають потоковий характер (метрики, логи)
Графове моделювання (Neo4j)	Представлення у вигляді вузлів та ребер; обмеження на вкладеність	Оптимально для складних зв’язків; ефективність запитів на знаходження шляхів	Складніша підтримка 1:1 зв’язків на програмному рівні	Висока для завдань аналітики, маршрутизації та соціальних мереж

Проведений в табл. 4.6 аналіз показав, що жоден з методів логічного проектування не є універсальним: кожен із них має специфічні переваги й обмеження. Тому для архітектури розподіленої телекомунікаційної системи було здійснено адаптацію методів під характерні етапи потокових обчислювальних конвеєрів. Це дозволило обґрунтувати вибір нормалізованих,

денормалізованих або графових моделей залежно від типу даних, критичності до узгодженості та вимог до аналітики. Узагальнене відображення таких рішень наведено у табл. 4.7.

Таблиця 4.7 – Відображення етапів конвеєра – модель збереження

Етап конвеєра	Що зберігаємо	Рекомендована модель	Чому
Попередні перетворення (фільтри, нормування)	Очищені зображення або тестові та числові дані у вигляді логів	Денормалізована MongoDB	Максимальна продуктивність читання та запису, низька критичність до узгодженості
Виділення ознак/ сегментів	Масиви ознак/сегментів, маски або агреговані дані	Денормалізована MongoDB	Максимальна продуктивність запису великих масивів даних, ефективна структура даних для обробки агрегатних даних
Класифікація та аналітика (вихідні дані конвеєра)	Структуровані дані, отримані в результаті обробки та результати аналізу	Нормалізована MongoDB	Максимальна зручність зберігання даних та їх подальшого вилучення для демонстрації користувачам або для аналітичної обробки рушіями
Топологія конвеєрів/ маршрути між кластерами	Вузли-етапи, ребра-зв’язки/ тунелі, ваги	Neo4j (граф)	Оптимізоване зберігання мережевої топології, яка фактично є графовою структурою

На основі аналізу табл. 4.7 було визначено наступні етапи формування обчислювального конвеєра та особливості збереження даних.

1. Етап попередніх перетворень (фільтри, нормування).

На цьому етапі зберігаються очищені зображення, тестові або числові дані у вигляді логів. Для цього рекомендовано використовувати денормалізовану

модель MongoDB, оскільки вона забезпечує максимальну швидкість читання і запису, а також не потребує жорсткої узгодженості даних. Наприклад, ця модель дозволяє швидко зчитувати останні N варіантів обробки в межах одного запиту.

2. Етап виділення ознак/сегментів.

У цьому блоці накопичуються масиви ознак, сегменти, маски та агреговані дані. Найбільш ефективним є збереження у денормалізованій MongoDB, що дозволяє швидко записувати великі обсяги даних і формувати структури, зручні для подальшої агрегованої обробки. Зберігання масивів ознак в одному об’єкті гарантує зручність подальшої обробки і узгоджується з концептуальними принципами моделювання даних.

3. Етап класифікації та аналітики (вихідні дані конвеєра).

На виході формуються структуровані результати обробки і аналітичні дані. Тут доцільно застосовувати нормалізовану MongoDB, яка забезпечує зручний доступ і отримання даних для подальшої демонстрації користувачам або інтеграції з аналітичними рушіями. Зберігання даних окремо оптимізує їх зчитування при розрахунку визначних показників або візуалізації.

4. Етап графового подання результатів.

Результати класифікації та взаємозв’язки між об’єктами зберігаються у Neo4j (графова модель). Це дає можливість ефективно відображати топологічні залежності, виконувати пошук за зв’язками та будувати сценарії складних взаємодій у потоках даних.

Проведений аналіз і адаптація методів логічного проектування дозволили сформувати збалансовану схему збереження даних у межах потокових обчислювальних конвеєрів. Використання денормалізованих моделей для етапів попередніх перетворень та виділення ознак, нормалізованих моделей – для результатів класифікації та аналітики, а також графового моделювання для описання топології конвеєрів, забезпечує оптимальний баланс між швидкістю доступу, цілісністю даних та функціональними можливостями.

Висновки за розділом 4

1. Запропоновано метод багатокритеріального розподілу потоків на основі еволюційного підходу, який враховує множину конфліктних критеріїв, серед яких мінімізація використання ресурсів, зменшення кількості делегацій, зниження ризику перевантажень та забезпечення збалансованості.

2. Сформульовано систему обмежень та функцію пристосованості, яка відображає специфіку архітектури РТС з кластерами і механізмами делегування. Вибір цілочисельного кодування хромосоми дозволив суттєво знизити обсяг пам’яті та підвищити масштабованість представлення рішень у порівнянні з бінарним підходом.

3. Розроблено формалізацію задачі, вибір цільових функцій і структуру хромосоми, що створює обґрунтовану основу для розробки модифікації алгоритму NSGA-III. Це забезпечує адаптивність до динамічних умов, гнучке управління ресурсами і створює підґрунтя для практичного застосування у високонавантажених телекомунікаційних середовищах.

4. Розроблено модифікацію алгоритму NSGA-III, який на відміну від класичного варіанта враховує прогноз сплесків навантаження, застосовує евристичне формування початкової популяції з резервуванням ресурсів, гібридну обробку обмежень, динамічне налаштування напрямків пошуку та частоти мутацій.

5. Проведений експеримент довів, що запропонований алгоритм забезпечує зменшення середнього перевантаження вузлів більш ніж на 50% у порівнянні з класичним NSGA-III та на 80% у порівнянні з відсутністю балансування, зниження пікових значень перевантаження майже утричі та підвищення рівномірності завантаження ресурсів у 2 рази. Це обґрунтовує його адаптивність і стійкість у змінних умовах роботи РТС.

6. Обґрунтовано вибір моделей збереження даних у потокових обчислювальних конвеєрах РТС: з використанням денормалізованої та нормалізованої MongoDB та графової моделі Neo4j.

ЗАГАЛЬНІ ВИСНОВКИ

1. Проведено аналіз сучасних архітектур та методів розподілу навантаження у РТС, в результаті чого встановлено основні фактори, що знижують ефективність обробки даних, а саме: зміни в топології мережі, обмеженість в обчислювальних ресурсах та зростання обсягів даних. Визначено, що традиційні централізовані моделі не забезпечують необхідний рівень масштабування та адаптації під пікові навантаження. На основі виявлених недоліків сформульовано вимоги до розробки нових методів кластеризації, вибору головного вузла, самоорганізації та оптимізації розподілу потоків.

2. Розроблено комплексний метод самоорганізації РТС, який враховує фізичні характеристики мережі, різну продуктивність вузлів та навантаження на канали зв'язку. Метод працює на кількох рівнях і дозволяє ефективніше обробляти інформаційні потоки. У ході проведеного експерименту доведено, що запропонований метод дозволяє підвищити загальну продуктивність обробки даних на 21,4%, зменшити середній час затримки на 18,6%, а також знизити дисбаланс навантаження між вузлами у 2,3 рази порівняно з традиційним одношаровим підходом до кластеризації.

3. Удосконалено метод багатокрокової ієрархічної кластеризації на основі алгоритму Лувена за рахунок впровадження рекурсивного механізму з адаптивним параметром роздільної здатності; розроблено відповідний алгоритм реалізації. Проведено експериментальне моделювання кластеризації на графах реалістичного розміру (2000 вузлів). Запропонований метод показав кращий баланс між кількістю кластерів (на 174 % більше, ніж у Лувен, і на 36 % менше, ніж у Лейден), модульністю (на 1,7 % вища за Лейден), розміром спільнот і затримками. Досягнуто зниження міжкластерної затримки на 8,4 % і внутрішньокластерної – на 5,3 % у порівнянні з Лувен, а стандартне відхилення розмірів кластерів зменшено на 53,5 %.

Запропонована модифікація забезпечить стабільну та адаптивну роботу РТС в умовах нерівномірного навантаження, сприятиме локальній

реконфігурації кластерів без потреби в глобальному переформуванні мережі та знизить витрати на міжкластерну маршрутизацію, що є критично важливим для високонавантажених РТС.

4. У межах розробленого інтегрованого методу керування інформаційними потоками удосконалено метод вибору головного вузла у кластеризованому середовищі РТС шляхом розробки модифікованого варіанту алгоритму «Пліткар» (Gossip) з асинхронно-узгодженим вибором координатора. Запропонований підхід, на відміну від класичних виборчих процедур, не потребує явної фази виборів, що дозволяє: зменшити час відновлення керування після відмови координатора до 4–6 секунд за рахунок попереднього ранжування кандидатів; знизити службовий трафік до 30 % завдяки періодичному обміну мінімальними метриками замість широкомовних повідомлень; підвищити стійкість до втрати повідомлень і забезпечити масштабованість до кластерів розміром 100+ вузлів, що є критично важливим для РТС з динамічною топологією; гарантувати миттєве перепризначення координатора без централізованого управління завдяки локальному зберіганню списків заміщення.

В запропонованому алгоритмі реалізації методу враховано топологічну близькість вузлів, ресурсну потужність та унікальний ідентифікатор, що формалізовано у вигляді оціночної функції, яка містить ресурсно-затримкову оцінку вузлів. Реалізація механізмів TTL, кешування повідомлень та антиентропії забезпечує стабільну роботу в умовах циклів і фрагментації мережі.

5. Розроблено метод формування обчислювальних конвеєрів у кластеризованому середовищі РТС, який забезпечує поетапну маршрутизацію потоків даних між кластерами з урахуванням послідовності завдань, продуктивності вузлів і пропускної здатності міжкластерних з’єднань. Запропонований підхід дозволяє балансувати навантаження, зменшувати затримки передачі даних та оптимізувати використання ресурсів кожного кластера шляхом призначення ролі кожному з них в рамках конкретного етапу обробки. Метод враховує ресурсні обмеження та обсяг трафіку, що дозволяє уникнути перевантажень та забезпечити масштабовану обробку потоків в умовах

динамічних змін.

6. Розроблено алгоритм реалізації методу формування обчислювальних конвеєрів на основі спрямованого багатошарового графа, який враховує затримки між кластерами, ресурсні обмеження вузлів та штрафи за перевантаження. Запропонований алгоритм містить механізм паралельного розрахунку маршрутів із попередньою фільтрацією непрохідних та перевантажених вузлів, що дозволяє забезпечити високу ефективність при масштабуванні до тисяч потоків. Додатково реалізовано механізм пріоритетного вирішення конфліктів, що знижує кількість відхилених маршрутів і забезпечує стабільність обробки при пікових навантаженнях. Створена програмна модель підтверджує придатність алгоритму до реального часу та гнучкість до умов зміни трафіку в РТС.

7. Удосконалено метод багатокритеріального розподілу потоків у РТС на основі розробки модифікованого варіанта генетичного алгоритму NSGA-III, який поєднує гібридне ранжування особин, адаптивне регулювання частоти мутацій та використання прогнозу навантаження на кластери для динамічного вибору ефективної стратегії делегування задач. Проведений експеримент довів, що запропонований алгоритм забезпечує зменшення середнього перевантаження вузлів більш ніж на 50% у порівнянні з класичним NSGA-III та на 80% у порівнянні з відсутністю балансування, зниження пікових значень перевантаження майже утричі та підвищення рівномірності завантаження ресурсів у 2 рази. Це обґрунтовує його адаптивність і стійкість у змінних умовах роботи РТС. На цій основі обґрунтовано доцільність використання розподілених баз даних у якості платформи зберігання результатів обробки обчислювальних конвеєрів.

Перспективи подальших досліджень полягають у розробці методів прогнозного керування потоками в умовах змінної топології та навантаження, а також у створенні метрик оцінювання якості функціонування адаптивних РТС при гетерогенному трафіку та високих вимогах до реального часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бешлей Г., Бешлей М., Боднар С., Климаш М., Селюченко М. Розробка платформи для дослідження автоматичного масштабування контейнерів та балансування навантаження у розподілених системах // Інфокомунікаційні технології та електронна інженерія. – Львів : Видавництво Львівської політехніки, 2022. – Вип. 4. № 2. – С. 38–48.https://doi.org/10.23939/ictee2024.02.038

2. Гордійчук-Бублівська О. В., Бешлей М. І., Кирик М. І., Климаш М. М. (2021) Підвищення ефективності оброблення великих обсягів інформації з використанням методу розподіленого аналізу даних //Телекомунікаційні та інформаційні технології. 2021. No 2(71), С. 15-23, DOI:10.31673/2412-4338.2021.021523.

3. Долотов І.О, Гук Н.А. Кластеризація зваженого вебграфу із використанням модулярності //Питання прикладної математики і математичного моделювання. Випуск 23, С. 45-52, doi: 10.15421/322305

4. Доценко С. І., Мойсеєнко В. І., Єрмоленко Л. П. Розвиток методології моделювання інформаційно-керуючих систем на залізничному транспорті// Інформаційно-керуючі системи на залізничному транспорті. 2020. № 2 (141). С. 33–42.

5. Доценко С. І., Мойсеєнко В. І., Клименко Л.А, Єрмоленко Л. П. Обґрунтування методології формування моделей діяльності підприємства// Інформаційно-керуючі системи на залізничному транспорті. 2021. №2 С. 43–52. https://doi.org/10.18664/ikszt.v26i2.235402.

6. Доценко С.І.; Брежнєв Є.В.; Будніченко Є.М. (2021) Природні інтелектуальні системи: протиріччя методологій цілісного і системного підходів та шляхи їх подолання// Radioelectronic and Computer Systems, No 1 (2021), https://doi.org/10.32620/reks.2021.1.01

7. Жебка В.В., Анахов П.В. Моніторинг сталості інформаційно-телекомунікаційної системи і опрацювання заходів її захисту від небезпек// Метрологія та прилади. №1(87). С. 23-29., 2021, DOI: 10.33955/2307-2180(1)

8. Журавель С. Розробка імітаційної моделі мережі для оцінки ефективності розподіленого консенсусу з урахуванням нестабільності мережевих з'єднань// Інфокомунікаційні технології та електронна інженерія. – Львів : Видавництво Львівської політехніки, 2024. – Випуск 4. № 1. С. 10–19.

9. Захаржевський А. Г. Інваріантність системи управління інфокомунікаційної мережі спеціального призначення за умови неповної апріорної інформації // Телекомунікаційні та інформаційні технології. 2023. № 3. С. 411. DOI: 10.31673/2412-4338.2023.030411

10. Ільченко М. Ю., Капштик С. В., Мельник О. М., Наритник Т. М. Стан і перспективи використання та розвитку супутникових телекомунікацій в світі і Україні. Частина 1. Цифрові технології. 2016. № 20. С. 46.

11. Інтелектуальні кібернетичні системи: еволюція принципів, теорій та безпекових технологій: кол. моногр./за ред. С. І. Доценка, В. С. Харченка. – Міністерство освіти і науки України, Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ». – К. «Видавництво «Юстон», 2023. – 312 с.

12. Климаш М., Гордійчук-Бублівська О., Чайковський І., Костів О. (2022) Дослідження ефективності використання розподілених баз даних у системах IIoT // Інфокомунікаційні технології та електронна інженерія. – Львів : Видавництво Львівської політехніки, 2022. – Випуск 2. № 1. С. 12–18. doi.org/10.23939/ictee2022.01.012

13. Коваленко А., Мірошниченко Р., Мартинцов А. Розподілені обчислювальні системи на основі використання Grid технологій// Системи управління, навігації та зв'язку. Збірник наукових праць. 2023. № 1. С. 101. https://doi.org/10.26906/SUNZ.2023.1.101

14. Кучук Г.А., Коваленко А.А., Лукова-Чуйко Н.В. Метод мінімізації середньої затримки пакетів у віртуальних з'єднаннях мережі підтримки

хмарного сервісу // Системи управління, навігації та зв'язку. Збірник наукових праць. 2017. Вип. 2(42). С. 117-120.

15. Кучук Н. Г. Мерлак В. Ю., Скороделов В. В. Метод зменшення часу доступу до слабкоструктурованих даних // Сучасні інформаційні системи : щоквартальний науково-технічний журнал. 2020. Т. 4, № 1. С. 97-102. doi: 10.20998/2522-9052.2020.1.14

16. Кучук Н. Г. Метод розрахунку максимальних інтенсивностей інформаційних потоків у гіперконвергентній системі//Системи управління, навігації та зв'язку. Полтава: ПНТУ, 2019. Вип. 4(56). С. 53-56 http://nbuv.gov.ua/UJRN/suntz_2019_4_13

17. Математичне моделювання в розподілених інформаційно-керуючих системах залізничного транспорту: [Текст]: монографія / С. В. Лістровий, С. В. Панченко, В. І. Мойсеєнко, В. М. Бутенко. – Харків: ФОП Бровін О. В., 2017. – 220 с.

18. Методи і моделі метричного оцінювання якості інтелектуальних систем / І. О. Васильєв, С. І. Доценко, О. І. Морозова, В. С. Харченко // Методи та технології забезпечення якості та безпеки інтелектуальних систем : кол. моногр./за ред. В. С. Харченка, О. І. Морозової. – Міністерство освіти і науки України, Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ». – К. «Видавництво «Юстон», 2023. – 352 с. С. 193-224.

19. Мойсеєнко В. І., Бутенко В. М., Головко О. В., Чуб С. Г. Проблеми випробувань комплексів технічних засобів керування та регулювання руху поїздів // Інформаційно-керуючі системи на залізничному транспорті. 2020. № 3(141). С. 31–38.

20. Мойсеєнко В.І., Сафін В.Т, Жебко В.О., Некрасов О.Л. Аналіз методів визначення швидкості об'єктів систем гіркової автоматизації // Інформаційно-керуючі системи на залізничному транспорті. 2021. № 2. С. 9–14. https://doi.org/10.18664/ikszt.v26i2.235265.

21. Мрак В., Климаш М. Модель системи розпізнавання обличчя для нерухомих систем відеоспостереження // Measuring and Computing Devices in Technological Processes (1), 68–73. https://doi.org/10.31891/2219-9365-2024-77-9

22. Мрак В., Климаш М., Бабинець В. Удосконалення методу виявлення динамічних об’єктів у відео послідовностях // Measuring and Computing Devices in Technological Processes (2), 2024, 195–204. DOI: https://doi.org/10.31891/2219-9365-2024-78-22.

23. Олійник С.В., Бойко Б.М. Порівняння методів кластеризації для створення цільових груп клієнтів у наборі даних// Науковий вісник НЛТУ України, 2023, т.33, №5, https://doi.org/10.36930/40330510

24. Петровська І. Ю., Кучук Г. А. Розподіл обчислювальних ресурсів у хмарних системах // Системи управління, навігації та зв’язку. Полтава: Національний університет «Полтавська політехніка імені Юрія Кондратюка», 2022. Вип. 2 (68). С. 75–78. DOI: 10.26906/SUNZ.2022.2.75.

25. Пиріг Я., Климаш М., Пиріг Ю., Лаврів О. (2023) Генетичний алгоритм як засіб розв’язання оптимізаційних задач. //Інфокомунікаційні технології та електронна інженерія. – Львів: Видавництво Львівської політехніки, 2023. – Випуск 2. № 3. С. 95–107. https://doi.org/10.23939/ictee2023.02.095

26. Пиріг, Я. (2024) Пошук маршруту передачі даних у безпроводовій сенсорній мережі із використання генетичного алгоритму// Інфокомунікаційні технології та електронна інженерія, №2(4), 2024, с. 72-81. https://doi.org/10.23939/ictee2024.02.072

27. Пиріг, Я. (2024). Оцінка обчислювальної складності генетичного алгоритму //Інфокомунікаційні технології та електронна інженерія, №4(1), 2024, с. 52-60. https://doi.org/10.23939/ictee2024.01.052

28. Самотий В., Дзелендзяк У. Використання генетичних алгоритмів для апроксимації функцій дійсними поліномами // SCSIT. 2011, Volume 694: pp. 313 – 318.

29. Свиридов А. С., Коваленко А. А., Кучук Г. А. Метод перерозподілу пропускної здатності критичної ділянки мережі на основі удосконалення ON/OFF-моделі трафіку // Сучасні інформаційні системи. Т.2, № 2. 139–144. DOI: https://doi.org/10.20998/2522-9052.2018.2.24

30. Трубчанінова К.А., Крощенко Д.О. Метод попереднього планування безпроводової локальної мережі стандарту IEEE 802.11 // Інформаційно-керуючі системи на залізничному транспорті – Харків: УкрДУЗТ, 2020. – № 2(25). – С. 27–31.

31. Трубчанінова К.А., Черкашин Є.А. Система «розумного» освітлення// Інформаційно-керуючі системи на залізничному транспорті. 2023. № 2. С. 14–21.

32. Федоришин Б., Краско О. Міграція сервісів в кластері Kubernetes на основі прогнозування навантаження // Інфокомунікаційні технології та електронна інженерія, №2 (4), с. 82-92. https://doi.org/10.23939/ictee2024.02.082.

33. Abdallah, W. and Val, T. (2020), Genetic-Voronoi algorithm for coverage of IoT data collection networks, 30th International Conference on Computer Theory and Applications (ICCTA), Alexandria, Egypt, PP. 16–22. DOI: https://doi.org/10.48550/arXiv.2202.13735.

34. Abdelmoneem R. M., Abderrahim B., Shaaban Eman (2020) Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures. Computer Networks. Volume 179, 9 October 2020, P.P 107348 - 107367 https://doi.org/10.1016/j.comnet.2020.107348.

35. Abdulhafedh, A. (2021). Incorporating K-means, Hierarchical Clustering and PCA in Customer Segmentation. Journal of City and Development, 3(1), 12–30. https://doi.org/10.12691/jcd-3-1-3

36. Alam M., Ahmed N., Matam R., Mukherjee M., Barbhuiya F.A.: SDN-Based Reconfigurable Edge Network Architecture for Industrial Internet of Things, IEEE Internet of Things Journal, 2023.

37. Alenizi F., Rana O. (2020) Minimising Delay and Energy in Online Dynamic Fog Systems//Computer Science. Networking and Internet Architecture, https://doi.org/10.48550/arXiv.2012.12745.

38. Apat, H. K., & Mishra, D. K. (2018). Service Placement in Fog Computing Environment. International Journal of Computer Sciences and Engineering, 6 (12), 1-5. DOI:10.1109/ICIT.2018.00062

39. Attiya, H., & Welch, J. (2004). Distributed Computing: Fundamentals, Simulations, and Advanced Topics. A JOHN WILEY & SONS, INC., PUBLICATION, P. 414 http://lib.ysu.am/disciplines_bk/c95d04e111f3e28ae4cc589bfda1e18b.pdf

40. Bakhshi, R., et al. (2008) Leader Election in Anonymous Rings: Franklin Goes Probabilistic// IFIP International Federation for Information Processing, Volume 273; PP. 57–72, https://satoss.uni.lu/members/jun/papers/TCS08.pdf

41. BeaulahSoundarabai, P., Thriveni, J., Venugopal, K. R., & Patnaik, L. M. (2013). An Improved Leader Election Algorithm for Distributed Systems. International Journal of Next-Generation Networks (IJNGN) Vol.5, No.1, March 2013, https://airccse.org/journal/ijngn/papers/5113ijngn02.pdf

42. Belgacem Ali. Dynamic resource allocation in cloud computing: analysis and taxonomies. *Computing*. 2022. Vol. 104, is. 3. P .681–710. DOI: https://doi.org/10.1007/s00607-021-01045-2.

43. Bergmayr, A., Breitenbücher, U., Ferry, N., Rossini, A., Solberg, A., Wimmer, M.,Kappel, G., Leymann, F. (2018) A systematic review of cloud modelinglanguages. ACM Comput. Surv. 51 (1), 22:1–22:38. http://dx.doi.org/10.1145/3150227

44. Bernardino, R., & Paias, A. (2018). Metaheuristics based on decision hierarchies for the traveling purchaser problem. International Transactions in Operational Research, 25(4):1269–1295, doi: 10.1111/itor.12330.

45. Beshley H., Shkoropad Y., Beshley M. , Klymash M. (2022) Convergence of Heterogeneous Wireless Networks for Future Communications: Architecture, QoS and Resource Management. Information and Communication Technologies, Electronic Engineering. Volume 4, №2, 2022, PP. 1-13, DOI: 10.23939/ictee2022.02.020.

46. Beshley, M., Kryvinska, N., Beshley, H., Yaremko, O., Pyrih, J.: Virtual router design and modeling for future networks with QoS guarantees. Electronics 10, 1139 (2021). https://doi.org/10.3390/electronics10101139(2021)

47. Beshley, M., Kryvinska, N., Seliuchenko, M., Beshley, H., Shakshuki, E., Yasar, A. (2020) End-to-end QoS «smart queue» management algorithms and traffic prioritization mechanisms for narrow-band internet of things services in 4G/5G networks. Sensors 20(8), 2324-1–2324-30 (2020) https://doi.org/10.3390/s20082324

48. Blank J, Deb K. (2020) Pymoo: Multi-Objective Optimization. IEEE Access, Vol. 8, pp. 89497-89509, 2020, ISSN: 2169-3536, doi: 10.1109/ACCESS.2020.2990567

49. Blank J., Deb K., Dhebar Y., Bandaru S., SeadaH. (2012) Generating well-spaced points on a unit simplex for evolutionary many-objective optimization. IEEE Transactions on Evolutionary Computation, 25(1):48–60, 2021. doi:10.1109/TEVC.2020.2992387.

50. Blondel, V.D. et al. Fast unfolding of communities in large networks. J. Stat. Mech 10008, 1-12(2008). https://doi.org/10.1088/1742-5468/2008/10/P10008.

51. Bonomi F. , Milito R., Natarajan P., and Zhu J. (2014). Fog Computing: A Platform for Internet of Things and Analytics. Springer International Publishing, Cham, 169-186. DOI:10.1007/978-3-319-05029-4_7

52. Botta A, de Donato W, Persico V, Pescape A (2016) Integration of cloud computing and internet of things: a survey. Fut Gener Comput Syst 56:684–700

53. Boza, E.F., Abad, C.L., Narayanan, S.P., Balasubramanian, B., Jang, M., 2019. Acase for performance-aware deployment of containers. In: Proc. 5th Intl.Workshop on Container Technologies and Container Clouds. WOC'10, ACM,pp. 25–30. http://dx.doi.org/10.1145/3366615.3368355.

54. Butt A., Saba T., Khan I., Mahmood T., Rehman Khan A. K., Singh S-K.,Daradkeh Y., Ullah I. (2025) Proactive and data-centric Internet of Things-based fog computing architecture for effective policing in smart cities. Computers and Electrical Engineering Volume 123, Part B, April 2025, 110030 https://ieeexplore.ieee.org/document/10851425

55. Carrión, C., 2023. Kubernetes scheduling: Taxonomy, ongoing issues and chal-lenges. ACM Comput. Surv. 55 (7), 138:1–138:37. http://dx.doi.org/10.1145/3539606.

56. Casalicchio, E. (2019). A study on performance measures for auto-scaling CPU-intensive containerized applications. Cluster Computing, 22(3), 995–1006, DOI:10.1007/s10586-018-02890-1.

57. Chandrakant K., Piwowarek G. (2024) Consensus Algorithms in Distributed Systems, https://www.baeldung.com/cs/consensus-algorithms-distributed-systems

58. Chang, Y.-J., & Jiang, S. (2022). The Energy Complexity of Las Vegas Leader Election. Computer Science - Data Structures and Algorithms, DOI: 10.48550/arXiv.2205.08642

59. Chen J., Wang Y. A Hybrid Method for Short-Term Host Utilization Prediction in Cloud Computing. Journal of Electrical and Computer Engineering. 2019. P. 1-14. DOI: 10.1155/2019/2782349

60. Cheng R., Jin Y., Olhofer M., Sendhoff B. (2016) A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation, 20(5):773–791, 2016. doi:10.1109/TEVC.2016.2519378.

61. Cheok, S. M., Hoi, L. M., Tang, S. K. & Tse, R. (2022) Crawling parallel data for bilingual corpus using hybrid crawling architecture // Procedia Computer Science. 2022, DOI: https://doi.org/10.1016/j.procs.2021.12.218.

62. Choi T., Kang S., Yoon S., Yang S., Song S. et al. «SuVMF: Software-defined unified virtual monitoring function for SDN-based large-scale networks» in Proc. CFI, ACM, Tokyo, Japan, pp. 1–6, 2014.

63. Coulson, N. C., Sotiriadis, S., & Bessis, N. (2020). Adaptive Microservice Scaling for Elastic Applications. IEEE Internet of Things Journal, 7(5), 4195–4202.

64. Dastjerdi AV, Gupta H, Calheiros RN, Ghosh SK, Buyya R (2016) Fog computing: principles, architectures, and applications. In: Internet of things: principles and paradigms, chap. 4, Morgan Kaufmann

65. Deb K, Abouhawwash M. (2016). An optimality theory-based proximity measure for set-based multiobjective optimization. IEEE Trans. Evolutionary Computation, 20(4):515–528, 2016. doi:10.1109/TEVC.2015.2483590.

66. Deb K., Jain H. (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Transactions on Evolutionary Computation, 18(4):577–601, 2014. doi:10.1109/TEVC.2013.2281535.

67. Dolev, S., Israeli, A., & Moran, S. (1997). Uniform dynamic self-stabilizing leader election. IEEE Transactions on Parallel and Distributed Systems (Volume: 8, Issue: 4, April 1997), PP. 424 – 440, DOI: 10.1109/71.588622

68. Dolotov I. O., Guk N. A (2024) Constructing a website graph using the crawling procedure // Herald of Advanced Information Technology. Vol.7 No.4, 2024, PP. 384–392, DOI: https://doi.org/10.15276/hait.07.2024.27.

69. Dotsenko S., Brezhniev E., Nor D., Klymenko L., Hnatchuk A. (2024) Logical-semantic models and methods of knowledge representation: cases for energy management systems and SMR digital infrastructures. Radioelectronic and Computer Systems. No 2 (2024) https://doi.org/10.32620/reks.2024.2.17.

70. Dotsenko S., Illiashenko O., Kamenskyi S., Kharchenko V. (2021) Embedding an Integrated Security Management System into Industry 4.0 Enterprise Management: Cybernetic Approach. Digital Transformation, Cyber Security and Resilience of Modern Societies (PP.279-296),_DOI:10.1007/978-3-030-65722-2_17, ISSN 1314-2119 (online).

71. Dotsenko S., Illiashenko O., Kharchenko V., Morozova O. (2022) Integrated Information Model of an Enterprise and Cybersecurity Management System: From Data to Activity. International Journal of Cyber Warfare and Terrorism (IJCWT), Volume: 12, Issue: 2, P. 21 DOI:10.4018/IJCWT.305860, ISSN: 1947-3435.

72. Dugué N., Anthony P. (2015) Directed Louvain : maximizing modularity in directed networks. [Research Report] Université d'Orléans. 2015. DOI:10.13140/RG.2.1.4497.0328.

73. Elmokashfi A., Kvalbein A., Dovrolis C. (2010) On the Scalability of BGP: The Role of Topology Growth. IEEE Journal on Selected Areas in Communications, Vol. 28, No 8, pp. 1250-1261, October 2010. DOI:10.1109/JSAC.2010.101003.

74. Ferreira Aluizio, Neto Rocha (2021) Edge-distributed Stream Processing for Video Analytics in Smart City Applications//Federal University of Rio Grande do Norte Exact and Earth Sciences Center Department of Informatics and Applied Mathematics Graduate Program in Systems and Computing PhD in Computer Science. Natal-RN March 2021, P. 119 https://repositorio.ufrn.br/handle/123456789/32743.

75. Fortunato, S. (2010) Community detection in graphs. Physics Reports, Volume 486, Issues 3–5, February 2010, Pages 75-174, https://doi.org/10.1016/j.physrep.2009.11.002

76. Fowler M. Patterns of Enterprise Application Architecture. 1st Edition. – Addison-Wesley Professional, 2002. – 560 P. – ISBN 978-0321127426.

77. Fowler M. Patterns of Enterprise Application Architecture. 1st Edition. – Addison-Wesley Professional, 2002. – 560 P. – ISBN 978-0321127426.

78. Franti P. K-sets and k-swaps algorithms for clustering sets. Pattern Recognition. 2023. Vol. 139, No. 13. 109454. P. 1-29. DOI: 10.1186/s40537-018-0122-y

79. Fruchterman T.M.J., Reingold E. M. (1991) Graph drawing by force-directed placement// Software: Practice and Experience, Volume 21, Issue 11, November 1991, P.P. 1129-1164 https://doi.org/10.1002/spe.4380211102.

80. Gao, M. (2020). Opposite and Chaos Searching Genetic Algorithm Based for UAV Path Planning, 2020 IEEE 6th International Conference on Computer and Communications, China, P. 2364-2369

81. Gilbert, S., Robinson, P., & Sourav, S. (2019). Leader Election in Well-Connected Graphs. Computer Science. Distributed, Parallel, and Cluster Computing, https://doi.org/10.48550/arXiv.1901.00342

82. Gubbi J., Buyya R., Marusic S., Palaniswami M. J. F., (2013) Internet of Things (IoT): A vision, architectural elements, and future directions. Future

Generation Computer Systems on elsevier journal,vol. 29, no. 7, PP.. 1645-1660, 2013, http://dx.doi.org/10.1016/j.future.2013.01.010

83. Guerrero C., Lera I.·Juiz C. (2018) Genetic algorithm for multi-objective optimization of container allocation in cloud architectures. Journal of Grid Computing. 16-1, pp. 113-135. 01/03/2018, https://doi.org/10.1007/s10723-017-9419-x

84. Guerrero, C., Lera, I., & Juiz, C. (2019). A Lightweight Decentralized Service Placement Policy for Performance Optimization in Fog Computing. Journal of ambient intelligence and humanized computing. 10 – 6, pp. 2447 – 2464. SPRINGER HEIDELBERG, 01.06.2019 https://doi.org/10.48550/arXiv.2401.12699

85. Guk, N. A., Dykhanov, S., Matiushchenko, O. (2020) Algorithm for building a website model// Bulletin of V. N. Karazin Kharkiv National University Series// Mathematical Modeling. Information Technology. Automated Control Systems. 2020; 47: 2–3. DOI: https://doi.org/10.26565/2304-6201-2020-47-03.

86. Hirschberg, D. S., & Sinclair, J. B. (1980). Decentralized Extrema-Finding in Circular Configurations of Processors. Communications of the ACM, 1980, Vol 23, Issue 11, P. 627, https://dl.acm.org/doi/pdf/10.1145/359024.359029

87. Huang, T.; Zhao, R.; Bi, L.; Zhang, D.; Lu, C. (2021) Neural embedding singular value decomposition for collaborative filtering. IEEE Trans. Neural Netw. Learn. Syst. 2021, 33, 6021–6029. DOI: 10.1109/TNNLS.2021.3070853

88. Jin J., Gubbi J., Marusic S., Palaniswami M. (2014) An information framework for creating a smart city through internet of things, IEEE Internet of Things journal, vol. 1, no. 2, PP. 112-121, 2014. DOI:10.1109/JIOT.2013.2296516.

89. Kamtam S.B., Lu Q., Bouali F., Haas Olivier C. L., Birrell S. (2024) Network Latency in Teleoperation of Connected and Autonomous Vehicles: A Review of Trends, Challenges, and Mitigation Strategies. Sensors 2024, 24(12), 3957 https://doi.org/10.3390/s24123957

90. Khan, M. S., & Ahmad, I. (2016). A Dynamic Leader Election Algorithm for Decentralized Networks, Journal of Computer and Communications, Vol.4 No.1, January 2016, DOI: 10.4236/jcc.2016.41001.

91. Kharchenko, V., Ponochovnyi, Y., Dotsenko, S., Illiashenko, O., Ivasiuk, O. (2024). Models of Resilient Systems with Online Verification Considering Changing Requirements and Latent Failures. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (eds) System Dependability – Theory and Applications. DepCoS-RELCOMEX 2024. Lecture Notes in Networks and Systems, vol 1026. Springer, Cham. https://doi.org/10.1007/978-3-031-61857-4_9.

92. Kim, T. W., & Kim, T. Y. (1995). Predictable Leader Election Algorithm, Concurrent Engineering, Volume 3, Issue 3, https://doi.org/10.1177/1063293X9500300305

93. Kovalenko, A., Kuchuk H. Methods for synthesis of informational and technical structures of critical application object's control system. Advanced Information Systems, 2018, Vol. 2, No. 1, pp. 22–27, DOI: https://doi.org/10.20998/2522-9052.2018.1.04

94. Kowalski, D. R., & Mosteiro, M. A. (2021). Time and Communication Complexity of Leader Election in Anonymous Networks. Computer Science. Distributed, Parallel, and Cluster Computing, https://doi.org/10.48550/arXiv.2101.04400

95. Lavault, C., & Louchard, G. (2006). Asymptotic Analysis of a Leader Election Algorithm. Computer Science. Distributed, Parallel, and Cluster Computing, https://doi.org/10.48550/arXiv.cs/0607032

96. Lebesbye, T., Mauro, J., Turin, G., Yu, I.C., 2021. Boreas - a service scheduler foroptimal Kubernetes deployment. In: Hacid, H., Kao, O., Mecella, M., Moha, N.,Paik, H. (Eds.), Proc. 19th Intl. Conf. on Service-Oriented Computing. ICSOC2021, In: Lecture Notes in Computer Science, vol. 13121, Springer, pp.221–237. http://dx.doi.org/10.1007/978-3- 030-91431-8_14.

97. Lera I., Guerrero C., Juiz C. (2019) Availability-aware Service Placement Policy in Fog Computing Based on Graph Partitions. IEEE Internet of Things Journal. 6 - 2, pp. 3641-3651. IEEE-Inst Electrical Electronics Engineers Inc, 01.04.2019 https://doi.org/10.48550/arXiv.2401.12690

98. Lian Y., Peng P., Jiang K., Xu W. (2024) Multi-objective compression for CNNs via evolutionary algorithm Information SciencesVolume 661, March 2024, 120155 https://doi.org/10.1016/j.ins.2024.120155

99. Liu X., Zhang D. An Improved SPEA2 Algorithm with Local Search for Multi-Objective Investment Decision-Making. Appl. Sci. 2019. Vol. 9, P. 1675. DOI: https://doi.org/10.3390/app9081675

100. Liu, S., Zhang, R., Liu, C. et al. An improved PBFT consensus algorithm based on grouping and credit grading, 2023, https://doi.org/10.1038/s41598-023-28856-x

101. Lysechko V., Syvolovskyi I., Shevchenko B., Nikitska A., Cherneva G.: Research of modern NoSQL databases to simplify the process of their design. Academic journal: Mechanics Transport Communications, 2023, vol. 21, issue 2, article №2363, ISSN 2367-6620

102. Ma Z., Wang Y. (2019) Evolutionary Constrained Multiobjective Optimization: Test Suite Construction and Performance Comparisons. IEEE Transactions on Evolutionary Computation, 23(6): 972–986, December 2019. doi:10.1109/TEVC.2019.2896967.

103. Mazurova O.; Syvolovskyi I.; Syvolovska O. (2022) NoSQL Database Logic Design Methods for MongoDB and Neo4j. Innovative Technologies and Scientific Solutions for Industries// Journal article DOI: 10.30837/ ITSSI.2022.20.052 http://journals.uran.ua/itssi/article/view/262054.

104. Mahmud R. Toosi A.-N. (2021) Con-Pi: A distributed container-based edge and fog computing framework. IEEE Internet of Things Journal, pp. 1–1, 2021.https://arxiv.org/pdf/2101.03533

105. Mahmud R., Pallewatta S., Goudarzi M., Buyya R. (2021) iFogSim2: An Extended iFogSim Simulator for Mobility, Clustering, and Microservice Management in Edge and Fog Computing Environments. Computer Science. Distributed, Parallel, and Cluster Computinghttps, Sep 2021. //arxiv.org/pdf/2109.05636

106. Mazurova O.; Syvolovskyi I.; Syvolovska O. (2022) NoSQL Database Logic Design Methods for MongoDB and Neo4j. Innovative Technologies and

Scientific Solutions for Industries 2022-06-30// Journal article DOI: 10.30837/ITSSI.2022.20.052 http://journals.uran.ua/itssi/article/view/262054.

107. Mehyadin, A. E., Abdulrahman, L. M., Ahmed, S. H., Qashi, R. (2023)// Distributed fundamentals based conducting the web crawling approaches and types (focused, incremental, distributed, parallel, hidden web, form focused and breadth first) crawlers. Journal of Smart Internet of Things, 2023 (2): 10–32. DOI: https://doi.org/10.2478/jsiot-2022-0002.

108. Modeling of vehicle movement in computer information-control systems // V. Moiseenko, O. Golovko, V. Butenko, K. Trubchaninova. – Radioelectronic and computer systems, 2022. Pages 36 – 49. Open access – DOI: https://doi.org/10.32620/reks.2022.1.03 (Scopus).

109. Naha R.K., Garg S., Georgekopolous D., Jayaraman P.P., Gao L., Xiang Y., Ranjan R.: Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions, IEEE Access, vol. 6, pp. 47980-48009, 2018.

110. Neto A.R.: Edge-distributed Stream Processing for Video Analytics in Smart City Applications, 2021, 10.13140/RG.2.2.10968.57604.

111. Oluchukwu, U. E., Sylvanus, O. A., Ifeoma, M. A. O. & Chukwuemeka, M. O. (2021) Semantic web ontology technology and Its Impact on E-Learning // American Journal of Embedded Systems and Applications, 2021; 8 (2), PP. 9–11. https://doi.org/10.11648/j.ajesa.20210802.11.

112. Ongaro D., Ousterhout J. (2014) In Search of an Understandable Consensus Algorithm// https://web.stanford.edu/~ouster/cgi-bin/papers/raft-atc14.pdf

113. Palakonda V., Tursunboev J., Kang J., Moon S. (2025) Metaheuristics for pruning convolutional neural networks: A comparative study. Expert Systems with Applications Expert Systems with Applications Volume 268, 5 April 2025, 126326 https://doi.org/10.1016/j.eswa.2024.126326

114. Papageorgiou A, Cheng B, Kovacs E (2015) Real-time data reduction at the network edge of internet-of-things systems. In: 11th International Conference on Network and Service Management. Barcelona, Spain, pp 284–291

115. Petrovska, I., & Kuchuk, H. (2022). Static allocation method in a cloud environment with a service model IaaS. Advanced Information Systems, 6(3), 99–106. https://doi.org/10.20998/2522-9052.2022.3.13

116. Plakhtii O., Trubchaninova K., Panchenko V., Anannieva O. Improving the Energy Efficiency of Railway Electrical Traction by Creating Smart Grid System 2020 Transportation and Logistics P. 337-343 https://link.springer.com/chapter/10.1007/978-3-030-39688-6_42

117. Przystupa K., Pyrih J., Beshley M., Klymash M. (2021) Improving the Efficiency of Information Flow Routing in Wireless Self-Organizing Networks Based on Natural Computing. Energies 2021, 14(8), 2255; https://doi.org/10.3390/en14082255 https://www.mdpi.com/1996-1073/14/8/2255

118. Pyrih, Y., Kaidan, M., Tchaikovskyi, I., & Pleskanka, M. (2019). Research of Genetic Algorithms for Increasing the Efficiency of Data Routing 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT), Lviv, Ukraine, pp. 157-160, doi: 10.1109/AIACT.2019.8847814.

119. Qian H, Wu Y, Qin R, An X, Chen Y, Zhou A (2025) Provable space discretization based evolutionary search for scalable multi-objective security games. Swarm and Evolutionary Computation 10.1016/j.swevo.2024.10177092 (101770) https://doi.org/10.1016/j.swevo.2024.101770

120. Qu T, Lei SP, Wang ZZ, Nie DX, Chen X, Huang GQ (2016) IoT-based real-time production logistics synchronization system under smart cloud manufacturing. Int J Adv Manuf Technol 84(1):147–164

121. Rainville, F., Fortin, F., Gardner, M., Parizeau, M., Gagné, C. (2012), DEAP: a python framework for evolutionary algorithms // 14th annual conference companion on Genetic and Evolutionary Computation (GECCO '12). Association for Computing Machinery, New York, NY, USA, PP. 85–92. DOI: 10.1145/2330784.2330799.

122. Reichardt J., Bornholdt S. (2004) Detecting Fuzzy Community Structures in Complex Networks with a Potts Model // APS. Physical Review Letters, 93, 218701 – Published 15 November, 2004, https://doi.org/10.1103/PhysRevLett.93.218701

123. Rohjans S, Dnekas C, Uslar M (2012) Requirements for Smart Grid ICT-architectures. In: 3rd IEEE PES international conference and exhibition on innovative smart grid technologies. Berlin, Germany, pp 1–8.

124. Rzepka Michał, Boryło Piotr, Marcos D. Assunção, Artur Lasoń, Laurent Lefèvre (2022) SDN-based Fog and Cloud Interplay for Stream Processing Future Generation Computer Systems/ Volume 131, June 2022, P.P. 1-17 https://doi.org/10.1016/j.future.2022.01.006

125. Salaht, S., Desprez, F., & Lebre, A. (2020). An Overview of Service Placement Problem in Fog and Edge Computing. ACM Computing Surveys, 53(3), 1-35. DOI: 10.1145/3391196

126. Seada H., Deb K.. (2016) A unified evolutionary optimization procedure for single, multiple, and many objectives. IEEE Transactions on Evolutionary Computation, 20(3):358–369, June 2016. doi:10.1109/TEVC.2015.2459718.

127. Seifikar M., Farzi S., Barati M. (2020) C-blondel: An efficient louvain-based dynamic community detection algorithm. IEEE Transactions on Computational Social Systems. Vol. 7, no. 2. 2020. P. 308–318.

128. Semenov S., Sira O., Gavrylenko S., Kuchuk N. Identification of the state of an object under conditions of fuzzy input data. *Eastern-European Journal of Enterprise Technologies*. 2019. Vol 1, No 4 (97). P. 22-30. DOI: 10.15587/1729-4061.2019.157085

129. Serkov A.A., Lazurenko B.A., Trubchaninova K.A., Horiushkina A.E. Security Improvement Techniques for mobile applications of Industrial Internet of Things. IJCSNS International Journal of Computer Science and Network Security. Vol.20, No 5, 2020. – P. 145 – 149 (Web of Science Core Collection).

130. Sharaf, A. and Pillai, M. (2019), Genetic Algorithm Based Clustering Techniques in Wireless Sensor Networks: A Comprehensive Study// 1st International

Conference on Innovations in Information and Communication Technology (ICIICT), Chennai, India, pp. 1–5. DOI: 10.1109/ICIICT1.2019.8741485.

131. Simon B., Adrian P., Weber P.; Felka P.; Hinz O., Klein A. (2025) A Bargaining Approach for Service Placement in Multi-Access Edge Computing with Information Asymmetries. IEEE Transactions on Mobile Computing (Early Access).P.P: 1 - 17, 23 January 2025 https://ieeexplore.ieee.org/document/10851425

132. Sinaeepourfard A., Garcia J., Masip-Bruin X. , Marin-Tordera E. (2016) Estimating Smart City sensors data generation// Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Vilanova i la Geltru, 2016, PP. 1-8.

133. Sinaeepourfard A., Krogstie J., Petersen S-A., Ahlers D. (2019) F2c2C-DM: A Fog-to-cloudlet-to-Cloud Data Management Architecture in Smart City. 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), PP. 590-595 DOI:10.1109/WF-IoT.2019.8767226

134. Singh J., Singh P., Sukhpal S. (2021) Fog computing: A taxonomy, systematic review, current trends and research challenges Journal of Parallel and Distributed Computing Volume 157, November 2021, P.P. 56-85 https://doi.org/10.1016/j.jpdc.2021.06.005

135. Singh, S., Chana, I. (2016) A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges. J Grid Computing 14, 217–264 (2016). https://doi.org/10.1007/s10723-015-9359-2

136. Singh, S., Chana, I. (2016) Cloud resource provisioning: survey, status and future research directions. Knowl. Inf. Syst. 49(3), 1005–1069 (2016). DOI 10.1007/s10115-016-0922-3. URL http://dx.doi.org/10.1007/s10115-016-0922-3

137. Sivaram M., Yuvaraj D., Amin Salih Mohammed, Porkodi V., Manikandan V. The Real Problem Through a Selection Making an Algorithm that Minimizes the Computational Complexity. *International Journal of Engineering and Advanced Technology*. 2018. Vol. 8, iss. 2. P. 95-100.

138. Skarlat O., Nardelli M., Schulte S., Borkowski M., Leitner P. Optimized IoT service placement in the fog. Special Issue Paper. Open access. SOCA. Volume 11, pages 427–443, (2017) https://doi.org/10.1007/s11761-017-0219-8

139. Souto, M.C.B.; Das Chagas Moura, M.; Lins, I.D. Particle swarm-optimized support vector machines and pre-processing techniques for remaining useful life estimation of bearings. Eksploat. I Niezawodn.—Maint. Reliab. 2019, 21, 610–618. DOI: https://doi.org/10.17531/ein.2019.4.10.

140. Sreekanth G.R., Ahmed Najat Ahmed S., Sarac M., Strumberger I., Bacanin N., Zivkovic M.: Mobile Fog Computing by Using SDN/NFV on 5G Edge Nodes, Computer Systems Science and Engineering, vol. 41, pp. 751-765, 2021.

141. Srirama, S. N., Adhikari, M., & Paul, S. (2020). Application deployment using containers with auto-scaling for microservices in cloud environment. Journal of Network and Computer. Applications Volume 160, 15 June 2020, 102629, https://doi.org/10.1016/j.jnca.2020.102629

142. Swarnakar, S., Kumar, N., Kumar, A. and Banerjee, C. (2020). Modified Genetic Based Algorithm for Load Balancing in Cloud Computing. IEEE 1st International Conference for Convergence in Engineering (ICCE), Kolkata, India, pp. 255-259. doi: 10.1109/ICCE50343.2020.9290563

143. Syvolovskyi, I. M., Lysechko, V. P., Komar, O. M., Zhuchenko, O. S., Pastushenko, V. V. Analysis of methods for organizing distributed telecommunication systems using the paradigm of Edge Computing. 2024. *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, 1(75), P. 206-211, https://doi.org/10.26906/SUNZ.2024.1.206.

144. Syvolovskyi, I., & Komar, O. (2025). A method of multicriteria data stream distribution in telecommunication networks based on an evolutionary approach// Computer-integrated technologies: education, science, production/ Lutsk National Technical University. – Lutsk. – 2025. (№59), P.P. 230-239. https://doi.org/10.36910/6775-2524-0560-2025-59-41.

145. Syvolovskyi, I. M., Lysechko V. P. (2025) A method of hierarchical clustering of nodes in distributed telecommunication systems using graph algorithms// *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, Vol. 2, № 80 (2025), P.P. 255-262, https://doi.org/10.26906/SUNZ.2025.2.255.

146. Syvolovskyi I.M., Lysechko V.P. Method for leader node selection and processing pipeline formation in distributed telecommunication systems. – National Aviation University. Science-intensive Technologies. Series: «Electronics, telecommunications and radio engineering», Kyiv, 2025. Vol. 66, № 2, PP. 190-200 https://doi.org/10.18372/2310-5461.66.20311

147. Takagi T., Takadama K., Sato H. (2020) Incremental lattice design of weight vector set. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, GECCO '20, 1486–1494. New York, NY, USA, 2020. Association for Computing Machinery. https://doi.org/10.1145/3377929.3398082

148. Tel, G. (2000). Introduction to Distributed Algorithms. Cambridge University Press, https://doi.org/10.1017/CBO9781139168724

149. Traag, V.A., Waltman, L. & van Eck, N.J. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep 9, 5233 (2019). https://doi.org/10.1038/s41598-019-41695-z

150. Ulukök Mehtap Köse, Sarıyıldız İrfan, Evri Vesile (2024) Hybrid Raft-PoW Blockchain Consensus Algorithm// IEEE Access, P.P(99):1-1, DOI:10.1109/ACCESS.2025.3562725

151. Vesikar Y., Deb K., Blank J. (2018) Reference point based NSGA-III for preferred solutions. In 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 1587–1594. Nov 2018. doi:10.1109/SSCI.2018.8628819

152. Wang J., Li D., Adaptive computing optimization in software-defined network-based industrial internet of things with fog computing, Sensors, vol. 18, no. 8, pp. 1–14, 2018.

153. White paper: OpenFog Reference Architecture for Fog Computing. https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf, accessed: 2023-10-22

154. Ye Z, Zhou X, Bouguettaya A (2011) Genetic algorithm based QoS-aware service compositions in cloud computing. In: 16th International conference on database systems for advanced applica-tions (DASFAA) part II. Springer, Berlin, Heidelberg, Hong Kong, China, pp 321–334

155. Yiming Z., Fang X., Hordiichuk-Bublivska O., Beshley H., Beshley M. (2023) Modified Masking-Based Federated Singular Value Decomposition Method for Fast Anomaly Detection in Smart Grid Systems. Energies 2023, 16(16), 5996; https://doi.org/10.3390/en16165996

156. Yunak O., Shpur O., Strykhaliuk B., Klymash M. (2021) Algorithm forming randomized system of iterative functions by based cantor structure. Information and communication technologies, electronic engineering, Volume 1, №2, 2021, PP. 71-80 DOI: 10.23939/ictee2021.02.071

157. Zelentsov D., Koryashkina S., Stanina L. Solving Continual Two- Stage Problems of Optimal Partition of Sets. *International Journal of Research Studies in Computer Science and Engineering*. 2017. Vol. 4, Is. 4. PP. 72-80.

158. Zhao, L., Yin, Z., Yu, K., Tang, X., Xu, L., Guo, Z. & Nehra, P. (2023) A fuzzy logic based solution fornetwork traffic problems in migrating parallel crawlers // International Journal of Advanced Computer Science and Applications. 2023, 14 (2). DOI: https://doi.org/10.14569/IJACSA.2023.0140252.

159. Zitar, R. (2022), A Review of the Genetic Algorithm and JAYA Algorithm Applications //15th International Congress on Image and Signal Processing, (CISP-BMEI), Beijing, China, PP. 1–7. DOI: 10.1109/CISPBMEI56279.2022.9980332.

ДОДАТОК А

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковано основні наукові результати:

1. Mazurova, O., Syvolovskyi, I., & Syvolovska, O. (2022). NoSQL database logic design methods for MongoDB and Neo4j// Innovative technologies and scientific solutions for industries/Kharkiv national university of radio electronics/ №2 (20), 52–63. https://doi.org/10.30837/ITSSI.2022.20.052.

2. Syvolovskyi, I. M., Lysechko, V. P., Komar, O. M., Zhuchenko, O. S., Pastushenko, V. V. (2024) Analysis of methods for organizing distributed telecommunication systems using the paradigm of Edge Computing. 2024. *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, 1(75), P. 206-211, https://doi.org/10.26906/SUNZ.2024.1.206.

3. Syvolovskyi, I., & Komar, O. (2025). A method of multicriteria data stream distribution in telecommunication networks based on an evolutionary approach// Computer-integrated technologies: education, science, production/ Lutsk National Technical University. – Lutsk. – 2025. (№59), P.P. 230-239. https://doi.org/10.36910/6775-2524-0560-2025-59-41.

4. Syvolovskyi, I. M., Lysechko V. P. (2025) A method of hierarchical clustering of nodes in distributed telecommunication systems using graph algorithms// *National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems*, Vol. 2, № 80 (2025), P.P. 255-262, https://doi.org/10.26906/SUNZ.2025.2.255.

5. Syvolovskyi I.M., Lysechko V.P. (2025) Method for leader node selection and processing pipeline formation in distributed telecommunication systems. – National Aviation University. Science-intensive Technologies. Series: «Electronics, telecommunications and radio engineering», Kyiv, 2025. Vol. 66, № 2, PP. 190-200 https://doi.org/10.18372/2310-5461.66.20311.

Опубліковані праці апробаційного характеру:

6. Lysechko V., Syvolovskyi I., Shevchenko B., Nikitska A., Cherneva G. Research of modern NoSQL databases to simplify the process of their design. Mechanics, Transport, Communications. Journal article № 2363, vol. 21, issue 2, 2023 https://mtc-aj.com/library/2363_EN.pdf. ***[Міжнародне видання].***

7. Syvolovskyi I., Pastushenko V. Analysis of edge computing architectures in distributed telecommunication systems// Інформаційно-керуючі системи на залізничному транспорті: тези доповідей та виступів учасників 36-ї Міжнародної науково-практичної конференції «Інформаційно-керуючі системи на залізничному транспорті». – 2023. – № 3 (додаток). – С. 10-12.

8. Syvolovskyi I.M., Frolov D.Y. Analysis of load distribution methods in distributed telecommunication systems //Перспективи розвитку озброєння та військової техніки сухопутних військ. Збірник тез доповідей Міжнародної науково-технічної конференції (Львів 15-16 травня 2024). Національна академія сухопутних військ імені гетьмана Петра Сагайдачного, Issue 30 P. 308-309**.**

9. Syvolovskyi, I.M. Methods to improve the performance of distributed telecommunication systems by changing their architecture/ I.M. Syvolovskyi, O.S. Zhuchenko, R.O. Sarapin// Інформаційно-керуючі системи на залізничному транспорті: тези доповідей та виступів учасників 37-ї Міжнародної науково-практичної конференції «Інформаційно-керуючі системи на залізничному транспорті» (Харків, 10-11 жовтня, 2024 р.). – 2024. – № 3 (додаток). – С. 95-96.

10. Syvolovskyi I.M., Lysechko V.P. Methods of network optimization for streaming data processing in distributed systems // Тези доповіді за матеріалами I міжвузівською наукової конференції *«Стан та перспективи розвитку інфокомунікацій в сучасних умовах»*. Харківський національний університет повітряних сил ім. І. Кожедуба. Харків: 22 листопада 2024. С. 87-88.

11. Syvolovskyi I.M., Zhuchenko O.S. Methods of formation and routing of a distributed telecommunication networks // Тези доповіді за матеріалами науково-практичної конференції «Сектор безпеки і оборони України на захисті національних інтересів: актуальні проблеми та завдання в умовах воєнного

стану». Національна академія Державної прикордонної служби України імені Богдана Хмельницького, 22-23 листопада 2024, С. 709-712**.**

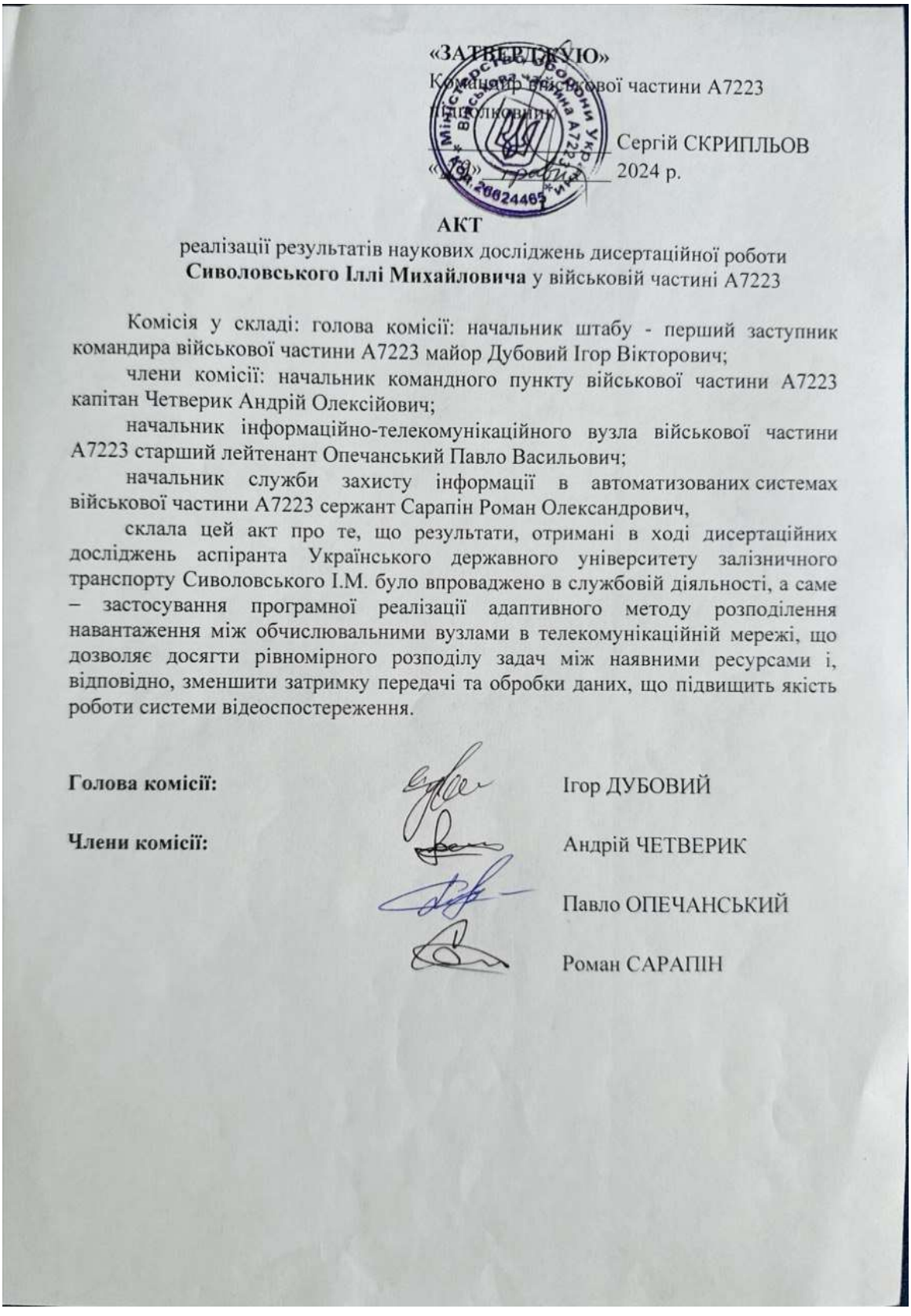
12. Syvolovskyi I.M., Lysechko V.P. Topology optimization method of a distributed telecommunication system for stream data processing // XXI Міжнародна наукова конференція «Новітні технології для захисту повітряного простору». Харківський національний університет повітряних сил ім. І. Кожедуба. Харків: 9 – 10 квітня 2025 року. С. 333-334.

13. Сиволовський І.М., Лисечко В.П., Пастушенко В.В. Багатокритеріальний розподіл інформаційних потоків у телекомунікаційних системах на основі модифікованого генетичного алгоритму// Перспективи розвитку озброєння та військової техніки сухопутних військ. Збірник тез доповідей Міжнародної науково-технічної конференції (Львів 14-15 травня 2025). Національна академія сухопутних військ імені гетьмана Петра Сагайдачного. – Львів: НАСВ, 2025, С. 91.

14. Мазурова О.О., Сиволовський І.М. Дослідження методів логічного моделювання NOSQL баз даних для ігрових серверних систем // Тези доповідей 12 міжнародної науково-технічної конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління», 27 – 28 квітня 2022 року, Том 2: секція 5, С.148-149.

ДОДАТОК Б

АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ НАУКОВИХ ДОСЛІДЖЕНЬ ДИСЕРТАЦІЙНОЇ РОБОТИ



«ЗАТВЕРДЖУЮ»
Командир військової частини А7223
підполковник
Сергій СКРИПЛЬОВ
«__» _____ 2024 р.

АКТ
реалізації результатів наукових досліджень дисертаційної роботи
Сиволовського Іллі Михайловича у військовій частині А7223

Комісія у складі: голова комісії: начальник штабу - перший заступник командира військової частини А7223 майор Дубовий Ігор Вікторович;

члени комісії: начальник командного пункту військової частини А7223 капітан Четверик Андрій Олексійович;

начальник інформаційно-телекомунікаційного вузла військової частини А7223 старший лейтенант Опечанський Павло Васильович;

начальник служби захисту інформації в автоматизованих системах військової частини А7223 сержант Сарапін Роман Олександрович,

склала цей акт про те, що результати, отримані в ході дисертаційних досліджень аспіранта Українського державного університету залізничного транспорту Сиволовського І.М. було впроваджено в службовій діяльності, а саме – застосування програмної реалізації адаптивного методу розподілення навантаження між обчислювальними вузлами в телекомунікаційній мережі, що дозволяє досягти рівномірного розподілу задач між наявними ресурсами і, відповідно, зменшити затримку передачі та обробки даних, що підвищить якість роботи системи відеоспостереження.

Голова комісії: Ігор ДУБОВИЙ

Члени комісії: Андрій ЧЕТВЕРИК

Павло ОПЕЧАНСЬКИЙ

Роман САРАПІН

«ЗАТВЕРДЖУЮ»

Проректор з науково-педагогічної роботи
Українського державного університету
залізничного транспорту
кандидат технічних наук, доцент
_____ Артур КАГРАМАНЯН
«__» _____ 2025 року

АКТ

впровадження у навчальний процес
Українського державного університету залізничного транспорту
результатів дисертаційного дослідження **Сиволовського Іллі Михайловича**

Комісія у складі:

голова – В.О. завідувача кафедри транспортного зв'язку, к.т.н., доц. Індик С.В.
члени – професор кафедри транспортного зв'язку, д.т.н., проф. Штомпель М.А.
– доцент кафедри транспортного зв'язку, к.т.н., доц. Жученко О.С.
– доцент кафедри транспортного зв'язку, к.т.н., доц. Єлізаренко А.О.

склала цей акт про те, що у навчальному процесі Українського державного університету залізничного транспорту при викладанні навчальних дисциплін за освітніми програмами першого (бакалаврського) та другого (магістерського) рівнів вищої освіти «Телекомунікації та радіотехніка» спеціальності 172 Електронні комунікації та радіотехніка: «Телекомунікаційні системи передачі», «Комп'ютерно-інтегровані мережеві системи», «Системи керування в телекомунікаціях», «Хмарні мережеві технології систем залізничного транспорту», а також при виконанні бакалаврських кваліфікаційних робіт було використано наступні результати дисертаційної роботи Сиволовського І.М.:

– розроблений метод ієрархічної кластеризації обчислювальних вузлів у розподілених телекомунікаційних системах, який, на відміну від традиційних підходів, враховує продуктивність вузлів, топологію мережі, затримки та пропускну здатність каналів зв'язку, що досягається за рахунок адаптивного налаштування параметрів кластеризації та багаторівневого аналізу мережевих зв'язків;

– удосконалений метод багатокрокової кластеризації на основі модифікованого алгоритму Лувена, який, на відміну від інших підходів, забезпечує контроль розміру та щільності кластерів завдяки динамічному регулюванню параметра роздільної здатності та рекурсивному уточненню кластерної структури;

– удосконалений метод вибору головного вузла та формування обчислювальних конвеєрів у РТС, який, на відміну від відомих підходів, поєднує мультипараметричну оцінку вузлів та модифікований алгоритм Gossip з детермінованим вибором координатора, що забезпечує оптимальне керування інформаційними потоками в умовах нестабільної мережі з мінімізацією затримок, за рахунок незначних додаткових витрат

на обчислення та адаптацію;

– удосконалений метод багатокритеріального розподілу потоків обробки даних у телекомунікаційних системах на основі модифікованого генетичного алгоритму NSGA-III, який використовує механізм гібридного ранжування та адаптивне регулювання частоти мутацій в залежності від прогнозованого навантаження, з метою високої точності розподілу та зниження числа делегацій між кластерами.

Впровадження результатів дисертаційної роботи Сиволовського І.М. дозволило збільшити науковий та методичний рівень вказаних навчальних дисциплін та сприяло удосконаленню навчального процесу.

Голова комісії:
В.О. завідувача кафедри
транспортного зв'язку
к.т.н., доцент _____ Сергій ІНДИК

Члени комісії:
д.т.н., професор, професор кафедри
транспортного зв'язку _____ Микола ШТОМПЕЛЬ
к.т.н., доцент, доцент кафедри
транспортного зв'язку _____ Олександр ЖУЧЕНКО
к.т.н., доцент, доцент кафедри
транспортного зв'язку _____ Андрій ЄЛІЗАРЕНКО

ДОДАТОК В

ФРАГМЕНТ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ДЛЯ ЕКСПЕРИМЕНТАЛЬНОЇ ПЕРЕВІРКИ БАГАТОРІВНЕВОЇ КЛАСТЕРИЗАЦІЇ НА ОСНОВІ МОДИФІКОВАНОГО АЛГОРИТМУ ЛУВЕНА

```
@dataclass
class ClusteringResult:
G: nx.Graph
partition: GraphPartition
def community_count(self) -> int:
return len(self.partition.communities)
def modularity(self) -> float:
return Modularity(1)(self.partition)
def get_max_community_size(self) -> int:
return max(self.get_community_lengths())
def get_min_community_size(self) -> int:
return min(self.get_community_lengths())
def get_community_count_mean(self) -> float:
return method_1.get_community_count_mean(self.partition.communities)
def get_community_count_stdev(self) -> float:
return method_1.get_community_count_stdev(self.partition.communities)
def get_average_latency(self) -> float:
return method_1.get_avg_latency(self.G)
def get_average_intracluster_latency(self) -> float:
return method_1.get_avg_intracluster_latency(self.G,
self.partition.communities)
def get_average_intercluster_latency(self) -> float:
return method_1.get_avg_intercluster_latency(self.G,
self.partition.communities)
def get_community_lengths(self) -> list[int]:
lengths = []
for i, nodes in enumerate(self.partition.communities):
lengths.append(len(nodes))
return lengths

def run_louvain_recursive_ranked(G: nx.Graph, quality_function =
Modularity(1), resolution_step = 0.5, count_upper_bound = 50,
count_lower_bound = 3, probe_count = 2) -> tuple[set, ...]:
print(f"Enter node count: {G.number_of_nodes()}")
```

```
success = False
p_res: GraphPartition = None
curr_quality_function = quality_function
for i in range(probe_count):
p_res = louvain(G, curr_quality_function, weight="weight")
contains_small_cluster = any(map(lambda c: len(c) <= count_lower_bound,
p_res.communities))
if not contains_small_cluster:
success = True
print(f"Probe successful. {[len(comm) for comm in p_res.communities]}")
break
else:
new_γ = max(0.5, curr_quality_function.γ - resolution_step / (2 ** (i + 1)))
print(f"Probe failed. Old mod: {curr_quality_function.γ}. New mod: {new_γ}.
{[len(comm) for comm in p_res.communities]}")
curr_quality_function = Modularity(new_γ)

if not success:
print("Return {1}")
return (set(G.nodes()),)

communities = ()
next_quality_function = Modularity(curr_quality_function.γ + resolution_step)
for community, nodes in enumerate(p_res.communities):
if (len(nodes) >= count_upper_bound):
subgraph = nx.induced_subgraph(G, nodes)
part_result = run_louvain_recursive_ranked(subgraph, next_quality_function,
resolution_step, count_upper_bound, count_lower_bound, probe_count)
communities += part_result
else:
communities += (nodes, )

return communities

tier_separated_nodes = {}
for n in G_internet.nodes():
tier = G_internet.nodes[n]["task"]



if tier not in tier_separated_nodes:
tier_separated_nodes[tier] = []
tier_separated_nodes[tier].append(n)
```

```
louvain_custom_internet_qual = Modularity(1)
total_communities = ()
for tier, tier_nodes in tier_separated_nodes.items():
if len(tier_nodes) < 2:
total_communities += (set([2000]), )
subgraph = nx.induced_subgraph(G_internet, tier_nodes)
louvain_custom_communities = run_louvain_recursive_ranked(subgraph,
louvain_custom_internet_qual, resolution_step=0.5,
count_upper_bound=50, count_lower_bound=1, probe_count=2)
total_communities += louvain_custom_communities

def louvain_custom_with_performance_check(*args, **kwargs):
random.seed(SEED)
start = timer()
communities = method_1. run_louvain_recursive_ranked(*args, **kwargs)
partition = GraphPartition. get_by_community_map(G_internet, communities,
"weight")
runtime = timer() - start
return partition, runtime

louvain_custom_communities = []
louvain_custom_times = []

for res in tqdm(resolutions, desc="Processing..."):
result, time = louvain_custom_with_performance_check(G_internet,
Modularity(res), resolution_step=0.5,
count_upper_bound=50, count_lower_bound=1, probe_count=2)
louvain_custom_communities.append(result)
louvain_custom_times.append(time)

print([len(part) for part in louvain_custom_communities])

louvain_custom_communities_mods = [mod(part) for part in
louvain_custom_communities]
louvain_custom_max_mod_idx = argmax(lambda x: x,
louvain_custom_communities_mods)[2]

louvain_custom_communities_counts = list(map(len, louvain_custom_communities))
louvain_custom_max_comm_idx = argmax(lambda x: x,
louvain_custom_communities_counts)[2]
```

```
louvain_custom_communities_stdev =
[method_1.get_community_count_stdev(part.communities) for part in
louvain_custom_communities]
louvain_custom_max_stdev_idx = argmax(lambda x: x,
louvain_custom_communities_stdev)[2]

louvain_custom_avg_intra_lats =
[method_1.get_avg_intracluster_latency(G_internet, part.communities) for part
in louvain_custom_communities]
louvain_custom_max_avg_intra_lat_idx = argmax(lambda x: x,
louvain_custom_avg_intra_lats)[2]

louvain_custom_avg_inter_lats =
[method_1.get_avg_intercluster_latency(G_internet, part.communities) for part
in louvain_custom_communities]
louvain_custom_max_avg_inter_lat_idx = argmax(lambda x: x,
louvain_custom_avg_inter_lats)[2]

print(f"Best result for Louvain (custom) algorithm with Mod.:
Modularity={louvain_custom_communities_mods[louvain_custom_max_mod_idx]:0.5f}
" +
f"with γ={resolutions[louvain_custom_max_mod_idx]:.3f}, yielding
{louvain_custom_communities_counts[louvain_custom_max_mod_idx]} communities")

# ...

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 4))
ax2.yaxis.set_tick_params(labelbottom=True)

x_loc = plticker.MultipleLocator(0.5, 0.5)

ax1.set_title("Modularity")
ax1.set_xlabel('Resolution $γ$ of quality function')
ax1.set_ylabel('Modularity of resulting partition')
ax1.grid(True)
ax1.xaxis.set_major_locator(x_loc)
louvain_line_mod, = ax1.plot(resolutions, louvain_communities_mods,
label='Louvain')
leiden_line_mod, = ax1.plot(resolutions, leiden_communities_mods,
label='Leiden')
louvain_custom_mod, = ax1.plot(resolutions, louvain_custom_communities_mods,
label='Louvain (modified)')
```

```
#leiden_custom_mod, = ax1.plot(resolutions, leiden_custom_communities_mods,
label='Leiden (modified)')
ax1.set_prop_cycle(None)
ax1.plot(resolutions[louvain_max_mod_idx],
louvain_communities_mods[louvain_max_mod_idx], 'x', markersize=8, label = "Max
value (Louvain)")
ax1.plot(resolutions[leiden_max_mod_idx],
leiden_communities_mods[leiden_max_mod_idx], 'x', markersize=8, label = "Max
value (Leiden)")
ax1.plot(resolutions[louvain_custom_max_mod_idx],
louvain_custom_communities_mods[louvain_custom_max_mod_idx], 'x',
markersize=8, label = "Max value (Louvain, modified)")
#ax1.plot(resolutions[leiden_custom_max_mod_idx],
leiden_custom_communities_mods[leiden_custom_max_mod_idx], 'x', markersize=8,
label = "Max value (Leiden, modified)")

ax1.set_xlim(left=0.5)

# ...

fig.tight_layout()
fig.subplots_adjust(wspace=0.4)
plt.show()
```

ДОДАТОК Г

ФРАГМЕНТ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕТОДУ ФОРМУВАННЯ КОНВЕЄРІВ ПОТОКОВОЇ ОБРОБКИ

```
@dataclass
class ResourceQuantities:
    def __init__(self, cpu: int, ram: int, storage: int):
        self.cpu = cpu
        self.ram = ram
        self.storage = storage

    def add(self, other: "ResourceQuantities") -> "ResourceQuantities":
        return ResourceQuantities(self.cpu + other.cpu, self.ram + other.ram,
self.storage + self.storage)

    def subtract(self, other: "ResourceQuantities") -> "ResourceQuantities":
        return ResourceQuantities(self.cpu - other.cpu, self.ram - other.ram,
self.storage - self.storage)

    def __copy__(self) -> "ResourceQuantities":
        cls = self.__class__
        cpy = cls.__new__(cls)
        cpy.cpu = self.cpu
        cpy.ram = self.ram
        cpy.storage = self.storage
        return cpy

    def __eq__(self, other: object) -> bool:
        if isinstance(other, "ResourceQuantities"):
            return self.cpu == other.cpu and self.ram == other.ram and
self.storage == other.storage
        return NotImplemented

    def __getitem__(self, key) -> int:
        if key in ['cpu', 'ram', 'storage']:
            match key:
                case 'cpu':
                    return self.cpu
                case 'ram':
                    return self.ram
                case 'storage':
```

```
                    return self.storage

stream_capacities = [
    ResourceQuantities(2_300, 980, 1_024),
    ResourceQuantities(3_930, 1_280, 2_500),
    ResourceQuantities(17_400, 4_580, 20_000),
    ResourceQuantities(38_000, 6_000, 45_000),
    ResourceQuantities(100_000, 100_000, 100_000),
]

cpu_penalty_coef = 0.5
ram_penalty_coef = 0.4
storage_penalty_coef = 0.1

stream_burst_coef = 1.2

weight_function : Callable[[float, int], float] = harmonic_mean_inv_latency

dataset_labels = {
    "Greedy": "Жадібний",
    "Dijkstra": "Дейкстра",
    "Modified Dijkstra": "Модифікований Дейкстра"
}

class PipelineStatistics:
    def __init__(self, pipeline_latencies_dict: dict[str, float],
load_percentages_dict: dict[str, float], overload_percentages_dict: dict[str,
float]):
        self.stream_count = len(pipeline_latencies_dict)
        self.node_count = len(load_percentages_dict)

        self.pipeline_latencies_dict = pipeline_latencies_dict
        self.load_percentages_dict = load_percentages_dict
        self.overload_percentages_dict = overload_percentages_dict

        self.pipeline_latencies = list(pipeline_latencies_dict.values())
        self.load_percentages = list(load_percentages_dict.values())
        self.overload_percentages = list(overload_percentages_dict.values())

        self.max_latency = max(self.pipeline_latencies)
        self.min_latency = min(self.pipeline_latencies)
        self.avg_latency = st.mean(self.pipeline_latencies)
```

```
        self.stddev_latency = st.stdev(self.pipeline_latencies)
        self.max_resource_load = max(self.load_percentages)
        self.min_resource_load = min(self.load_percentages)
        self.avg_resource_load = st.mean(self.load_percentages)
        self.stddev_resource_load = st.stdev(self.load_percentages)
        self.max_resource_overload = max(self.overload_percentages)
        self.min_resource_overload = min(self.overload_percentages)
        self.avg_resource_overload = st.mean(self.overload_percentages)
        self.stddev_resource_overload = st.stdev(self.overload_percentages)

def get_datacenter_node(G: nx.Graph):
    query = [n for n in G.nodes() if G.nodes[n]["task"] == 5]
    return (query or [None])[0]

def get_node_resources(G: nx.Graph, node_id) -> ResourceQuantities:
    cpu = G.nodes[node_id]["cpu"]
    ram = G.nodes[node_id]["ram"]
    storage = G.nodes[node_id]["storage"]
    node_resources = ResourceQuantities(cpu, ram, storage)
    return node_resources

def calculate_pipeline_statistics(G: nx.Graph, pipelines: dict[str, list]) ->
PipelineStatistics:
    latencies = [(id, nx.path_weight(G, path, 'latency')) for id, path in
pipelines.items()]
    latencies_dict = dict(latencies)

    load_statistics = calculate_load_statistics(G, pipelines)
    load_percentages = [(node, calculate_load_percentage(usage[0], usage[1]))
for node, usage in load_statistics.items()]
    load_percentages_dict = dict(load_percentages)
    overload_percentages = [(node, max(0, load_percentage - 1)) for node,
load_percentage in load_percentages_dict.items()]
    overload_percentages_dict = dict(overload_percentages)

    return PipelineStatistics(
        pipeline_latencies_dict=latencies_dict,
        load_percentages_dict=load_percentages_dict,
        overload_percentages_dict=overload_percentages_dict
    )
```

```
def calculate_load_statistics(G: nx.Graph, pipelines: dict[str, list]) ->
dict[str, (ResourceQuantities, ResourceQuantities)]:
    node_load_statistics = {}
    for n in G.nodes():
        if G.nodes[n]["type"] != 'S' and G.nodes[n]["task"] != 5:
            node_resources = get_node_resources(G, n)
            node_load_statistics[n] = (ResourceQuantities(0, 0, 0),
node_resources)

    for stream_id, path in pipelines.items():
        for path_node in path:
            if G.nodes[path_node]["type"] == 'S' or G.nodes[path_node]["task"]
== 5:
                continue

            task = G.nodes[path_node]["task"]
            stream_capacity = stream_capacities[task - 1]
            if path_node in node_load_statistics:
                current_used, current_total = node_load_statistics[path_node]
                new_used = current_used.add(stream_capacity)
                node_load_statistics[path_node] = (new_used, current_total)
            else:
                node_resources = get_node_resources(G, path_node)
                node_load_statistics[path_node] = (stream_capacity,
node_resources)
    return node_load_statistics


def calculate_load_percentage(in_use: ResourceQuantities, total:
ResourceQuantities):
    cpu_part = cpu_penalty_coef * (in_use.cpu / total.cpu)
    ram_part = ram_penalty_coef * (in_use.ram / total.ram)
    storage_part = storage_penalty_coef * (in_use.storage / total.storage)
    return cpu_part + ram_part + storage_part

def create_directed_graph(G: nx.Graph):
    G_exp = G.copy()
    to_remove = [(u, v) for u, v, type in G_exp.edges(data='type') if type !=
'EXT']
    G_exp.remove_edges_from(to_remove)
    return G_exp
```

```
def calculate_pipelines_greedy(G: nx.Graph) -> dict[str, list]:
    pipelines = {}
    end_node = get_datacenter_node(G)
    streams = [n for n in G.nodes() if G.nodes[n]["task"] == -1]
    G_exp = create_directed_graph(G)
    for stream in streams:
        current_position = stream
        path = [stream]
        for task in range(1, 6):
            adjacent_nodes = [(n, G.edges[current_position, n]["latency"]) for
n in G.adj[current_position] if G.nodes[n]["task"] == task]
            node, latency = min(adjacent_nodes, key=lambda x: x[1])
            path.append(node)
            current_position = node
        pipelines[stream] = path
    return pipelines

def calculate_pipelines_dijkstra(G: nx.Graph) -> dict[str, list]:
    pipelines = {}
    end_node = get_datacenter_node(G)
    streams = [n for n in G.nodes() if G.nodes[n]["task"] == -1]
    G_exp = create_directed_graph(G)
    for stream in streams:
        path = nx.shortest_path(G_exp, stream, end_node, weight='latency',
method='dijkstra')
        pipelines[stream] = path
    return pipelines

def calculate_pipelines_custom(G: nx.Graph) -> dict[str, list]:
    pipelines: dict[str, list] = {}
    load_states: dict[str, ResourceQuantities] = {}

    end_node = get_datacenter_node(G)
    streams = [n for n in G.nodes() if G.nodes[n]["task"] == -1]
    G_exp = create_directed_graph(G)

    for stream in streams:
        load_states = calculate_load_statistics(G, pipelines)
        recalculate_weights_using_load(G_exp, load_states)
        ###
        path = nx.shortest_path(G_exp, stream, end_node, weight='mod_weight',
method='dijkstra')
```

```
        pipelines[stream] = path
    return pipelines

def recalculate_weights_using_load(G: nx.Graph, load_states: dict[str,
ResourceQuantities]):
    for from_id, to_id in G.edges():
        G.edges[from_id, to_id]['mod_weight'] =
penalty_based_weight_function(G, load_states, from_id, to_id)

def penalty_based_weight_function(G: nx.Graph, load_states: dict[str,
ResourceQuantities], from_id: str, to_id: str):
    edge_latency = G.edges[from_id, to_id]['latency']

    from_node_task = G.nodes[from_id]['task']
    to_node_task = G.nodes[to_id]['task']
    if from_node_task > to_node_task:
        (from_id, to_id) = (to_id, from_id)
        (from_node_task, to_node_task) = (to_node_task, from_node_task)

    stream_resources = stream_capacities[to_node_task - 1]
    target_node_resources = get_node_resources(G, to_id)
    target_node_used_resources = load_states[to_id][0] if to_id in load_states
else ResourceQuantities(0, 0, 0)

    edge_penalty = sum([
        cpu_penalty_coef * max(0, (target_node_used_resources.cpu +
stream_burst_coef * stream_resources.cpu - target_node_resources.cpu) /
target_node_resources.cpu),
        ram_penalty_coef * max(0, (target_node_used_resources.ram +
stream_burst_coef * stream_resources.ram - target_node_resources.ram) /
target_node_resources.ram),
        storage_penalty_coef * max(0, (target_node_used_resources.storage +
stream_burst_coef * stream_resources.storage - target_node_resources.storage)
/ target_node_resources.storage),
    ])
    new_weight = edge_latency * (1 + edge_penalty)
    return new_weight
```