**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Кваліфікаційна наукова
праця на правах рукопису

**ЖУРАВЕЛЬ СТАНІСЛАВ СЕРГІЙОВИЧ**

УДК 621.391

# ДИСЕРТАЦІЯ

# МЕТОДИ ТА МОДЕЛІ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНИХ СЕРВІСНИХ СИСТЕМ

172 – Телекомунікації та радіотехніка
(шифр і назва спеціальності)

17 «Електроніка та телекомунікації»
(галузь знань)

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ / Журавель Станіслав Сергійович /

Науковий керівник Климаш Михайло Миколайович д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

ЛЬВІВ – 2025

# АНОТАЦІЯ

Журавель С.С. Методи та моделі підвищення ефективності функціонування розподілених сервісних сисиетм. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 172 – Телекомунікації та радіотехніка. – Національний університет «Львівська політехніка» Міністерства освіти і науки України, Львів, 2025.

В дисертаційній роботі розв’язано науково-практичне завдання підвищення ефективності функціонування розподілених сервісних систем шляхом розроблення методів та моделей адаптивного покращення функціонування алгоритмів консенсусу через удосконалення механізмів виявлення відмов, зокрема шляхом оптимізації таймаутів як основного інструменту детекції збоїв.

Метою представленої дисертаційної роботи є підвищення ефективності функціонування розподілених сервісних систем шляхом розроблення методів та моделей адаптивного покращення функціонування алгоритмів консенсусу через удосконалення механізмів виявлення відмов, зокрема шляхом оптимізації таймаутів як основного інструменту детекції збоїв.

Об'єктом дослідження є процес функціонування розподілених сервісних систем в умовах змінних мережевих параметрів.

Предметом дослідження є методи та моделі підвищення ефективності функціонування алгоритмів консенсусу, які забезпечують узгодженість та надійність розподілених сервісних систем, зокрема механізми виявлення відмов та адаптивного управління параметрами мережевої взаємодії.

У ході дослідження використано методи теорії графів, математичного моделювання, машинного навчання, ймовірнісного аналізу, алгоритмічного аналізу, імітаційного моделювання та методи оцінки ефективності алгоритмів консенсусу.

У вступі обґрунтовано актуальність теми дисертаційної роботи, констатовано зв’язок роботи з науковими програмами, темами, сформульовано

мету і завдання дослідження, наукову новизну та практичне значення отриманих результатів. Наведено дані про впровадження результатів роботи, її апробацію, публікації та особистий внесок здобувача.

У першому розділі «**Аналіз методів та алгоритмів консенсусу розподілених сервісних систем**» Проведено аналіз основних методів і моделей, які забезпечують функціонування розподілених сервісних систем. Встановлено, що основою узгодженості функціонування розподілених систем є робота алгоритмів консенсусу, що забезпечують їх стабільну роботу та є основним механізмом, який гарантує узгодженість даних між вузлами навіть у випадку збоїв. Забезпечення надійності розподілених сервісних систем є складною задачею, що вимагає комплексного підходу з урахуванням різноманітних джерел збоїв, зокрема апаратних, програмних та мережевих. Особливу складність становить високий рівень динамічності, масштабованості та гетерогенності сучасних середовищ, у яких мікросервісні додатки взаємодіють через нестабільні мережеві шляхи та можуть мати різну поведінку залежно від навантаження, що зумовлює необхідність впровадження інтелектуальних засобів моніторингу, адаптації та самовідновлення.

Ключовим елементом підтримки стійкості розподіленої системи є своєчасне виявлення збоїв (fault detection), що забезпечується застосуванням як традиційних методів контролю стану (наприклад, перевірка часів відповіді та heartbeat-механізмів), так і сучасних статистичних та машинних підходів до виявлення відмов. Ефективним вирішенням цієї задачі є ймовірнісні підходи, які дозволяють оцінювати ступінь ймовірності відмов залежно від поведінкових та мережевих метрик, що сприяє підвищенню точності реагування. Підвищення толерантності до збоїв досягається шляхом використання технік реплікації сервісів, балансування навантаження, резервного виділення ресурсів, а також застосування алгоритмів досягнення консенсусу для забезпечення узгодженості стану системи. Здатність системи до швидкої реконфігурації в умовах часткових

відмов є критичною передумовою збереження її функціональної цілісності та надання безперервних сервісів кінцевим користувачам.

Розглянуто основні переваги та недоліки при застосуванні існуючих методів забезпечення стабільної роботи алгоритмів консенсусу в розподілених системах із динамічно змінною структурою мережі. Сформульовано основні невирішені задачі в галузі забезпечення узгодженості та працездатності системи в умовах постійних відмов та реконфігурацій її функціональних вузлів. Визначено, що в умовах нестабільних і непередбачуваних мережевих затримок ефективність виявлення відмов критично залежить від раціонального вибору порогового значення часу очікування повідомлень від вузлів. Занижене значення тайм-ауту дозволяє системі оперативно реагувати на потенційні збої, але водночас підвищує ризик хибнопозитивних спрацювань, що може призводити до зайвих реконфігурацій, порушення узгодженості даних і зниження продуктивності системи. Таким чином, актуальною є розробка нових методів і моделей прогнозованого адаптивного виявлення відмов, які враховуватимуть поточні характеристики мережевого середовища, динаміку затримок і поведінку вузлів у кластері.

У другому розділі «**Розроблення адаптивних методів підвищення ефективності алгоритмів консенсусу в розподілених сервісних системах**» увагу зосереджено на вирішенні проблеми підвищення ефективності фунціонування алгоритмів консенсусу, особливо в умовах нестабільності функціонування мережевих з’єднань в розподілених системах, шляхом розроблення нових методів та алгоритів. Дослідження показало, що нестабільність мережі, така як затримки, втрата пакетів і переривання зв’язку, має суттєвий вплив на узгодженість даних та ефективність роботи системи. Хибні спрацьовування, викликані короткочасними мережевими збоями, часто призводять до помилкових переобрань лідера, додаткових витрат ресурсів та загального зниження продуктивності.

Як наслідок, набув подальшого розвитку метод виявлення відмов в роботі розподілених сервісних систем, який на відміну від існуючих аналізує стан функціонування системи та мінімізує кількість помилкових рішень про відмови, як викликані короткочасними мережевими збоями, шляхом інтеграції ймовірнісного байєсівського підходу. Використання теореми Байєса для аналізу наявності збоїв, дозволяє системі приймати більш обґрунтовані, ймовірнісні рішення щодо збоїв вузлів. Динамічно оновлюючи ймовірності збоїв, система може зменшити кількість хибнопозитивних спрацьовувань, тим самим підвищуючи надійність алгоритму консенсусу та дасть змогу забезпечувати стабільне функціонування розподілених сервісних систем.

Важливу роль у стабільності функціонування системи та роботі алгоритмів консенсусу, відіграють налаштування параметрів часу очікування вузлом на відповідь та інтервалу повідомлень працездатності (heardbeat). Надто короткий час очікування збільшують чутливість до короткочасних флуктуацій мережевої затримки та збільшують кількість реконфігурацій кластеру, тоді як надто тривалі – можуть затримувати реакцію на реальні збої. Для мінімізації кількості реконфігурацій удосконалено метод адаптивного визначення порогу відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA. Суть даного методу полягає у часовому прогнозуванні затримки часу відповіді в мережі, та визначенні порогу виявлення відмов. Основою прогнозування є статистичний аналіз даних, отриманих під час роботи системи. Кожен вузол у мережі регулярно збирає та аналізує інформацію про затримки, пов’язані з його з’єднаннями з іншими вузлами. Використовуючи історичні дані та модель машинного навчання SARIMA, вузли обчислюють очікувану затримку через часовий аналіз, що надає їм можливість прогнозувати потенційні відмови. Маючи набір значень прогнозованих затримок, кожне значення якого відповідає певному вузлу, вузол може ухвалити рішення щодо його несправності шляхом коригування порогу тайм-ауту.

Для пришвидшення процесу коригування порогу тайм-ауту вперше запропоновано автономний адаптивний метод підвищення продуктивності РСС, який на основі знань про лідер-орієнтовану структуру алгоритму консенсусу, дозволяє динамічно коригувати параметри затримки між вузлами кластеру, обирати лідера в кластері, та забезпечити максимальну стабільність і продуктивність роботи системи вцілому. Ключовою особливістю методу є використання мережевої затримки як основного критерію для визначення пріоритетності вузлів на роль лідера. Кожен вузол аналізує час затримки до інших вузлів більшості (кворуму), оцінює свою потенційну придатність до виконання ролі лідера та, залежно від цього, адаптує значення часу очікування на повідомлення працездатності (heardbeat). Таким чином, вузли з гіршими мережевими характеристиками штучно сповільнюють своє ініціювання виборів, поступаючись вузлам із кращими умовами з'єднання. Додатково у формування часу очікування інтегровано стохастичну компоненту, діапазон якої задається адміністративно та масштабується відповідно до поточних умов у мережі. Це дозволяє зменшити ймовірність одночасного спрацювання тайм-аутів у кількох вузлах, що є однією з основних проблем базових реалізацій таких алгоритмів як Raft.

У третьому розділі "**Моделювання та дослідження продуктивності методів підвищення ефективності розподілених систем**" для оцінки ефективності функціонування розподіленої системи в умовах динамічного середовища та непередбачуваних мережевих затримок на основі методу виявлення відмов з використанням ймовірнісного байєсівського підходу проведено розрахунок апостеріорної ймовірності відбою вузла. Проведені розрахунки доводять, що навіть за умов наявності окремих ознак потенційної відмови, таких як пропущені повідомлення працездатності (heardbeat) або підвищена затримка, метод не схильний до передчасних висновків, а натомість дозволяє зважено формувати рішення на основі комбінації факторів. Такий підхід значно зменшує кількість хибнопозитивних спрацьовувань приблизно на

4–5%. Це покращення досягається завдяки більш гнучкому, накопичувальному аналізу подій та адаптивному порогу, що враховує історію поведінки вузла і додаткові ознаки збою.

Для оцінки ефективності виявлення відмови в розподіленій сервісній системі системи, що функціонує на основі методу адаптивного визначення порогу відмов з використанням моделі машинного навчання SARIMA розроблено імітаційну модель структури IoT кластера, як частини РСС. В результаті роботи моделі отримано залежність, аналіз якої свідчить, що при прогнозуванні тривалостей часу відповіді за допомогою моделі машинного навчання SARIMA вдається динамічно коригувати поріг відмов та чітко розмежовувати пікові моменти функціонування системи та відмови в обслуговуванні. Це дає змогу краще враховувати зміни навантаження та зменшує кількість хибних спрацювань. Після проведення моделювання із використанням методу адаптивного визначення порогу відмов та моделі машинного навчання SARIMA вдалося підвищити точність виявлення збоїв на 20%, забезпечуючи при цьому адаптивну та ефективну реакцію системи на нестабільні умови функціонування мережі.

Імітаційне моделювання кластеру розподіленої сервісної системи для визначення потенційного лідера з використанням запропонованого автономного адаптивного методу засвідчило його ефективність у динамічному мережевому середовищі. Результати моделювання продемонстрували здатність алгоритму визначення тайм-ауту точно і гнучко визначати вузол, який має найсприятливіші умови для виконання ролі лідера, з урахуванням фактичних характеристик мережі. Обчислені значення тайм-аутів показали адаптивність підходу до поточних умов: мінімізація часу реакції на відмову чинного лідера, водночас із достатнім буфером часу для уникнення хибних спрацювань в умовах флуктуацій затримок. Для повноцінної валідації методу у наступному розділі буде проведено подальше моделювання та практична впровадження за різноманітних сценаріїв та умов функціонування.

У четвертому розділі “**Практична реалізація та функціонування розроблених методів підвищення ефективності розподілених сервісних систем**” завдяки інтеграції автомномного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем у імітаційну модель кластеру вдалося досягти зменшення ймовірності передчасної реконфігурації системи у 3 рази у стандартному режимі функціонування. Водночас у періоди пікового навантаження впровадження даного підходу забезпечує додаткове підвищення ймовірності реконфігурації на 5,6%, що свідчить про своєчасну реакцію системи на динамічну зміну структури з’єднань, підкреслює автономність та адаптивність методу, і як наслідок, можливість забезпечити узгодженість роботи кластеру незалежно від умов її функціонування.

Розроблено програмно-апаратний комплекс з автономним адаптивним підходом до управління для підвищення ефективності функціонування розподіленої системи, що дозволить на практиці підтвердити ефективність запропонованих методів та алгоритмів, залучаючи при цьому не лише програмну складову, а й комплекс реального мережевого обладнання. В результаті роботи розробленого програмно-апаратного комплексу підтверджується необхідність ефективного адаптивного підходу до коригування параметру затримки між вузлами кластеру, що впливає на вибір лідера в кластері, та зменшення ймовірності реконфігурації кластеру і, як, наслідок, підвищення якості надання послуг.

Завдяки впровадженню автономного адаптивного методу у роботу програмно-апартного комплексу вдалося зменшити стандартне відхилення затримки на 44%, що вказує на підвищену стабільність і передбачуваність роботи системи. Як наслідок, вдалося досягти зменшення ймовірності реконфігурації кластеру на 4.5%, що дозволить обробляти запити в моменти пікових навантажень без переобрання нових лідерів. Ефективна інтеграція та застосування методів виявлення відмов та адаптивного визначення порогу

відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA дозволила підвищити швидкість обробки повідомлень системою на 19.5%, що свідчить про здатність адаптивної системи краще утилізовувати ресурси без втрати стабільності.

Висновки до дисертації включають узагальнені результати дослідження та рекомендації щодо їх практичного застосування. Зокрема, розроблені науково-прикладні рішення щодо інтеграції ймовірнісного підходу до виявлення відмов, адаптивного визначення порогу спрацювання на основі моделі машинного навчання (SARIMA) та автономного адаптивного методу виявлення відмов можуть бути використані науково-дослідними установами, розробниками розподілених сервісних систем і провайдерами хмарних рішень для підвищення надійності, продуктивності та стабільності роботи кластерів у середовищах із високою динамікою мережевих умов.

Наукові та практичні результати виконаних досліджень використані в навчальному процесі кафедри інформаційно-комунікаційних технологій Національного університету «Львівська політехніка» для модернізації курсу лекцій для здобувачів другого (магістерського) рівня вищої освіти спеціальності 172 Електронні комунікації та радіотехніка ОНП «Телекомунікації та радіотехніка» з дисципліни «Розподілені сервісні системи та Cloud-технології» та лабораторних робіт з дисциплін «Розподілені інфомаційно-комунікаційні мережі» для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Інформаційно-комунікаційні системи».

Основні результати дисертаційної роботи використано і впроваджено з метою підвищення ефективності функціонування розподілених сервісних систем при їх інтеграції у сучасні інформаційно-комунікаційні сервісно-орієнтовані мережі, зокрема ТзОВ «Вітертек» та ТзОВ ВТФ «Контех», що підтверджено актами впровадження.

**Ключові слова:** Розподілені сервісні системи, алгоритми консенсусу, виявлення відмов, адаптивне управління, ймовірнісний аналіз, стабільність системи, машинне навчання, лідер-орієнтовані алгоритми, імітаційне моделювання, мережеві затримки, автономні системи

Список публікацій здобувача:

**Наукові праці, у яких опубліковані основні результати дисертації**

1. Zhuravel S., Klymash M., Shpur O., Mrak V. “Reducing the impact of unstable connections among nodes of wireless IIoT clusters using machine learning methods”, Lecture Notes in Electrical Engineering. Vol. 1198: Digital ecosystems: interconnecting advanced networks with AI applications. P. 144–159, 2024. doi: 10.1007/978-3-031-61221-3_8

2. S. Zhuravel, O. Shpur, and M. Klymash, “Adaptive consensus algorithms: Designing for durability against unstable network connections,” International Journal of Computing, vol. 23, no. 4, pp. 574–582, 2024. doi: 10.47839/ijc.23.4.3756.

3. N. Peleh, S. Zhuravel, O. Shpur, and O. Rybytska, “Structured and unstructured log analysis as a method to detect DDoS attacks in SDN networks,” Internet of Things (IoT) and Engineering Applications, vol. 6, no. 1, pp. 1–9, 2021. doi: 10.23977/iotea.2021.060101.

4. S. Zhuravel, “Development of network simulation model for evaluating the efficiency of distributed consensus taking into account the instability of network connections,” Infocommunication Technologies and Electronic Engineering, vol. 4, no. 1, pp. 10–19, 2024. doi: 10.23939/ictee2024.01.010.

5. S. Zhuravel, O. Shpur, and Y. Pyrih, “Method of achieving consensus in distributed service systems,” Infocommunication Technologies and Electronic Engineering, vol. 2, no. 2, pp. 58–66, 2022. doi: 10.23939/ictee2022.02.058.

6. Журавель С., Думич С., Шпур О. “Дослідження методів збору та обробки даних в розподілених інформаційних системах” Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. vol. 1, № 1, с. 20–38, 2021. doi: 10.23939/ictee2021.01.020

**Наукові праці, які засвідчують апробацію матеріалів дисертації**

7. S. Zhuravel, “Adaptive probabilistic methods for boosting the reliability of consensus algorithms in distributed systems,” in Proc. 2024 IEEE 17th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 2024, pp. 145–149. doi: 10.1109/TCSET64720.2024.10755585.

8. M. Klymash, O. Shpur, O. Lavriv, and S. Zhuravel, “Achieving consistency and consensus of distributed infocommunication systems,” in Proc. 2022 IEEE 16th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 2022, pp. 386–389. doi: 10.1109/TCSET55632.2022.9767019.

9. M. Klymash, S. Zhuravel, O. Shpur “Machine learning approaches to stabilize wireless IIoT clusters facing network instabilities” Наукоємні технології в інфокомунікаціях: збірник наукових праць Міжнародної науково-практичної конференції HICT'2023, Харків - Кам'янець-Подільський, Україна, 2023, с. 90–92.


**ABSTRACT**

Zhuravel S.S. Methods and models for enhancing the efficiency of distributed service systems. – Qualification research paper as a manuscript.

The thesis for the Doctor of Philosophy Degree in the specialty 172 – Telecommunications and Radio Engineering. – Lviv Polytechnic National University, Ministry of Education and Science of Ukraine, Lviv, 2022.

This thesis addresses the scientific and practical task of improving the efficiency of distributed service systems by developing methods and models for the adaptive enhancement of consensus algorithm performance through the refinement of failure detection mechanisms, in particular by optimizing timeouts as the primary tool for failure detection.

The aim of the presented dissertation is to improve the efficiency of distributed service systems by developing methods and models for adaptively enhancing the functioning of consensus algorithms through the improvement of failure detection mechanisms, specifically by optimizing timeouts as the core mechanism for detecting faults.

The object of the study is the functioning process of distributed service systems under conditions of varying network parameters.

The subject of the study is the methods and models for improving the efficiency of consensus algorithms that ensure the consistency and reliability of distributed service systems, including mechanisms for failure detection and adaptive management of network interaction parameters.

The research employs methods of graph theory, mathematical modeling, machine learning, probabilistic analysis, algorithmic analysis, simulation modeling, and methods for evaluating the efficiency of consensus algorithms.

The introduction substantiates the relevance of the dissertation topic, outlines its connection to scientific programs and themes, and formulates the research goal and objectives, scientific novelty, and practical significance of the obtained results. It also presents information on the implementation of the research outcomes, their validation, related publications, and the author's personal contribution.

Chapter one **"Analysis of methods and algorithm for ensuring consistency in distributed service systems"** presents an analysis of the main methods and models that support the operation of distributed service systems. It is established that the core of consistency in distributed systems lies in the operation of consensus algorithms, which ensure their stable performance and serve as the key mechanism guaranteeing data consistency across nodes, even in the event of failures. Ensuring the reliability of distributed service systems is a complex task that requires a comprehensive approach, considering various sources of failures, including hardware, software, and network-related issues. A particular challenge is posed by the high level of dynamism, scalability, and heterogeneity of modern environments, in which microservice

applications interact over unstable network paths and exhibit variable behavior depending on the load. This necessitates the implementation of intelligent mechanisms for monitoring, adaptation, and self-healing.

A key element in maintaining the resilience of a distributed system is timely fault detection, achieved through both traditional status control methods (e.g., response time checks and heartbeat mechanisms) and modern statistical and machine learning approaches. Probabilistic methods provide an effective solution to this task, enabling the evaluation of failure probability based on behavioral and network metrics, which improves the accuracy of response mechanisms. Fault tolerance is further enhanced by techniques such as service replication, load balancing, reserved resource allocation, and the use of consensus algorithms to maintain system state consistency. The ability of a system to reconfigure quickly under partial failures is a critical prerequisite for preserving its functional integrity and providing uninterrupted services to end users.

The key advantages and disadvantages of existing methods for ensuring the stable operation of consensus algorithms in distributed systems with dynamically changing network structures are examined. The main unresolved issues in maintaining system consistency and operability under continuous failures and reconfigurations of functional nodes are identified. It is established that under conditions of unstable and unpredictable network delays, the effectiveness of failure detection critically depends on the rational selection of the threshold value for the message timeout from nodes. A low timeout value allows the system to respond quickly to potential failures but increases the risk of false positives, which can lead to unnecessary reconfigurations, data inconsistency, and reduced system performance. Therefore, the development of new methods and models for predictive adaptive failure detection, which take into account current network environment characteristics, delay dynamics, and node behavior within a cluster, is a relevant task.

Chapter two, **"Development of adaptive methods for improving the efficiency of consensus algorithms in distributed service systems"** focuses on solving the problem of enhancing the performance of consensus algorithms, particularly under

conditions of unstable network connectivity in distributed systems, through the development of new methods and algorithms. The research demonstrates that network instability—such as delays, packet loss, and connection interruptions—has a significant impact on data consistency and system efficiency. False positives triggered by short-term network failures frequently result in erroneous leader re-elections, increased resource consumption, and a general decline in system performance.

As a result, the method of failure detection in the operation of distributed service systems has been further developed. Unlike existing approaches, it analyzes the operational state of the system and minimizes the number of erroneous failure decisions caused by short-term network disruptions through the integration of a probabilistic Bayesian approach. The use of Bayes' theorem for analyzing the presence of failures enables the system to make more reasoned, probabilistic decisions regarding node failures. By dynamically updating the probabilities of failures, the system can reduce the number of false positives, thereby increasing the reliability of the consensus algorithm and ensuring stable operation of distributed service systems.

An important role in the stability of system functioning and the operation of consensus algorithms is played by the configuration of the node’s response timeout and the heartbeat message interval. Too short a waiting time increases sensitivity to short-term network delay fluctuations and raises the number of cluster reconfigurations, while overly long timeouts can delay responses to actual failures. To minimize the number of reconfigurations, the method of adaptive failure threshold determination has been improved by forecasting the expected time of receiving messages from a potentially non-operational node using the SARIMA machine learning model. The essence of this method lies in time-based prediction of network response delays and in determining the failure detection threshold. The basis of the prediction is statistical analysis of the data collected during system operation. Each node in the network regularly collects and analyzes information about delays related to its connections with other nodes. Using historical data and the SARIMA machine learning model, nodes calculate the expected delay through time analysis, which allows them to predict

potential failures. Having a set of predicted delay values, where each value corresponds to a specific node, the node can make a decision regarding its failure by adjusting the timeout threshold.

To accelerate the timeout threshold adjustment process, an autonomous adaptive method for improving the performance of distributed service systems is proposed for the first time. Based on knowledge of the leader-oriented structure of the consensus algorithm, it allows dynamic adjustment of delay parameters between cluster nodes, selection of a leader in the cluster, and ensures maximum stability and performance of the system as a whole. A key feature of the method is the use of network delay as the main criterion for determining node priority for the leader role. Each node analyzes the delay time to other nodes in the majority (quorum), evaluates its potential suitability for the leader role, and, depending on this, adjusts its heartbeat message timeout. Thus, nodes with poorer network characteristics artificially delay the initiation of elections, yielding to nodes with better connection conditions. Additionally, a stochastic component is integrated into the timeout calculation, the range of which is set administratively and scaled according to current network conditions. This reduces the probability of simultaneous timeout triggers across multiple nodes, which is one of the key issues in basic implementations of algorithms such as Raft.

Chapter three, **"Modeling and performance analysis of methods for improving the efficiency of distributed systems,"** presents an evaluation of the efficiency of a distributed system functioning under dynamic environments and unpredictable network delays, based on a failure detection method using a probabilistic Bayesian approach. The calculation of the posterior probability of node failure was performed. The results of the calculations demonstrate that even in the presence of certain signs of potential failure, such as missed “heartbeat” messages or increased delay, the method does not tend to make premature conclusions but instead enables balanced decision-making based on a combination of factors. This approach significantly reduces the number of false positives by approximately 4–5%. This improvement is achieved

through more flexible, cumulative event analysis and an adaptive threshold that considers the node's behavioral history and additional failure indicators.

To assess the effectiveness of failure detection in a distributed service system operating on the basis of the adaptive failure threshold determination method using the SARIMA machine learning model, a simulation model of an IoT cluster structure was developed as part of the distributed service system. As a result of the model's operation, a dependency was obtained, the analysis of which shows that when predicting response time durations using the SARIMA model, it is possible to dynamically adjust the failure threshold and clearly distinguish between system peak activity moments and actual service failures. This makes it possible to better account for load changes and reduces the number of false positives. After conducting simulation using the adaptive threshold determination method and the SARIMA model, a 20% increase in failure detection accuracy was achieved, ensuring an adaptive and effective system response to unstable network operation conditions.

Simulation modeling of the distributed service system cluster to identify a potential leader using the proposed autonomous adaptive method confirmed its effectiveness in a dynamic network environment. The modeling results demonstrated the ability of the timeout determination algorithm to accurately and flexibly identify the node with the most favorable conditions for performing the leader’s role, taking into account the actual network characteristics. The calculated timeout values showed the approach’s adaptability to current conditions: minimizing response time to the active leader’s failure while maintaining a sufficient buffer to avoid false triggers under delay fluctuations. For full validation of the method, further modeling and practical implementation under various scenarios and operating conditions will be carried out in the next chapter.

In Chapter four, **"Practical implementation and operation of the developed methods for improving the functioning of distributed service systems,"** the integration of the autonomous adaptive method for improving the efficiency of distributed service systems into the simulation model of the cluster made it possible to

achieve a threefold reduction in the probability of premature system reconfiguration under standard operating conditions. At the same time, during peak load periods, the implementation of this approach provides an additional increase of 5.6% in the probability of reconfiguration, which indicates a timely system response to dynamic changes in the connection structure, emphasizes the autonomy and adaptability of the method, and, as a result, the ability to ensure the consistency of cluster operation regardless of its operating conditions.

A hardware-software complex with an autonomous adaptive control approach has been developed to improve the efficiency of the distributed system's operation, which will allow for practical validation of the effectiveness of the proposed methods and algorithms, involving not only software components but also a set of real network equipment. As a result of the operation of the developed hardware-software complex, the necessity of an effective adaptive approach to adjusting the delay parameter between cluster nodes is confirmed, which affects the leader selection in the cluster and reduces the probability of cluster reconfiguration and, consequently, improves the quality of service delivery.

Thanks to the implementation of the autonomous adaptive method in the operation of the hardware-software complex, it was possible to reduce the standard deviation of delay by 44%, which indicates increased stability and predictability of the system's operation. As a result, a 4.5% reduction in the probability of cluster reconfiguration was achieved, which will allow processing requests during peak load moments without the need to re-elect new leaders. The effective integration and application of failure detection methods and adaptive threshold determination for failures through the prediction of message waiting times from potentially failed nodes using the SARIMA machine learning model enabled an increase in the system's message processing speed by 19.5%, indicating the adaptive system's ability to utilize resources more efficiently without sacrificing stability.

The conclusions of the dissertation include generalized research results and recommendations regarding their practical application. In particular, the developed

scientific and applied solutions concerning the integration of the probabilistic approach to failure detection, adaptive determination of the triggering threshold based on the machine learning model (SARIMA), and the autonomous adaptive method of failure detection can be used by research institutions, developers of distributed service systems, and cloud solution providers to increase the reliability, performance, and stability of cluster operation in environments with highly dynamic network conditions.

The scientific and practical results of the conducted research have been used in the educational process of the Department of Information and Communication Technologies at Lviv Polytechnic National University for the modernization of the lecture course for students of the second (master's) level of higher education in specialty 172 "Electronic Communications and Radio Engineering" of the Educational and Scientific Program "Telecommunications and Radio Engineering" in the discipline "Distributed service systems and cloud technologies", as well as laboratory works in the discipline "Distributed information and communication networks" for students of the first (bachelor's) level of higher education of the educational and professional program "Information and communication systems".

The main results of the dissertation work have been used and implemented in order to improve the efficiency of distributed service systems when integrated into modern information and communication service-oriented networks, in particular by LLC "Vitertek" and LLC VTF "Kontekh", as confirmed by the acts of implementation.

**Keywords:** Distributed service systems, consensus algorithms, failure detection, adaptive control, probabilistic analysis, system stability, machine learning, leader-oriented algorithms, simulation modeling, network delays, autonomous systems.

The list of author’s publications:

**Proceedings where basic scientific results of thesis were published**

1. Zhuravel S., Klymash M., Shpur O., Mrak V. “Reducing the impact of unstable connections among nodes of wireless IIoT clusters using machine learning

methods", Lecture Notes in Electrical Engineering. Vol. 1198: Digital ecosystems: interconnecting advanced networks with AI applications. P. 144–159, 2024. doi: 10.1007/978-3-031-61221-3_8

2. S. Zhuravel, O. Shpur, and M. Klymash, "Adaptive consensus algorithms: Designing for durability against unstable network connections," International Journal of Computing, vol. 23, no. 4, pp. 574–582, 2024. doi: 10.47839/ijc.23.4.3756.

3. N. Peleh, S. Zhuravel, O. Shpur, and O. Rybytska, "Structured and unstructured log analysis as a method to detect DDoS attacks in SDN networks," Internet of Things (IoT) and Engineering Applications, vol. 6, no. 1, pp. 1–9, 2021. doi: 10.23977/iotea.2021.060101.

4. S. Zhuravel, "Development of network simulation model for evaluating the efficiency of distributed consensus taking into account the instability of network connections," Infocommunication Technologies and Electronic Engineering, vol. 4, no. 1, pp. 10–19, 2024. doi: 10.23939/ictee2024.01.010.

5. S. Zhuravel, O. Shpur, and Y. Pyrih, "Method of achieving consensus in distributed service systems," Infocommunication Technologies and Electronic Engineering, vol. 2, no. 2, pp. 58–66, 2022. doi: 10.23939/ictee2022.02.058.

6. Zhuravel S., Dumych S., Shpur O. "Research on Data Collection and Processing Methods in Distributed Information Systems." Infocommunication Technologies and Electronic Engineering, vol. 1, no. 1, pp. 20–38, 2021. doi: 10.23939/ictee2021.01.020

**Proceedings that certify an approvement of thesis materials**

7. S. Zhuravel, "Adaptive probabilistic methods for boosting the reliability of consensus algorithms in distributed systems," in Proc. 2024 IEEE 17th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 2024, pp. 145–149. doi: 10.1109/TCSET64720.2024.10755585.

8. M. Klymash, O. Shpur, O. Lavriv, and S. Zhuravel, “Achieving consistency and consensus of distributed infocommunication systems,” in Proc. 2022 IEEE 16th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 2022, pp. 386–389. doi: 10.1109/TCSET55632.2022.9767019.

9. M. Klymash, S. Zhuravel, O. Shpur “Machine learning approaches to stabilize wireless IIoT clusters facing network instabilities” Наукоємні технології в інфокомунікаціях: збірник наукових праць Міжнародної науково-практичної конференції HICT'2023, Харків - Кам'янець-Подільський, Україна, 2023, с. 90–92.

## СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

*Публікації, у яких опубліковані основні наукові результати дисертації:*

1. Zhuravel S., Klymash M., Shpur O., Mrak V. “Reducing the impact of unstable connections among nodes of wireless IIoT clusters using machine learning methods”, *Lecture Notes in Electrical Engineering. Vol. 1198: Digital ecosystems: interconnecting advanced networks with AI applications*. P. 144–159, 2024. doi: 10.1007/978-3-031-61221-3_8

*Особистий внесок здобувача: запропонував удосконалення методу адаптивного визначення порогу відмов, що дозволяє зменшити вплив нестабільних з'єднань між вузлами бездротових кластерів IIoT за допомогою методів машинного навчання*

2. S. Zhuravel, O. Shpur, and M. Klymash, “Adaptive consensus algorithms: Designing for durability against unstable network connections,” *International Journal of Computing*, vol. 23, no. 4, pp. 574–582, 2024. doi: 10.47839/ijc.23.4.3756.

*Особистий внесок здобувача: розроблення імітаційної моделі кластеру розподіленої сервісної системи для оцінки ефективності визначення потенційного лідера*

3. N. Peleh, S. Zhuravel, O. Shpur, and O. Rybytska, “Structured and unstructured log analysis as a method to detect DDoS attacks in SDN networks,” *Internet of Things (IoT) and Engineering Applications*, vol. 6, no. 1, pp. 1–9, 2021. doi: 10.23977/iotea.2021.060101.

*Особистий внесок здобувача: моделювання та оцінка ефективності функціонування хмарних систем в умовах обслуговування групових потоків запитів*

4. S. Zhuravel, “Development of network simulation model for evaluating the efficiency of distributed consensus taking into account the instability of network connections,” *Infocommunication Technologies and Electronic Engineering*, vol. 4, no. 1, pp. 10–19, 2024. doi: 10.23939/ictee2024.01.010.

*Особистий внесок здобувача: проведення дослідження та оцінки впливу параметру тайм-ауту на процес реконфігурації кластерів в розподіленій системі*

5. S. Zhuravel, O. Shpur, and Y. Pyrih, “Method of achieving consensus in distributed service systems,” *Infocommunication Technologies and Electronic Engineering*, vol. 2, no. 2, pp. 58–66, 2022. doi: 10.23939/ictee2022.02.058.

*Особистий внесок здобувача: розроблення автомномного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем*

6. Журавель С., Думич С., Шпур О. “Дослідження методів збору та обробки даних в розподілених інформаційних системах” *Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія*. vol. 1, № 1, с. 20–38, 2021. doi: 10.23939/ictee2021.01.020

*Особистий внесок здобувача: проведено порівняльний аналіз алгоритмів збору та обробки даних, їх обмеження щодо адаптивності до змінних мережевих умов.*

*Наукові праці, які засвідчують апробацію матеріалів дисертації:*

7. S. Zhuravel, “Adaptive probabilistic methods for boosting the reliability of consensus algorithms in distributed systems,” in *Proc. 2024 IEEE 17th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2024, pp. 145–149. doi: 10.1109/TCSET64720.2024.10755585.

*Особистий внесок здобувача: проведення імітаційного моделювання автономного адаптивного методу для дослідження ефективності функціонування розподілених сервісних систем*

8. M. Klymash, O. Shpur, O. Lavriv, and S. Zhuravel, “Achieving consistency and consensus of distributed infocommunication systems,” in *Proc. 2022 IEEE 16th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2022, pp. 386–389. doi: 10.1109/TCSET55632.2022.9767019.

*Особистий внесок здобувача: проведення дослідження впливу нестабільних з'єднань між вузлами бездротових кластерів IIoT*

9. M. Klymash, S. Zhuravel, O. Shpur “Machine learning approaches to stabilize wireless IIoT clusters facing network instabilities” Наукоємні технології в інфокомунікаціях: збірник наукових праць Міжнародної науково-практичної конференції НІСТ'2023, Харків - Кам'янець-Подільський, Україна, 2023, с. 90–92.

*Особистий внесок здобувача: узагальнення та опрацювання результатів експериментальних досліджень, проведення порівняльного аналізу отриманих значень, апробація матеріалів на конференції.*

**ЗМІСТ**

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ ..... 21
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ..... 26
ВСТУП ..... 27
РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ТА АЛГОРИТМІВ КОНСЕНСУСУ РОЗПОДІЛЕНИХ СЕРВІСНИХ СИСТЕМ ..... 34
1.1. Принципи та проблематика консенсусу у розподілених сервісних системах ..... 34
1.2. Аналіз методів покращення функціонування розподілених сервісних систем ..... 42
1.3. Забезпечення надійної та узгодженої роботи алгоритмів розподілених сервісних систем при динамічності зміни умов функціонування мережі ..... 52
1.3.1. Стабільність структурних параметрів як критерій ефективного функціонування алгоритмів консенсусу ..... 54
1.3.2. Вплив помилкових спрацювань на ефективність функціонування розподілених сервісних систем ..... 56
1.4. Висновки до першого розділу ..... 58
РОЗДІЛ 2. РОЗРОБЛЕННЯ АДАПТИВНИХ МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ КОНСЕНСУСУ В РОЗПОДІЛЕНИХ СЕРВІСНИХ СИСТЕМАХ ..... 60
2.1. Метод виявлення відмов у роботі розподілених сервісних систем на основі ймовірнісного байєсівського підходу. ..... 61
2.2. Зменшення впливу нестабільності мережевих з’єднань на функціонування алгоритмів консистентності ..... 69
2.2.1 Адаптивне визначення порогу відмов за рахунок впровадження моделі машинного навчання SARIMA ..... 73
2.3. Розроблення автономного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем ..... 76
2.4. Висновки до другого розділу ..... 90

РОЗДІЛ 3. МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНИХ СИСТЕМ .... 92
3.1. Розрахунок ймовірності виявлення відмови вузла із використанням ймовірнісного байєсівського підходу ..... 92
3.2. Дослідження ефективності методу адаптивного визначення порогу відмов у мережах IoT з використанням моделі машинного навчання SARIMA на основі розробленої імітаційної моделі ..... 100
3.3. Розроблення імітаційної моделі кластеру розподіленої сервісної системи для оцінки ефективності визначення потенційного лідера з використанням автономного адаптивного методому ..... 109
3.3.1. Особливості реалізації імітаційної моделі кластеру для оцінювання працездатності алгоритмів консенсусу за умов мережевих флуктуацій.... 109
3.3.2. Моделювання стану кластерної системи в режимі реального часу функціонування мережі..... 119
3.4. Висновки до третього розділу..... 122
РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ФУНКЦІОНУВАННЯ РОЗРОБЛЕНИХ МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНИХ СЕРВІСНИХ СИСТЕМ ..... 124
4.1 Дослідження ефективності застосування автомномного адаптивного методу та його вплив на функціонування розподіленої сервісної системи .. 124
4.1.1. Статистична оцінка мережевих характеристик для верифікації параметрів імітаційної моделі кластеру РСС..... 125
4.1.2. Оцінка впливу параметру тайм-ауту на процес реконфігурації кластерів в розподіленій системі..... 131
4.1.3. Дослідження ефективності функціонування розподіленої сервісної системи в умовах змінної працездатності мережі з використанням автономного адаптивного методу ..... 141

4.2 Розроблення програмно-апаратного комплексу на основі автономного адаптивного підходу для управління функціонуванням розподіленої системи . ..... 155
4.2.1. Дослідження ефективності використання запропонованих рішень та їх вплив на якість обслуговування ..... 157
4.3 Висновки до четвертого розділу ..... 160
ОСНОВНІ РЕЗУЛЬТАТИ ТА ВИСНОВКИ ..... 162
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ ..... 165
ДОДАТОК А. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНИХ ДОСЛІДЖЕНЬ ..... 182
ДОДАТОК Б. СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ ТА ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ ..... 185

## ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API — Application Programming Interface, інтерфейс прикладного програмування

CPU — Central Processing Unit, центральний процесор

DB — Database, база даних

IoT — Internet of Things, інтернет речей

LAN — Local Area Network, локальна мережа

QoS — Quality of Service, якість обслуговування

SARIMA — Seasonal AutoRegressive Integrated Moving Average, сезонна авторегресійна інтегрована модель ковзного середнього

UDP — User Datagram Protocol, протокол користувацьких дейтаграм

ICMP — Internet Control Message Protocol, протокол керування повідомленнями Інтернету

FUSE — Failure Understanding and State Evaluation, інтерфейс виявлення та оцінки збоїв

ID — Identifier, ідентифікатор

ML — Machine Learning, машинне навчання

ARIMA — AutoRegressive Integrated Moving Average, авторегресійна інтегрована модель ковзного середнього

P99 — 99-й процентиль, показник розподілу затримок

РСС – розподілена сервісна система

# ВСТУП

**Актуальність теми.**

В умовах стрімкого розвитку цифрової інфраструктури та масштабування обчислювальних ресурсів розподілені сервісні системи (РСС) стали критично важливою основою для побудови високонавантажених, масштабованих і відмовостійких інформаційно-комунікаційних сервісів. Хмарні обчислення, системи IoT, Edge- і Fog-обчислення, мікросервісні архітектури — усі ці парадигми ґрунтуються на використанні розподілених сервісних компонентів, функціонування яких повинно залишатися стабільним навіть за умов часткових збоїв або динамічних змін навантаження. Ефективне функціонування РСС вимагає вирішення низки складних задач: реалізація транзакції, виявлення та локалізація відмов, координація вузлів, гарантування унікальності значень, забезпечення консистентності даних та визначення оптимального часу відгуку вузлів. Традиційні алгоритми, такі як Paxos і Raft мають обмеження щодо адаптованості до змінних мережевих умов, що робить їх менш придатними для використання у високонавантажених та масштабованих системах. Raft досягнув простоти завдяки чіткій лідер-орієнтованій структурі та простому механізму виявлення збоїв, який базується на тайм-аутах. Лідер у кластері централізує прийняття рішень, знижуючи необхідність складних координаційних механізмів, що спрощує реалізацію та обслуговування системи. У зв’язку з цим постає необхідність розробки моделей та методів, здатних не лише реагувати на поточний стан системи, а й передбачати її поведінку за умов невизначеності.

Особливої актуальності набуває впровадження інтелектуальних методів аналізу стану системи, які дозволяють не лише реагувати на поточні події, а й здійснювати прогнозування подальшої поведінки. Такі підходи включають використання моделей часових рядів, ймовірнісних механізмів оцінок, а також алгоритмів адаптивного керування, здатних автономно приймати рішення щодо виявлення відмов, відновлення сервісів або змін у розподілі ролей між вузлами.

Зважаючи на постійне зростання обсягів даних, кількості взаємодіючих компонентів та критичну залежність бізнес-процесів від безперебійної роботи розподілених систем, підвищення їх ефективності, оптимізації та якості функціонування постає як надзвичайно актуальне завдання. У своїх роботах над його вирішенням працювали представники наукових шкіл професора Поповського В.В., професора Згуровського М.З., професора Глоби Л.С., професора Климаша М.М.. Активно долучаються до вирішення проблем узгодженості систем Золотухін О.В., Демидов І.В., Шпур О.М. Серед іноземних дослідників слід відзначити роботи F. Munoz, C. Berger, Hans P. Reiser, W.Lu, M.Raynal та інших.

Основна частина робіт згаданих авторів спрямована на покращення узгодженості роботи розподілених систем, які орієнтовані на надання композитних сервісів шляхом удосконаленого управління мережними ресурсами та їх планування. Проте недостатньо уваги приділено адаптивному та прогнозованому виявленню відмов з урахуванням поточних умов функціонування системи. Для досягнення цієї мети слід розв’язати протиріччя між мінімізацією значення часу очікування вузлом на відоповідь, який дозволяє системі оперативно реагувати на потенційні збої та сприяє швидкому переобранню лідера, і підвищенням ймовірності хибнопозитивного спрацювання механізму виявлення відмов, що спричиняє зайві процедури реконфігурації, порушення узгодженості даних та загальне зниження продуктивності системи. Таким чином, необхідність балансування між швидкістю реагування на відмови та точністю функціонування механізмів виявлення збоїв потребує розв’язання науково-практичного завдання – розроблення нових методів та моделей підвищення ефективності функціонування РСС з прогнозованим адаптивним виявленням відмов, які враховуватимуть поточні характеристики мережевого середовища, динаміку затримок і поведінку вузлів у кластері та визначати їх порогові значення.

**Зв'язок роботи з науковими програмами, планами, темами**. Тематика дисертаційного дослідження виконувалась у відповідності до наукового напряму кафедри інформаційно-комунікаційних технологій Національного університету "Львівська політехніка".

**Мета і завдання дослідження.** Метою дисертаційної роботи є підвищення ефективності функціонування розподілених сервісних систем шляхом розроблення методів та моделей адаптивного покращення функціонування алгоритмів консенсусу через удосконалення механізмів виявлення відмов, зокрема шляхом оптимізації таймаутів як основного інструменту детекції збоїв.

Досягнення поставленої мети здійснюється розв'язанням таких завдань:

1. Аналіз існуючих методів та моделей функціонування розподілених сервісних систем.
2. Розроблення методу виявлення відмов у роботі розподілених сервісних систем на основі ймовірнісного байєсівського підходу.
3. Розроблення методу адаптивного визначення порогу відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделей машинного навчання
4. Розроблення автономного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем, який дозволяє покращити можливість вузлів відрізняти тимчасові проблеми навантаження мережі та вузлів від повної їх відмови, що досягається за рахунок обміну даними між вузлами про затримки в кластері.
5. Розроблення імітаційної моделі кластеру розподіленої сервісної системи для оцінки ефективності визначення потенційного лідера.
6. Дослідження ефективності функціонування розподіленої сервісної системи в умовах змінної працездатності мережі.
7. Дослідження практичних аспектів та викликів реалізації розроблених методів покращення функціонування розподілених сервісних систем.

8. Дослідження та оцінка ефективності функціонування консистентної розподіленої сервісної системи в умовах зміни працездатності мережі із використанням запропонованих рішень та їх вплив на якість обслуговування.

**Об'єктом дослідження** є процес функціонування розподілених сервісних систем в умовах змінних мережевих параметрів.

**Предметом дослідження** є методи та моделі підвищення ефективності функціонування алгоритмів консенсусу, які забезпечують узгодженість та надійність розподілених сервісних систем, зокрема механізми виявлення відмов та адаптивного управління параметрами мережевої взаємодії.

**Методи дослідження**. У дисертації використано методи теорії графів, математичного моделювання, машинного навчання, ймовірнісного аналізу, алгоритмічного аналізу, імітаційного моделювання та методи оцінки ефективності алгоритмів консенсусу.

**Наукова новизна отриманих результатів**

1. Набув подальшого розвитку метод виявлення відмов в роботі розподілених сервісних систем, який на відміну від існуючих аналізує стан функціонування системи та мінімізує кількість помилкових рішень про відмови, шляхом інтеграції ймовірнісного байєсівського підходу, що дасть змогу забезпечувати стабільне функціонування розподілених сервісних систем.

2. Удосконалено метод адаптивного визначення порогу відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA, що дозволяє зменшити вплив нестабільності мережевих з’єднань на функціонування алгоритму консистентності, і як наслідок, на працездатність роботи розподіленої сервісної системи

3. Вперше запропоновано автономний адаптивний метод підвищення продуктивності розподілених сервісних систем, який на основі знань про лідер-орієнтовану структуру алгоритму консенсусу, дозволяє динамічно коригувати

параметри затримки між вузлами кластеру, обирати лідера в кластері, та забезпечити максимальну стабільність і продуктивність роботи системи вцілому.

**Практичне значення одержаних результатів** полягає в тому, що:

1. На основі методу виявлення відмов з використанням ймовірнісного байєсівського підходу розроблено алгоритм розрахунку апостеріорної ймовірності відбою вузла, який дозволив зменшити кількість хибнопозитивних спрацьовувань на 4,5% за рахунок більш гнучкого визначення порогу відмови та оцінки стану функціонування системи.

2. Удосконалено метод адаптивного визначення порогу відмов, що дало змогу підвищити точність виявлення збоїв на 20% за рахунок впровадження моделі машинного навчання SARIMA, яка забезпечує прогнозування відмов на основі аналізу тайма-ауту між вузлами системи та чітко розмежовує пікові моменти функціонування системи та відмови в обслуговуванні.

3. На основі інтеграції автомномного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем у модель кластеру зменшено ймовірність передчасної реконфігурації системи у 3 рази у стандартному режимі функціонування. Водночас у періоди пікового навантаження впровадження даного підходу забезпечує додаткове підвищення ймовірності реконфігурації на 5,6% завдяки своєчасній реакції системи на динамічну зміну структури з’єднань

4. Розроблено алгоритм коригування порогу тайм-ауту, що дало змогу зменшити стандартне відхилення затримки на 44% за рахунок оцінки потенційної придатності до виконання ролі лідера та адаптації значення часу очікування на повідомлення перевірки працездатності (heartbeat).

5. Розроблено програмно-апаратний комплекс на основі автономного адаптивного підходу для управління функціонування розподіленої системи, що забезпечило зменшення ймовірності реконфігурації кластеру на 5%

Наукові та практичні результати виконаних досліджень використані в навчальному процесі кафедри інформаційно-комунікаційних технологій

Національного університету «Львівська політехніка» для модернізації курсу лекцій для здобувачів другого (магістерського) рівня вищої освіти спеціальності 172 Електронні комунікації та радіотехніка ОНП «Телекомунікації та радіотехніка» з дисципліни «Розподілені сервісні системи та Cloud-технології» та лабораторних робіт з дисциплін «Розподілені інфомаційно-комунікаційні мережі» для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Інформаційно-комунікаційні системи».

Основні результати дисертаційної роботи використано і впроваджено з метою підвищення ефективності функціонування розподілених сервісних систем при їх інтеграції у сучасні інформаційно-комунікаційні сервісно-орієнтовані мережі, зокрема ТзОВ «Вітертек» та ТзОВ ВТФ «Контех», що підтверджено актами впровадження.

**Особистий внесок здобувача.** Основні наукові результати дисертаційної роботи отримано автором самостійно. У працях, опублікованих у співавторстві, внесок здобувача є вирішальним, зокрема авторові належать (*нумерація згідно Додатку Б*: у роботах [1, 8, 9] – удосконалення методу адаптивного визначення порогу відмов, що дозволяє зменшити вплив нестабільних з'єднань між вузлами бездротових кластерів IIoT за допомогою методів машинного навчання, [2] – розроблення імітаційної моделі кластеру розподіленої сервісної системи для оцінки ефективності визначення потенційного лідера; [3] – моделювання та оцінка ефективності функціонування хмарних систем в умовах обслуговування групових потоків запитів, [4] – оцінка впливу параметру тайм-ауту на процес реконфігурації кластерів в розподіленій системі; [5, 7] – розроблення автомномного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем; [6] – дослідження методів збору та обробки даних в розподілених інформаційних системах.

**Апробація результатів дисертації.** Основні наукові результати і положення дисертації представлені, доповідались та обговорені на 3-ох міжнародних науково-технічних конференціях: Міжнародна науково-технічна

конференція «Сучасні проблеми радіоелектроніки, телекомунікацій, комп’ютерної інженерії» (м. Львів-Славське 2022, 2024 рр.); Наукоємні технології в інфокомунікаціях НІСТ'2023 (Харків - Кам'янець-Подільський, Україна, 2023 р.). Крім цього, дисертаційна робота у повному обсязі представлена на наукових семінарах кафедри інформаційно-комунікаційних технологій Національного університету «Львівська політехніка».

**Публікації.** За результатами досліджень, які викладені у дисертаційній роботі, опубліковано 9 наукових праць, з них 1 стаття у науковому періодичному виданні інших держав, що входять до міжнародних науково-метричних баз даних, 1 стаття у наукових фахових виданнях України, що входять до міжнародних науково-метричних баз даних, 1 стаття у науковому періодичному виданні інших держав; 3 статті у наукових фахових виданнях України, 3 публікації у збірниках тез наукових конференцій (зокрема 2 – у виданнях, які входять до наукометричних баз даних Scopus та Web of Science).

**Структура та обсяг роботи.** Робота складається з переліку умовних скорочень, вступу, 4 розділів, висновків, списку використаних джерел і 2 додатків. Загальний обсяг роботи складає 186 сторінок друкарського тексту, із них 7 сторінок вступу, 151 сторінок основного тексту, 63 рисунків, 6 таблиць, список використаних джерел із 149 найменувань, 2 додатки на 5 сторінках. Додатки містять обрані акти впровадження результатів дисертаційної роботи, а також список праць автора.

# РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ТА АЛГОРИТМІВ КОНСЕНСУСУ РОЗПОДІЛЕНИХ СЕРВІСНИХ СИСТЕМ

На сьогоднішній день більшість інфокомунікаційних систем є розподіленими. Такі системи, як правило, функціонують в умовах можливих збоїв мережі, а відтак, потребують можливості зберігати працездатність, навіть в умовах відмов частин системи. Однак наявність великої кількості вузлів, як окремо взятих елементів системи, породжує проблему роз-синхронізації її роботи. Крім того існує велика кількість практичних проблем, які необхідно вирішити для працездатності розподілених систем. До таких проблем відносяться лінійні операції порівняння та призначення, атомарні операції для транзакцій, повний порядок операцій в розподіленій системі, можливість реалізацій “замків” для операцій, реалізація координаційних сервісів та сервісів “групи”, можливість гарантування унікальності значення. Всі ці можливості так чи інакше можуть бути зведені до вирішення проблеми консенсусу в розподіленій системі.

## 1.1. Принципи та проблематика консенсусу у розподілених сервісних системах

Загалом досягнення консенсусу в розподіленій системі означає прийняття рішення групою вузлів таким чином, щоб усі вузли погоджувалися з цим рішенням та могли забезпечити його виконання. Консенсус у свою чергу є спорідненим з проблемами лінійності операції та повного порядку у доставленні повідомлень всім членам розподіленої системи, оскільки для досягнення згоди всі вузли мають отримувати однакову інформацію в однаковому порядку. Лінійність операцій забезпечує, що всі учасники мають однакове уявлення про послідовність змін у системі, що критично для консистентності [1-3]. Повний порядок доставки повідомлень гарантує, що всі вузли отримують і обробляють повідомлення в однаковій послідовності, запобігаючи суперечностям [4-6]. Таким чином, консенсус залежить від упорядкованого і синхронізованого обміну

інформацією між учасниками системи, тобто проблему консенсусу можна вирішити за допомогою лінійності операцій, оскільки всі вузли в системі будуть знаходитись в тому ж стані що й інші якщо всі операції які були виконані на вузлі були однаковими та в тому ж порядку, це ж відповідно працює і в зворотному порядку, тобо якщо уявити що система в кожен момент часу є консистентною це ж і означає що всі операції виконуються в тому ж порядку [1-5].

Ідея лінійності операцій також відома як атомарна узгодженість або зовнішня узгодженість полягає в тому, щоб система виглядала так, ніби є лише одна копія даних, і всі операції над ними є атомарними. Завдяки цій особливості, при фунціонуванні системи можуть створюватися кілька реплік (копій) одного функціонального елементу, а системі не потрібно турбуватися про них. У такого роду системі як тільки один клієнт успішно завершує запис, усі клієнти, які читають дані, повинні мати можливість побачити щойно записане значення. Зберігати ілюзію єдиної копії даних означає гарантувати, що прочитане значення є найновішим, і не походить із застарілого кешу чи репліки. Іншими словами, лінійність є гарантією новизни [3].

Для наочності, на рисунку 1.1. представлено виконання завдань у системі, яка комбінує лінійний та нелінійний підхід до обробки операцій. Лінійна частина (зліва) демонструє послідовне виконання операцій, що забезпечує атомарність і єдину узгоджену історію змін. Нелінійна частина (справа) ілюструє можливість одночасного виконання кількох операцій, де порядок їх завершення може відрізнятися залежно від взаємозв’язків і доступності ресурсів. Іншими словами, для клієнта розподіленої системи повинна зберігатися ілюзія, що всі операції виконуються у строгому лінійному порядку, навіть якщо насправді вони можуть оброблятися паралельно або в іншій послідовності, як це демонструється у лінійній підсистемі на рисунку 1.1 (зліва).

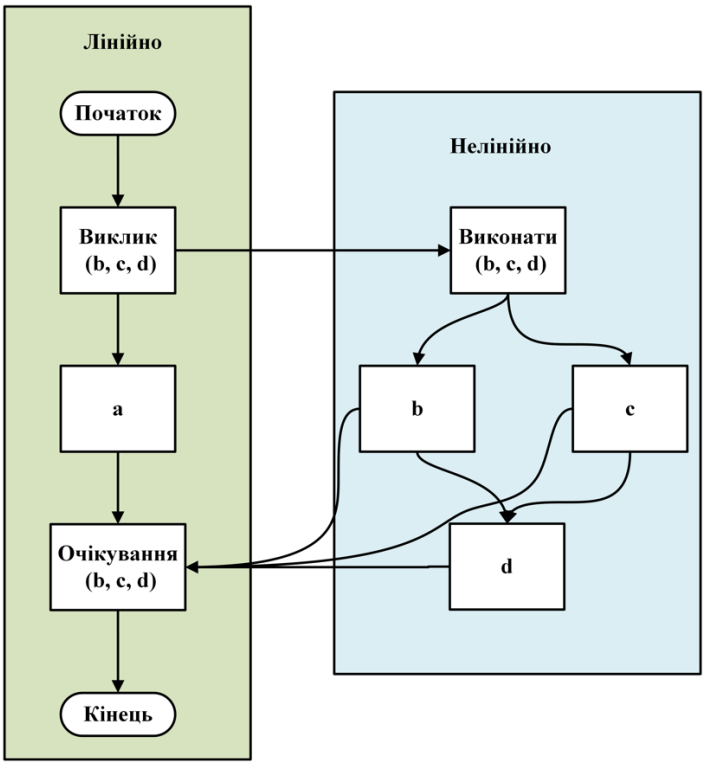



Рис. 1.1. Приклад виконання завдань в підсистемі з лінійним виконанням операцій (зліва) та не лінійним (зправа)

Перехід до трансляції повного порядку природно доповнює лінійність, оскільки забезпечує узгодженість обміну повідомленнями між вузлами. Принцип трансляції повного порядку гарантує, що всі вузли отримують повідомлення у встановленій послідовності, задовольняючи ключові властивості узгодженості й синхронізації. Він вимагає, щоб завжди були задоволені дві властивості:

- Надійна доставка. Повідомлення не втрачаються, тобто якщо повідомлення доставляється одному вузлу, воно доставляється на всі вузли.
- Повний порядок доставки для всіх вузлів. Повідомлення доставляються кожному вузлу в тому самому порядку.

Алгоритм повинен гарантувати, що надійність і властивості впорядкування завжди задовольняються, навіть якщо вузол або мережа є несправними. Звичайно, повідомлення не будуть доставлені, коли мережа перервана, але

алгоритм може продовжувати повторні спроби, щоб повідомлення пройшли, коли мережа врешті-решт буде відновлена (і тоді вони все одно повинні бути доставлені в правильному порядку). Рисунок 1.2 ілюструє цей процес: наприклад, якщо вузол 1 виконує обчислення автомату станів (state machine) і має передати результати всім іншим вузлам (повідомлення 1 та повідомлення 2), алгоритм повинен забезпечити, щоб ці повідомлення були оброблені в тому ж порядку, що дозволить точно відтворити стан вузла 1 на інших вузлах [7-9].

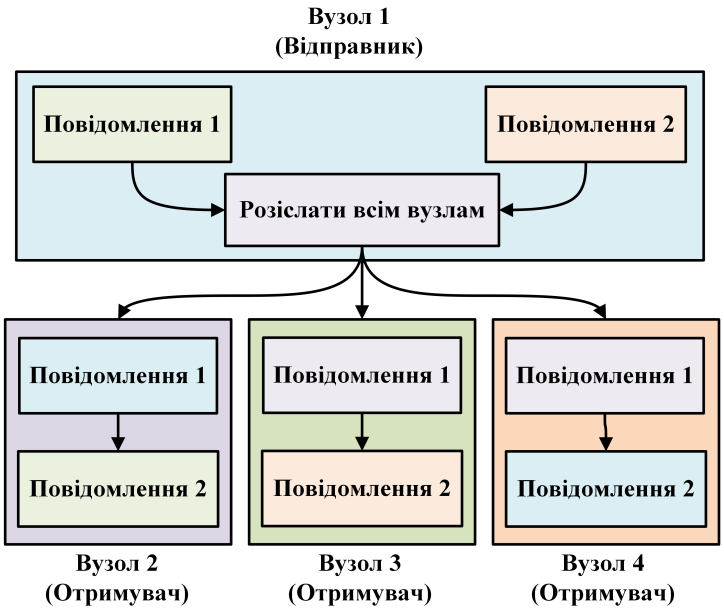



Рис. 1.2. Принцип повного порядку повідомлень

Проблему консенсусу можна вирішити шляхом забезпечення лінійності операцій або впровадження принципу повної передачі повідомлень. У випадку, коли всі вузли у кластері виконують дії в однаковій послідовності (лінійність операцій) або отримують інформацію про виконані операції в строго визначеному порядку, що дозволяє їм самостійно відтворити ці дії, можна гарантувати консистентність системи. Це означає, що стан кожного вузла в системі буде ідентичним, а клієнт, незалежно від того, до якого вузла він звернеться, отримає однакові дані про стан системи [10-12].

Зазвичай консенсус формалізується так: один або кілька вузлів можуть запропонувати значення, і алгоритм консенсусу визначає єдине рішення, яке

буде прийняте всіма учасниками системи. Це значення може мати різний зміст залежно від контексту:

Якщо консенсус використовується для забезпечення узгодженого зберігання даних у розподіленій системі, значенням може бути набір даних або конфігураційні параметри, які потрібно зберегти в усіх вузлах у консистентному стані.

Якщо кластер виконує розподілений автомат станів, значенням може бути команда (операція), яку необхідно узгодити, щоб забезпечити однакову послідовність змін стану на всіх вузлах.

Наприклад, у системі бронювання місць на літак, коли кілька клієнтів одночасно намагаються купити останнє місце, вузли можуть пропонувати унікальний ідентифікатор клієнта як значення, що потрібно узгодити. Якщо система працює у вигляді розподіленого автомату станів, кожен вузол може запропонувати команду “резервації” (передавши параметрами: номір місця та ідентифікатор клієнта), і алгоритм консенсусу визначить єдиний набір команд, який буде виконано в однаковому порядку на всіх вузлах [3, 13-14].

Таким чином, консенсус забезпечує узгодженість даних або порядок виконання операцій у розподіленій системі, незалежно від можливих збоїв, мережевих затримок чи розбіжностей у пропозиціях різних вузлів.

У цьому формулюванні алгоритм консенсусу повинен задовольняти наступним властивостям [4]:

- Узгодженість - немає двох вузлів які б вирішили по-різному.
- Цілісність - жоден вузол не приймає рішення двічі.
- Вірність - якщо вузол визначає значення, то саме це значення, яке обрав вузол, є запропонованим значенням (існує в основному для того, щоб виключити тривіальні рішення. Наприклад, можна створити алгоритм, який завжди вирішує нуль, незалежно від того, що було запропоновано).
- Завершеність - кожен вузол, який є в працездатному стані, в кінцевому підсумку пропонує певне значення.

Найбільш відомі алгоритми консенсусу включають Viewstamped Replication (VSR), Paxos [15-19], Raft [20-21] та Zab. Серед них Paxos і його наступник Raft заклали основу для подальшого розвитку консенсусних алгоритмів. Незважаючи на спільну мету, їх структура кардинально відрізняється [21-23]. Розглянемо їх детальніше.

Raft обирає лідера в кластері, який диктує всім іншим які команди/повідомлення виконати/зберегти а також надсилає повідомлення працездатності (heardbeat) всім учасникам кластеру для підтвердження своєї присутності та працездатності (рисунок 1.3). Сам процес вибору лідера проходить шляхом голосування, який пропонується одним з учасників кластеру. Як тільки вузол кластеру перестане отримувати повідомлення працездатності (heardbeat) від лідера – лідер переобирається. Новим лідером стає кандидат, який отримав голоси більшості кластера.

Алгоритми, в яких один із вузлів обирається лідером і відповідає за визначення порядку повідомлень для всіх інших учасників, належать до класу алгоритмів, що працюють на основі лідера. У таких підходах лідер виступає центральним координатором, забезпечуючи єдиний, узгоджений порядок виконання операцій і доставки повідомлень. Цей підхід спрощує синхронізацію в розподіленій системі, оскільки всі вузли дотримуються вказівок лідера, що мінімізує суперечності та гарантує коректність порядку дій [20,24-26].

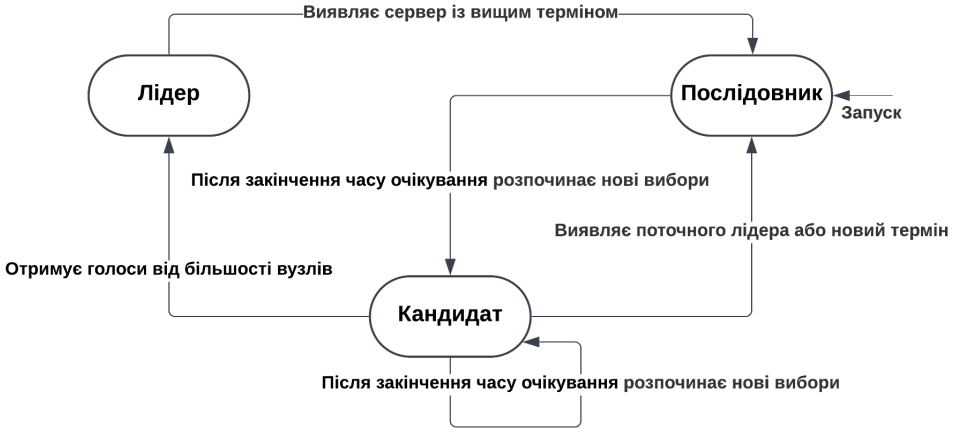



Рис. 1.3. Процес вибору лідера в кластерів відповідно до роботи алгоритму Raft [7]

Paxos, у свою чергу, працює без обрання лідера (так званий “безлідерний” алгоритм). При його функціонуванні не вимагається наявність лідера, проте він є значно складнішим в реалізації. Функціонування алгоритму Paxos виглядає наступним чином: вузол Paxos може виконувати будь-яку або всі з трьох ролей: того хто пропонує значення, того хто підтверджує запропоноване значення (приймач) та учня (рисунок 1.4).

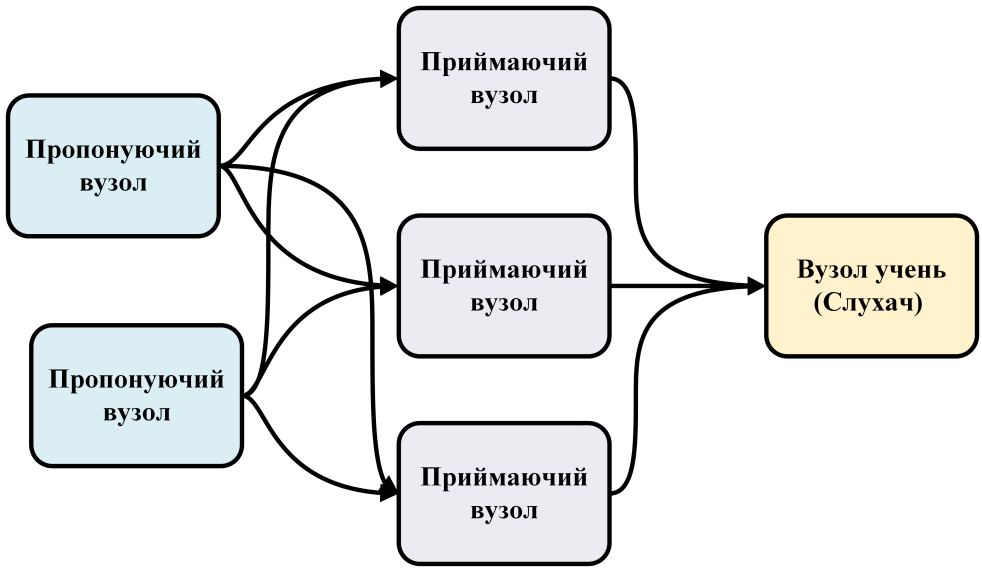



Рис. 1.4. Принцип роботи алгоритму Paxos [15]

Один з вузлів пропонує значення, яке він хоче узгодити з кластером (щоб досягнути консенсусу системи) [7,17-18,20,27]. Для цього вузол надсилає пропозицію, всім вузлам які виступають в ролі “приймачів”, які вирішують, чи приймати це значення чи ні. Кожен приймач самостійно обирає значення, а також отримати декілька пропозицій, кожну від різних вузлів. В результаті послідовного перебору, надсилає своє рішення учням, які визначають, чи було досягнуто консенсусу в виборі значення чи ні. Щоб значення було прийняте у Paxos, більшість приймачів повинні вибрати те саме значення.

Paxos, як і Raft, є складними алгоритмами з великою кількістю специфічних ситуацій та технік з їх вирішення. Розглянемо найпростішу ситуацію, коли всі приймачі погоджуються із вибором значення (рисунок 1.5).

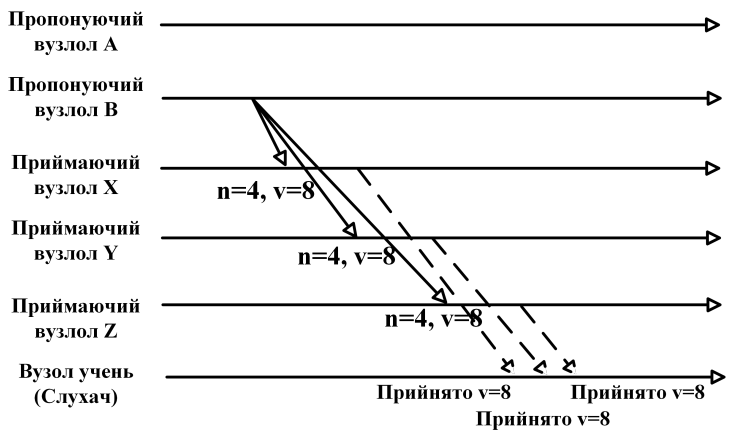



Рис. 1.5. Раунд алгоритму Paxos в якому всі вузли погоджуються з запропонованим значенням.

Процес, за допомогою якого вузли голосують за прийняття рішення, є свого роду синхронною реплікацією. Проведемо паралель з базами даних які часто налаштовані на використання асинхронної реплікації. У цій конфігурації деякі дані які вважались надійно збереженими потенційно можуть бути втрачені під час відмови одного з вузлів системи, але багато інженерів вирішують прийняти цей ризик заради кращої продуктивності системи.

Для досягнення консенсусу системи і як наслідок підвищення продуктивності її функціонування, завжди потрібна більшість. Це означає, що потрібно мінімум три вузли, щоб витримати один збій (решта - два з трьох, формують більшість), або мінімум п’ять вузлів, щоб витримати дві відмови (решта - три з п’яти, утворюють більшість). Якщо збій мережі відрізає деякі вузли від більшості, лише більша частина вузлів разом може досягти прогресу, а решта, відрізана збоями в мережі, блокується [7,9,13].

Більшість алгоритмів консенсусу припускають фіксований набір вузлів, які беруть участь у голосуванні, що означає, що не можна просто додати чи видалити вузли в кластері. Існують розширення (Vertical Paxos, Raft’s joint consensus, Dynamic Plain Paxos), до алгоритмів консенсусу які дозволяють змінювати кількість вузлів у кластері з часом, але вони менш зрозумілі та продуктивні, ніж алгоритми статичного членства [27-28].

Для досягнення консенсусу в системі зазвичай покладаються на затримки для виявлення несправних вузлів. У середовищах із дуже мінливими мережевими затримками, особливо в географічно розподілених системах, часто буває, що вузол помилково вважає, що лідер вийшов з ладу через тимчасову проблему мережі. Хоча ця помилка не завдає шкоди роботі системи в загальному, але це призводить до переобрання лідера, оскільки інша частина системи помилково вважає що лідер не є працездатним. З іншого боку часте переобрання лідера призводить до просідання продуктивності системи загалом, оскільки система може витрачати більше часу на вибір лідера, ніж на виконання будь-якої корисної роботи [29-31].

Іноді алгоритми консенсусу особливо чутливі до проблем мережі. Наприклад, в результаті роботи алгоритму Raft [7] може настати випадок, коли вся мережа працює правильно, за винятком одного конкретного мережевого зв'язку, який постійно підводить систему. При цьому відбувається постійна зміна лідера, що в свою чергу провокує додаткові затримки в роботі всієї системи, а відтак система фактично ніколи не прогресує [32-34]

## 1.2. Аналіз методів покращення функціонування розподілених сервісних систем

Останні дослідження у сфері розподілених обчислень показують, що підвищення продуктивності таких систем значною мірою залежить від здатності адаптивно виявляти відмови. Важливою віхою в цьому напрямі стало запровадження концепції ненадійних детекторів збоїв — механізмів, які з певною похибкою оцінюють доступність процесів у системі. Ці детектори відіграють ключову роль у досягненні консенсусу в асинхронному середовищі [35-36].

При цьому доведено, що в повністю асинхронній системі неможливо досягти детермінованого консенсусу, якщо хоча б один процес може вийти з ладу [37-38]. Це обмеження створює серйозні виклики для класичних алгоритмів у

нестабільних мережах. Ненадійні детектори відмов, хоч і не гарантують повну точність, дозволяють компенсувати цю проблему, надаючи достатньо інформації для побудови практично працездатних алгоритмів консенсусу, здатних функціонувати навіть за умов збоїв і затримок [39]. Тож для вирішення проблем пов'язаних із досягненням консенсусу розподілених інфокомунікаційних систем і як наслідок підвищення відмовостійкості розподілені сервісні системи використовують детектори відмов [40-41].

У розподілених системах виявлення збоїв є критично важливим для підтримання надійності та доступності системи. Механізми виявлення збоїв визначають і реагують на збої в мережі, дозволяючи системі вжити коригувальних заходів, щоб запобігти подальшим проблемам, таким як каскадні збої або зниження продуктивності. Асинхронна природа розподілених систем, разом з ненадійністю мережі, робить виявлення збоїв складним, але необхідним елементом для досягнення консенсусу та стійкості до збоїв [42-47].

Виявлення збоїв у свою чергу в розподілених системах є складним завданням через:

- Асинхронний зв'язок: В асинхронних середовищах немає гарантій щодо часу доставки повідомлень, що ускладнює розрізнення між затримкою відповіді та збоєм.
- Збої процесів та затримки: Процес може повністю вийти з ладу, працювати повільно або затримуватися з відповіддю через мережеві перевантаження. Розрізнення цих сценаріїв може призвести як до хибно-позитивних результатів (неправильна підозра на працездатний процес), так і до хибно-негативних (нездатність виявити відмови) [48].
- Компроміс між безпекою та життєздатністю: Потрібно зберігати баланс між безпекою (гарантія відсутності хибних рішень про відмови) і життєздатністю (гарантія, що всі збої будуть зрештою виявлені), що є центральним аспектом у проєктуванні засобів виявлення збоїв [49].

Відмови можуть бути виявлені на двох рівнях:

- Рівень зв’язку: Збої у зв’язку між процесами (наприклад, втрата або затримка повідомлень)[50].
- Рівень процесу: Збої всередині самого процесу (наприклад, аварійне завершення або тривала нечутливість)[51].

До основних механізмів виявлення збоїв можна віднести:

1. *Механізми надсилання сигналу працездатності (heardbeat) та запитів опитування (ping)*

Стан віддалених процесів можна перевіряти за допомогою одного з двох періодичних механізмів:

- Запит опитування (ping): процеси надсилають повідомлення іншим віддаленим процесам, перевіряючи їхню активність, і очікують відповідь протягом визначеного часу.
- Повідомлення сигналу працездатності (heardbeat): процес активно інформує своїх сусідів про те, що він працює, надсилаючи їм відповідні повідомлення.

Для прикладу розглянемо механізм запитів опитування, однак ту саму задачу можна вирішити за допомогою сигналу працездатності (heardbeat), досягаючи схожих результатів.

Кожен процес підтримує список інших процесів (активних, тих які відмовили та підозрюваних) та оновлює його, зберігаючи час останньої відповіді від кожного процесу [52-53]. Якщо процес не відповідає на запит протягом тривалого часу, він позначається як підозрюваний [54].

На рисунку 1.6 зображено нормальну роботу системи: процес A запитує стан сусіднього вузла B, який відповідає підтвердженням.

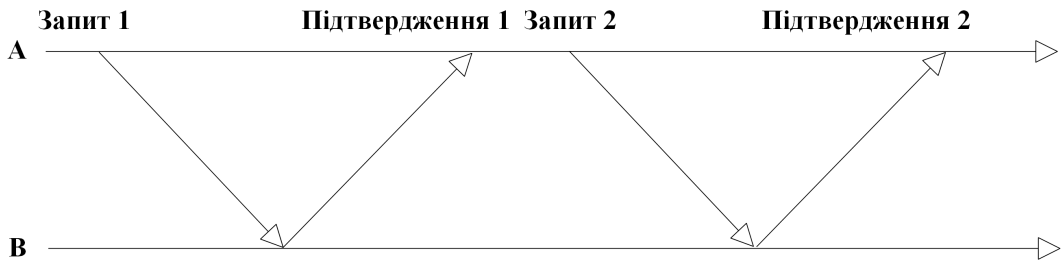



Рис. 1.6. Використання запитів опитування для виявлення збоїв: нормальна робота, без затримок повідомлень.

У протилежність цьому, на рисунку 1.7 зображено, як затримки у відповіді можуть призвести до хибного позначення активного процесу як збійного.

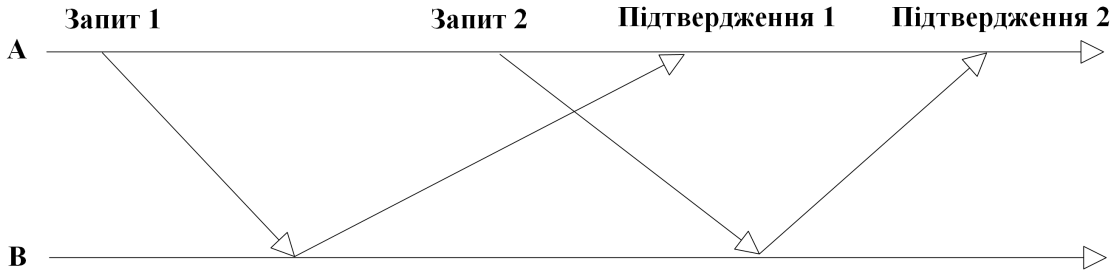



Рис. 1.7. Використання запитів опитування для виявлення відмов: відповіді затримані, надходять після наступного запиту.

Багато алгоритмів виявлення збоїв базуються на механізмах heardbeat і граничному часі очікування. Наприклад, Akka, популярний фреймворк для побудови розподілених систем (в основі якого лежить алгоритм консистентності), реалізує такий детектор збоїв. Він використовує сигнал працездатності (heardbeat) та повідомляє про відмову процесу, якщо той не зареєструвався протягом фіксованого інтервалу часу [55].

Цей підхід має кілька потенційних недоліків: його точність залежить від ретельного налаштування частоти запитів опитування і тайм-аутів, а також він не враховує видимість процесу з точки зору інших процесів.

*2. Детектор відмов без використання мехінізмів очікування*

Механізм heardbeat, який не використовує тайм-аутів, визначає збої процесів, базуючись лише на підрахунку повідомлень. Цей підхід дозволяє

виявляти відомви процесів за даними лічильників сигналу працездатності (heardbeat), працюючи в умовах асинхронності системи.

Детектор передбачає, що всі коректні процеси з'єднані один з одним через канали зв'язку, які гарантують отримання повідомлень та кожен процес знає про існування всіх інших процесів у мережі.

Кожен процес підтримує список сусідів і лічильників, асоційованих із ними. Процеси починають із надсилання повідомлень своїм сусідам. Кожне повідомлення містить шлях, яким воно пройшло, і унікальний ідентифікатор для уникнення повторного надсилання того самого повідомлення.

Коли процес отримує нове повідомлення, він збільшує лічильник для всіх учасників, зазначених у шляху, і пересилає повідомлення тим, кого ще немає в шляху, додаючи себе до цього шляху. Процеси припиняють розповсюдження повідомлень, щойно всі відомі процеси вже отримали повідомлення (тобто ідентифікатори процесів присутні у шляху).

Оскільки повідомлення передаються через різні процеси, а шляхи повідомлень містять агреговану інформацію від сусідів, можна правильно позначити недосяжний процес як активний, навіть якщо прямий зв'язок між двома процесами несправний.

Лічильники повідомлень представляють глобальну й нормалізовану картину системи. Вони демонструють, як повідомлення поширюються відносно одне одного, дозволяючи порівнювати процеси. Проте одним із недоліків цього підходу є складність інтерпретації лічильників повідомлень: необхідно обрати поріг, який забезпечить надійні результати. Без чітко визначеного порогу алгоритм може помилково позначати активні процеси як підозрювані [56].

*3. Детектор відомов на основі делегованого механізму heardbeat.*

Делеговані сигнали працездатності (heardbeat), які реалізовані в протоколах на кшталт SWIM, підвищують надійність завдяки використанню інформації про живучість процесу з перспективи його сусідів. Цей підхід не вимагає, щоб

процеси знали про всі інші процеси в мережі, достатньо лише підмножини пов'язаних вузлів.

Як показано на рисунку 1.8, процес A надсилає повідомлення процесу B. Якщо B не відповідає, A обирає кілька випадкових членів (наприклад, C і D). Ці випадкові вузли намагаються надіслати повідомлення до B і, якщо отримують відповідь, ухвалити точніші рішення [57].

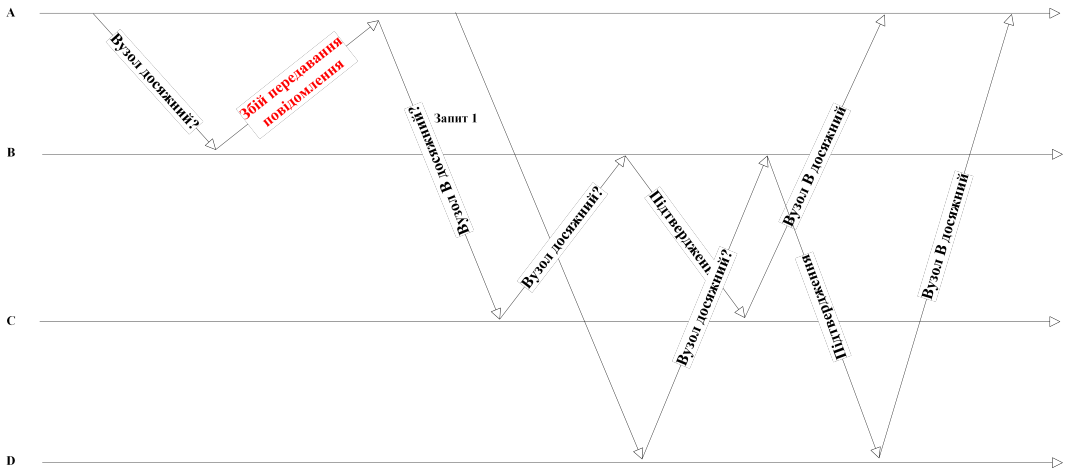



Рис. 1.8. Використання детектору делегованого сигналу працездатності (heardbeat) для виявлення відмов.

Це дозволяє враховувати як пряму, так і непряму досяжність. Наприклад, якщо є процеси A, B і C, стан C можна перевірити з перспективи як A, так і B.

Делеговані повідомлення забезпечують надійне виявлення відмов шляхом розподілу відповідальності за рішення серед групи учасників. Цей підхід не вимагає широкомовного розсилання повідомлень до великої групи вузлів. Оскільки запити делегованих серцебиттів можуть запускатися паралельно, метод дозволяє швидко зібрати більше інформації про підозрювані процеси та ухвалити точніші рішення [58].

**4. φ - чутливий детектор збоїв**

φ -детектор ґрунтується на аналізі часу отримання чергового повідомлення від вузла [59]. Нехай $T$ — випадкова змінна, що представляє собою час надходження повідомлення. Припустимо, що $T$ розподілений експоненційно,

відповідно ймовірність того, що час надходження наступного повідомлення $T$ перевищує значення $t$, дорівнює експоненті з від'ємним параметром $\lambda t$:

$$P(T > t) = e^{-\lambda t} \tag{1.1}$$

де $\lambda$ — це параметр інтенсивності отримання повідомлень, а $t$ — час, що минув від останнього отриманого повідомлення.

Функція $\varphi$ -детектора визначається як:

$$\varphi(t) = \lambda \times t \times \log_{10} e \tag{1.2}$$

Таким чином, $\phi(t)$ є лінійною функцієюасу зростання, що пропорційна інтенсивності $\lambda$.

φ -детектор ухвалює рішення про збій на основі порівняння значення $\phi(t)$ із заздалегідь визначеним порогом $\phi_{\text{threshold}}$. Якщо у певний момент часу виконується умова 1.3 то вузол вважається збійним.

$$\phi > \phi_{\text{threshold}} \tag{1.3}$$

Значення порогу $\phi_{\text{threshold}}$ визначається на основі статистичного аналізу попередніх даних про функціонування системи, що дозволяє адаптувати детектор до специфіки конкретного середовища.

Цей підхід був розроблений дослідниками з Японського інституту передових наук і технологій та зараз використовується в багатьох розподілених системах, таких як Cassandra і Akka (поряд із детектором збоїв на основі сигналу працездатності (heardbeat)) [59-63].

*5. Виявлення відмов на основі пліток*

Протоколи на основі пліток (gossip) використовують випадковий обмін повідомленнями між вузлами для виявлення збоїв [64-65]. Кожен вузол підтримує список інших членів системи, їхні лічильники сигналу працездатності (heardbeat) та часові мітки, що вказують, коли востаннє був збільшений

лічильник сигналу працездатності (heardbeat). Періодично кожен вузол збільшує свій лічильник сигналу працездатності (heardbeat) та передає оновлений список випадковому сусіду. При отриманні повідомлення сусідній вузол об'єднує отриманий список зі своїм, оновлюючи лічильники сигналів працездатності (heardbeat) для інших вузлів [66]. Крім того, вузли періодично перевіряють свій список. Якщо вузол не оновлював свій лічильник сигналу працездатності (heardbeat) протягом визначеного часу, він вважається збійним. Цей період часу очікування потрібно налаштовувати обережно, щоб мінімізувати ймовірність хибнопозитивних результатів.

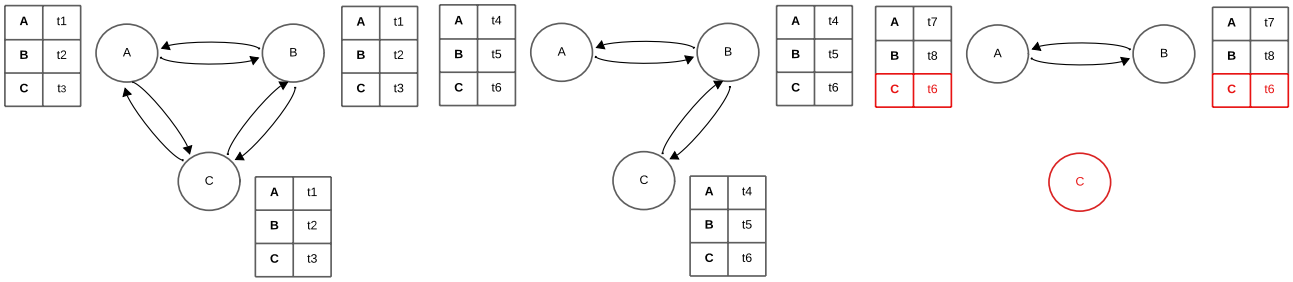



Рис. 1.9. Реплікована таблиця сигналу працездатності (heardbeat) для виявлення збоїв.

Система працює наступним чином:

1. Всі вузли можуть обмінюватися повідомленнями та оновлювати часові мітки.
2. Якщо вузол (наприклад, C) не може напряму спілкуватися з іншим вузлом (A), його статус все одно може бути переданий через проміжний вузол (B).
3. У випадку збою вузла (наприклад, C), оскільки він припиняє надсилати оновлення, інші вузли виявляють його як збійний.

Цей підхід дозволяє виявляти як збої вузлів, так і недосяжність вузлів для інших членів кластера. Рішення є надійним, оскільки огляд стану кластера агрегується з багатьох вузлів. У випадку збою зв'язку між двома вузлами сигналу працездатності (heardbeat) все одно можуть поширюватися через інші вузли. Використання пліток для розповсюдження станів системи збільшує кількість повідомлень у системі, але забезпечує надійніше поширення інформації.

*6. Служи сповіщення про відмови FUSE.*

Підхід служби сповіщення про відмови (FUSE) [67] забезпечує надійне та економічне поширення інформації про відмови навіть у випадках мережевих поділів.

Для виявлення збоїв процесів FUSE організовує всі активні процеси в групи. Якщо одна з груп стає недоступною, усі її учасники виявляють відмову. Іншими словами, кожен раз, коли виявляється відмова одного процесу, вона перетворюється на групову відмову і поширюється на всі інші вузли. Це дозволяє виявляти збої за будь-якої схеми відключень, поділів або збоїв вузлів.

Учасники групи періодично надсилають повідомлення опитування іншим членам, запитуючи, чи вони все ще активні. Якщо один із членів не може відповісти через відмову, мережевий поділ або пошкодження зв'язку, ініціатор запиту сам припиняє відповідати на повідомлення інших членів.

Система працює таким чином:

- Початковий стан: усі процеси активні та можуть взаємодіяти.
- Один із процесів (наприклад, B) припиняє відповідати на повідомлення (відмова).
- Інший процес (наприклад, D) виявляє відмову B і сам припиняє відповідати.
- Згодом інші процеси (A та C) помічають, що як B, так і D не відповідають, відмова поширюється на всю групу.

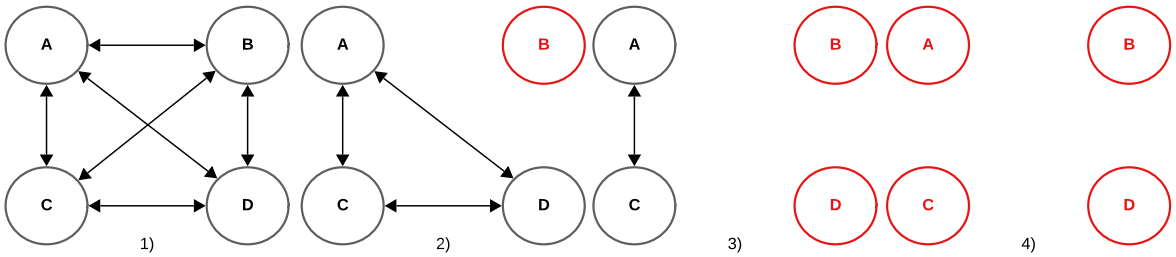



Рис. 1.10. Виявлення помилок за допомогою служби сповіщення про відмови FUSE.

Таким чином, усі збої поширюються системою від джерела відмови до всіх інших учасників. Учасники поступово припиняють відповідати на пінги, перетворюючи індивідуальний збій на груповий.

Таблиця 1.1

Порівняння методів виявлення відмов

| **Метод** | **Повнота** | **Точність** | **Масштабованість** | **Ідеальний випадок використання** |
|---|---|---|---|---|
| Heardbeat/запит-опитування | Середня | Низька | Висока | Прості, невеликі системи |
| Делеговані heardbeat | Висока | Середня | Середня | Стійкі до збоїв розподілені середовища |
| $\varphi$-чутливий | Висока | Висока | Середня | Адаптивні, мережі з високою затримкою |
| На основі пліток | Середня | Середня | Дуже висока | Великі, децентралізовані системи |
| Служби сповіщень про відомви (FUSE) | Висока | Середня | Висока | Розподілені мережі з узгодженістю у групах |

Засоби виявлення збоїв є незамінними для підтримання надійності розподілених систем. Вибір відповідного методу виявлення збоїв дозволяє досягти оптимального балансу між життєздатністю, точністю та ефективністю. Однак вибір детектора повинен відповідати конкретним потребам системи, включаючи масштабованість, стабільність мережі та стійкість до збоїв.

В асинхронному середовищі жоден засіб виявлення збоїв не є ідеальним. Кожен підхід вимагає компромісів, і часто використовується комбінація методів для досягнення надійності [68-69].

## 1.3. Забезпечення надійної та узгодженої роботи алгоритмів розподілених сервісних систем при динамічності зміни умов функціонування мережі

У реальних умовах розподілені сервісні системи, які використовують алгоритми консенсусу, стикаються з непередбачуваними змінами в мережевому середовищі. Це створює значні труднощі для забезпечення надійної та узгодженої роботи алгоритмів, оскільки мережеві затримки, втрата пакетів і короткочасні збої зв'язку здатні вплинути на узгодженість даних та ефективність системи в цілому.

Мережеві затримки є змінними і часто залежать від таких факторів, як завантаженість мережі, фізична відстань між вузлами, конфлікти між пакетами, зміна пріоритетів обробки даних та сторонні перешкоди, особливо в бездротових середовищах. У дослідженні [20] встановлено, що висока затримка під час передачі пакетів у консенсусних протоколах, таких як Paxos і Raft, може спричиняти значні порушення у виборі лідера, збільшуючи кількість помилкових рішень про відомви та викликаючи додаткові витрати на обчислення, необхідні для відновлення узгодженості [35,66,70-71].

Залежно від конфігурації мережі, тимчасові затримки можуть варіюватися від кількох мілісекунд до кількох секунд, що ускладнює прогнозування роботи алгоритмів і коректне налаштування параметрів. Такі зміни затримок не завжди можна заздалегідь врахувати, що значно ускладнює побудову точних моделей для прогнозування поведінки алгоритму у реальних умовах. Крім того, цей аспект перешкоджає визначенню порогів, коли мережеві умови ще є прийнятними для алгоритму або коли затримки вже створюють загрозу узгодженості даних у системі [72].

Втрата пакетів і переривання зв'язку можуть спричиняти додаткові ускладнення. Якщо вузол не отримує сигналу від лідера через втрату пакету або короткочасне переривання зв'язку, це може спричинити помилкове рішення про переобрання лідера або переналаштування системи, хоча фактично вузол працює

коректно [73]. Така помилка може бути дуже вагомою у критичних системах, оскільки помилкові рішення не лише знижують продуктивність, але й призводять до порушень узгодженості та до потенційної втрати даних. У літературі наголошується, що в системах із високим рівнем навантаження мережеві збої, навіть короткочасні, можуть провокувати непотрібні переобрання лідера та переналаштування, що значно знижує ефективність алгоритму консенсусу [74-76].

Існує також проблема «ефекту доміно», коли нестабільність у мережі одного вузла може спричинити серію помилкових спрацьовувань і переналаштувань на інших вузлах. Це явище є особливо критичним у великих системах, де кожен вузол залучений до численних процесів синхронізації та узгодження даних з іншими вузлами. У дослідженні [77] продемонстровано, що в умовах високої варіативності мережевих затримок вузли можуть багаторазово втрачати лідерство, через що система переходить у нестабільний стан, де всі вузли не можуть дійти згоди протягом тривалого часу, суттєво знижуючи загальну ефективність роботи [78-79].

Додатковим ускладненням є неможливість стабілізації параметрів алгоритмів під такі умови. Невідповідність параметрів налаштувань, таких як інтервали затримки та сигнал працездатності (heardbeat), для змінних мережевих умов може призводити до нестабільної роботи алгоритму, що викликає непотрібні обчислювальні витрати [80]. Наприклад, якщо час очікування на повідомлення занадто короткий, система може часто реагувати на короткочасні збої як на постійні відмови вузлі. Навпаки, довгі тайм-аути можуть призводити до , що також може негативно вплинути на узгодженість і надійність роботи системи [81-83].

Зважаючи на все зазначене, проблеми, пов’язані з мережевою нестабільністю, суттєво ускладнюють оцінку надійності алгоритмів консенсусу [84]. Розробка адаптивних моделей, здатних враховувати реальні умови, стає

важливим напрямом досліджень, що дозволить покращити точність оцінки роботи консенсусних алгоритмів у змінних мережевих середовищах [85].

### 1.3.1. Стабільність структурних параметрів як критерій ефективного функціонування алгоритмів консенсусу

Оцінка ефективності алгоритмів консенсусу в розподілених сервісних системах суттєво ускладнюється через неможливість створення стабільних, постійних умов тестування в реальному середовищі. Кожна операція та передача даних у таких системах може мати унікальні затримки і різні шляхи доставки, що ускладнює забезпечення послідовності та повторюваності результатів. У розподілених середовищах важко отримати однакові умови тестування навіть для коротких часових періодів, а найменша зміна в завантаженні мережі або в топології системи може суттєво вплинути на продуктивність алгоритму.

Розподілені сервісні системи зазвичай мають багато незалежних вузлів, які піддаються впливу зовнішніх факторів, таких як мережеві аномалії, збої обладнання або програмні помилки. Це створює додаткові труднощі в прогнозуванні й повторюванні мережевих умов, необхідних для ефективної оцінки алгоритмів консенсусу. Наприклад, мережеві збої можуть мати випадковий характер: якщо на одному з етапів тестування виникне збій у певному вузлі або з'єднанні, то це може суттєво вплинути на результат роботи алгоритму. У наступному тестуванні такий збій може не виникнути, що призведе до значної різниці у результатах [86-89].

Ця проблема особливо критична для алгоритмів консенсусу, де коректність і стабільність роботи залежать від синхронності дій вузлів, які можуть втрачати зв'язок або зупинятися з причин, не пов'язаних з самим алгоритмом. Тому навіть незначні зміни умов можуть вплинути на результати роботи алгоритмів консенсусу [90-92]. Наприклад, якщо під час одного тесту у системі спостерігаються короткочасні затримки в передачі даних, це може спричинити хибні рішення про відмову вузлів, які призведуть до зміни лідера, хоча алгоритм

у цей момент функціонує правильно. Інше тестування, проведене за відсутності таких збоїв, може показати кардинально інші результати, створюючи труднощі у проведенні послідовних оцінок.

Ще однією важливою проблемою є вплив різного навантаження на систему, що може суттєво змінюватися протягом тестування. Наприклад, інтенсивне навантаження може спричиняти збільшення мережевих затримок, що призведе до хибних результатів оцінки. Різниця у навантаженні може вплинути на те, як швидко вузли обробляють інформацію, викликаючи зміни в поведінці алгоритмів консенсусу та підвищуючи кількість помилкових реконфігурацій системи [93-94].

Більше того, оцінка алгоритмів консенсусу ускладнюється через відсутність повного контролю над мережею в реальних умовах, що часто унеможливлює відтворення певних типів відмов. У наукових дослідженнях часто використовуються симуляційні середовища для імітації певних мережевих умов, але навіть такі методи не завжди можуть врахувати всю складність реальної мережі, модельовані умови часто є спрощеними і не можуть повністю відображати всі можливі аномалії реальних мережевих середовищ.

До складнощів з відтворенням умов тестування також відноситься і залежність від поточних налаштувань мережі та параметрів конфігурації, які можуть постійно змінюватися. Наприклад, якщо тестування проводиться в середовищі, де параметри, такі як затримки передачі сигналів, автоматично змінюються в залежності від навантаження, це ускладнює відтворення результатів тестування і отримання стабільних оцінок для алгоритмів консенсусу [95-96].

Таким чином, оцінка ефективності алгоритмів консенсусу в розподілених сервісних системах в умовах реального середовища обмежується мінливістю зовнішніх факторів і складністю забезпечення постійних умов для тестування. Для зменшення впливу цих факторів можна використовувати спеціальні імітаційні моделі та симуляційні середовища, які дозволяють контролювати

змінні та відтворювати деякі критичні умови, однак вони не завжди повністю замінюють реальні мережеві тести.

### 1.3.2. Вплив помилкових спрацювань на ефективність функціонування розподілених сервісних систем

Оцінка ефективності алгоритмів консенсусу значно ускладнюється через помилкові виявлиння відмов, коли система помилково ідентифікує працюючий вузол як непрацездатний. Це може бути викликано короткочасними мережевими затримками, втратою пакетів або перериванням зв'язку, що сприймаються системою як сигнал про збої вузлів. Неправильне виявлення таких відмов може призвести до передчасних переналаштувань, переобрання лідера або інших ресурсомістких процесів, які суттєво впливають на роботу та стабільність системи, особливо в реальних розподілених мережах [97].

Проблема тестування ефективності алгоритмів консенсусу ускладнюється тим, що у разі хибного сигналу про несправність система змушена реагувати так, як насправді в ситуації реальної відмови. Внаслідок цього:

- *Виникає непередбачувана поведінка:* якщо система сприймає помилкове спрацьовування за реальний збій, алгоритм активує процедури відновлення, такі як переобрання лідера або переналаштування структури кластеру. Кожне переналаштування змінює розподіл навантаження і розподіл задач у системі, що може впливати на її загальну продуктивність та пропускну здатність. Це робить тестування неточним, оскільки кожне хибне спрацьовування може генерувати нову конфігурацію системи з різним рівнем ефективності, що ускладнює вимірювання стабільності та продуктивності алгоритму в однорідних умовах [98-99].
- *Додаткові витрати на обчислення:* помилкові спрацьовування призводять до запуску алгоритмів відновлення, які споживають додаткові ресурси. Наприклад, процес переобрання лідера або переналаштування топології кластеру вимагає значних витрат часу й ресурсів на повторну синхронізацію вузлів і

оновлення зв’язків між ними. Це ускладнює отримання точних даних про продуктивність алгоритму, оскільки в реальних умовах система витрачає ресурси не лише на обробку даних, але й на управління помилковими відмовами [100-108].

- *Відсутність стабільних умов тестування:* під час тестування у змодельованих умовах можна налаштувати параметри алгоритму так, щоб вони мінімізували ймовірність хибних спрацьовувань. Однак у реальних мережах, де параметри такі як затримки, пропускна здатність і навантаження постійно змінюються, уникнути помилкових спрацьовувань складно. Це означає, що система під час тестування та в реальній роботі може показувати суттєво різні результати [109-113].

- *Взаємозалежність між хибними рішеннями про відмови та адаптивними параметрами алгоритму:* багато алгоритмів консенсусу, таких як Raft або Paxos, використовують фіксовані параметри для налаштування інтервалів сигналу працездатності (heardbeat) (heartbeat) та очікування повідомлень. Надто короткий час очікування підвищує чутливість системи до хибних спрацьовувань, але надто довгий – знижує чутливість до реальних збоїв, що може затримувати реакцію системи. Оптимізація цих параметрів для конкретних умов у реальному середовищі є складною задачею, а її досягнення потребує частих експериментів, які можуть призвести до додаткових хибних спрацьовувань [114-117].

- *Вплив хибних рішень про відмови на результативність системи:* кожне помилкове рішення про відмову створює додаткові навантаження, що може змінити результати тестування. Наприклад, якщо під час тестування виникають помилкові переналаштування або переобрання, система використовує значну частину ресурсів на стабілізацію замість обробки запитів, що призводить до хибної оцінки продуктивності алгоритму. У результаті тестування не відображає реальних умов роботи, коли система повинна обробляти запити без постійних переналаштувань і переобрань [118-120].

Таким чином, хибні рішення про відмови створюють значні виклики для алгоритмів консенсусу в реальних умовах. Кожне хибне рішення не тільки спотворює реальну продуктивність системи, але й ускладнює аналіз результатів, оскільки алгоритм намагається обробляти не лише запити, а й зайві переналаштування. Це робить тестування неточним і вимагає від дослідників застосування складних моделей або симуляцій для оцінки роботи алгоритмів у штучно контрольованих умовах, які не завжди точно відображають реальну поведінку алгоритму у нестабільному мережевому середовищі.

### 1.4. Висновки до першого розділу

1. Проведено аналіз основних методів і моделей, які забезпечують функціонування розподілених сервісних систем. Встановлено, що основою узгодженості функціонування розподілених систем є робота алгоритмів консенсусу, що забезпечують їх стабільну роботу та є основним механізмом, що гарантує узгодженість даних між вузлами навіть у випадку збоїв. У результаті проведеного аналізу встановлено, що забезпечення надійності розподілених сервісних систем є складною задачею та вимагає комплексного підходу, здатного враховувати різноманітні джерела збоїв, зокрема апаратні, програмні та мережеві. Особливу складність становить високий рівень динамічності, масштабованості та гетерогенності сучасних середовищ, у яких мікросервісні додатки взаємодіють через нестабільні мережеві шляхи та можуть мати різну поведінку залежно від навантаження. Це зумовлює необхідність впровадження інтелектуальних засобів моніторингу, адаптації та самовідновлення.

2. Ключовим елементом підтримки стійкості розподіленої системи є своєчасне виявлення збоїв (fault detection), що забезпечується застосуванням як традиційних методів контролю стану (наприклад, перевірка часів відповіді та heartbeat-механізмів), так і сучасних статистичних та машинних підходів до виявлення відмов. Ефективним вирішенням цієї задачі є ймовірнісні підходи, яі дозволють оцінювати ступінь ймовірності відмов залежно від поведінкових та

мережевих метрик, що сприяє підвищенню точності реагування. Підвищення толерантності до збоїв (fault tolerance) досягається через використання технік реплікації сервісів, балансування навантаження, резервного виділення ресурсів, а також застосування алгоритмів досягнення консенсусу для забезпечення узгодженості стану системи. Здатність системи до швидкої реконфігурації в умовах часткових відмов є критичною передумовою збереження її функціональної цілісності та надання безперервних сервісів кінцевим користувачам.

3. Розглянуто основні переваги та недоліки при застосуванні існуючих методів забезпечення стабільної роботи алгоритмів консенсусу в розподілених системах із динамічно змінною структурою мережі. Сформульовано основні невирішені задачі в галузі забезпечення узгодженості та працездатності системи в умовах постійних відмов та реконфігурацій її функціональних вузлів. У розподілених сервісних системах, що функціонують в умовах нестабільних і непередбачуваних мережевих затримок, ефективність виявлення відмов критично залежить від раціонального вибору порогового значення часу очікування повідомлень від вузлів (порогового тайм-ауту). Занижене значення тайм-ауту дозволяє системі оперативно реагувати на потенційні збої, що сприяє швидкому переобранню лідера та мінімізації простоїв. Водночас надмірне зменшення цього порогу підвищує ймовірність хибнопозитивного спрацювання механізму виявлення відмов, що може спричинити зайві процедури реконфігурації, порушення узгодженості даних та загальне зниження продуктивності системи. Таким чином, необхідність балансування між швидкістю реагування на відмови та точністю функціонування механізмів виявлення збоїв потребує розв’язання науково-практичного завдання – розроблення нових методів та моделей прогнозованого адаптивного виявлення відмов, які враховуватимуть поточні характеристики мережевого середовища, динаміку затримок і поведінку вузлів у кластері та визначати їх порогові значення.

# РОЗДІЛ 2. РОЗРОБЛЕННЯ АДАПТИВНИХ МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ КОНСЕНСУСУ В РОЗПОДІЛЕНИХ СЕРВІСНИХ СИСТЕМАХ

У сучасних розподілених сервісних системах алгоритми консенсусу відіграють ключову роль у забезпеченні узгодженості даних і стійкості до збоїв. Проте їх застосування стикається з численними викликами, пов’язаними зі зростаючою складністю середовища, динамічністю мереж і непередбачуваними змінами в умовах роботи. Особливо критичними є проблеми, що виникають через затримки в мережі, хибнопозитивні визначення збоїв та неефективне переобрання лідера.

Традиційні алгоритми, такі як Raft, зазвичай використовують статичні параметри, які, хоча й забезпечують базову функціональність, часто виявляються недостатніми для динамічного середовища. Статичність таких параметрів, як тайм-аути, обмежує адаптивність системи, що призводить до непотрібних реконфігурацій, втрати продуктивності та збільшення часу досягнення консенсусу. Ці проблеми підкреслюють важливість впровадження нових підходів, здатних враховувати мінливість умов у реальному часі [20].

У цьому розділі розглядаються адаптивні методи, спрямовані на підвищення ефективності функціонування алгоритмів консенсусу. Запропоновані підходи охоплюють динамічне коригування параметрів на основі поточного стану мережі, використання технік машинного навчання для прогнозування можливих збоїв і інтеграцію ймовірнісного аналізу, зокрема теореми Байєса, для зменшення хибнопозитивних рішень.

Особливу увагу приділено розробці методу, який не лише може виявляти збої в системі, але й коригувати вибір оптимального лідера в кластері. У межах цього методу розроблено алгоритм адаптивного налаштування тайм-аутів, який враховує як затримки між вузлами, так і стан мережі в цілому. Цей підхід дозволяє знизити кількість помилкових переобрань лідера, оптимізувати продуктивність кластера та забезпечити стабільність роботи системи.

## 2.1. Метод виявлення відмов у роботі розподілених сервісних систем на основі ймовірнісного байєсівського підходу.

Ефективне функціонування розподілених сервісних систем критично залежить від здатності виявляти збої вузлів і вирішувати проблеми консенсусу в умовах динамічного середовища та непередбачуваних мережевих затримок.

Пропонуємо інтегрувати концепції адаптивного налаштування параметрів із використанням теореми Байєса, що дозволяє системам враховувати нові дані для прийняття більш обґрунтованих рішень, шляхом мінімізації хибнопозитивних рішень про відмови і підвищенню загальної стійкості системи до тимчасових мережевих порушень.

Для зменшення кількості хибнопозитивних рішень про відмови важливо вбудувати в алгоритми консенсусу механізми виявлення збоїв, які можуть точно розрізняти справжні збої та тимчасові проблеми. Розвиваючи $\varphi$-чутливий детектор збоїв, запропонований, що використовує ймовірнісний підхід до виявлення збоїв, розширимо цю концепцію за допомогою адаптивних технік. Робота [59] акцентує увагу на адаптивності шляхом динамічного налаштування параметрів для зменшення хибнопозитивних спрацьовувань.

Теорема Байєса є фундаментальною концепцією теорії ймовірностей, яка надає математичну основу для оновлення ймовірності гіпотези на основі нових подій. Ця теорема особливо корисна для прийняття рішень в умовах невизначеності, оскільки вона дозволяє постійно уточнювати переконання у міру надходження нових даних. У контексті алгоритмів консенсусу теорема Байєса може бути застосована для покращення механізмів виявлення відмов, забезпечуючи ймовірнісну оцінку того, чи вузол дійсно вийшов з ладу, чи пропущений сигнал працездатності (heardbeat) є наслідком тимчасової проблеми, такої як затримка в мережі.

Байєсівський висновок працює шляхом поєднання попереднього знання (початкового припущення про ймовірність збою) із новими доказами (наприклад, пропущеним сигналом працездатності (heardbeat) або затриманим

повідомленням) для обчислення апостеріорної ймовірності. Ця апостеріорна ймовірність відображає оновлене переконання щодо статусу збою вузла. Ключовою перевагою цього підходу є його гнучкість у пристосуванні до нової інформації, що допомагає зменшити кількість хибнопозитивних рішень в алгоритмах консенсусу.

Використовуючи теорему Байєса, системи можуть уникати поспішних рішень, заснованих лише на одному доведенні. Замість цього вони можуть зважувати ймовірність різних факторів (таких як затримка доставки повідомлень, завантаженіть вузлів, рівень використання оперативної пам’яті) і приймати більш обґрунтовані, ймовірнісні рішення. Це зменшує ймовірність непотрібних переобрань лідера чи активації резервних механізмів відновлення після відмов, що в підсумку підвищує загальну надійність і продуктивність розподілених систем.

Теорему Байєса можна систематично застосовувати в алгоритмах консенсусу для покращення виявлення збоїв, постійно уточнюючи ймовірність збою вузла на основі нових подій. Ось як цей підхід реалізовуватимемо поетапно, загальний підхід можна побачити на рисунку 2.1.

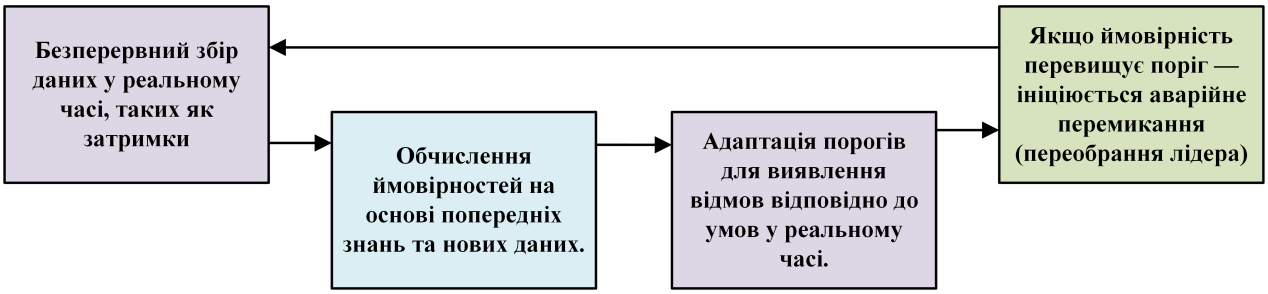



Рис. 2.1. Концепція підходу на основі теореми Байєса до розрахунку ймовірнісного виявлення збоїв

*I. Визначення апріорної ймовірності*

Спершу визначимо апріорну ймовірність $P(Failure)$, яка відображатиме початкове припущення про ймовірність того, що вузол вийде з ладу. Ця ймовірність може базуватися на історичних даних, наприклад, на попередніх

випадках збоїв вузлів у подібних ситуаціях, або на загальному рівні відмов вузлів у системі. У нашому випадку будемо здійснювати аналіз системного журанлу подій для наповнення масиву даних ймовірностей [58].

Далі збиратимемо дані з системи в реальному часі, які можуть вказувати на потенційні збої. За основу візьмемо пропущені сигнали працездатності (heardbeat), затримки у відповідях або аномальну поведінку вузла, позначимо її як $E$. Ці події можуть надходити як в реальному часі (наприклад, пропущені сигнали працездатності (heardbeat)), так і з результату аналізу системного журанлу. [121-123]

*1. Визначення ймовірності (likelihood).*

На наступному етапі обчислимо ймовірність $P(E \mid Failure)$, яка відображає ймовірність спостереження іподій $E$, якщо вузол дійсно вийшов з ладу. Це може включати статистичні моделі минулих збоїв або емпіричні дані про те, як збої вузлів проявляються в системі. Взалежності від умов функціонування системи ймовірність отримання даних про подію $E$ , можна оцінюватимемо одним із варіантів:

*2. Емпіричні дані та спостереження*

Щоб визначити $P(E \mid Failure)$, можна проаналізувати минулі збої та спостерігати, як часто певні типи подій (наприклад, пропущені сигнали працездатності (heardbeat)) виникали під час цих збоїв.

Наприклад, у сценарії, де було зафіксовано 100 збоїв вузлів, у 80% випадків спостерігалися пропущені сигнали працездатності (heardbeat), у 50% — затримки у відповідях, а в 90% — аномальна поведінка процесора (CPU) вузла. Ці відсотки відображають ймовірність кожної поведінки під час збою вузла. Тобто, існує 80% ймовірність спостереження пропущеного сигналу працездатності (heardbeat), 50% ймовірність зіткнутися із затримкою у відповіді та 90% ймовірність виявити аномальну поведінку процесора під час збою вузла.

*3. Статистичні моделі*

За відсутності достатніх емпіричних даних, для оцінки $P(E \mid Failure)$ можна використовувати статистичні моделі. Нижче наведено кілька моделей, які можуть використовуватимуться в нашому випадку:

1) Логістична регресія. Це модель, яка дозволяє використовувати докази, такі як пропущені сигнали працездатності (heardbeat) або затримки, для прогнозування ймовірності спостереження певного доказу за умови виникнення збою. Наприклад, логістична регресія може моделювати взаємозв'язок між збоєм $F$ і спостережуваним доказом $E$, а також надавати ймовірності того, що доказ $E$ буде спостерігатися за умови, що стався збій. Формула логістичної регресії в такому випадку виглядає так:

$$P(E \mid Failure, E_1, E_2, E_3) = \frac{1}{1 + \exp(-(b_0 + b_1 Failure + b_2 E_1 + b_2 E_2 + b_3 E_3))}, \quad (2.1)$$

Де $E_1$, $E_2$, $E_3$ представляють різні типи подій (наприклад, пропущений сигнал працездатності (heardbeat), затримка у відповіді тощо), а $b_1, b_2, b_3$ — це коефіцієнти, отримані з даних.

2) Наївний байєсівський класифікатор. Це простий ймовірнісний класифікатор, який передбачає, що всі докази є умовно незалежними за умови збою. Ймовірність розраховується шляхом множення індивідуальних ймовірностей кожного доказу за умови збою:

$$P(E \mid Failure) = P(E_1 \mid Failure) \times P(E_2 \mid Failure) \times \ldots \times P(E_n \mid Failure), \quad (2.2)$$

*4. Моделі симуляції*

В дуже складних системах, де неможливо безпосередньо спостерігати достатню кількість збоїв, можна використовувати симуляції або аналітичні моделі. Наприклад, можна створити симуляційне середовище для відтворення розподіленої системи, де вводяться контрольовані сценарії збоїв, такі як мережеві збої, апаратні несправності або перевантаження системи. Під час цих

симуляцій можуть використовуватися інструменти моніторингу для відстеження та запису таких показників, як пропущені сигнали "серцебиття", затримані відповіді та аномальне використання процесора (CPU). Зібрані дані потім аналізуються для визначення $P(E \mid Failure)$.

*5. Моделі машинного навчання*

У деяких застосунках моделі машинного навчання можуть бути навчені на великих наборах даних для автоматичного виявлення шаблонів збоїв вузлів. Наприклад, моделі виявлення аномалій можуть навчитися розпізнавати характерні ознаки несправних вузлів і оцінювати $P(E \mid Failure)$, виходячи з того, наскільки поведінка є аномальною.

*II. Визначення апостеріорної ймовірності*

Наступним важливим кроком є застосування теореми Байєса для обчислення апостеріорної ймовірності. В нашому випадку $P(Failure \mid E)$, яка оновлює ймовірність збою вузла з урахуванням нових подій, визначатиметься:

$$P(E \mid Failure) = \frac{P(E \mid Failure) \times P(Failure)}{P(E)} \tag{2.3}$$

Вона відображатиме уточнену оцінку того, чи вузол вийшов з ладу, з урахуванням як початкового припущення (апріорного), так і нових подійподій.

*III. Порівняння з пороговим значенням*

Після обчислення апостеріорної ймовірності її слід порівняти із заздалегідь визначеним пороговим значенням. Якщо апостеріорна ймовірність $P(Failure \mid E)$ перевищує цей поріг, система може оголосити вузол несправним і ініціювати процеси аварійного відновлення (failover) або переобрання лідера. Якщо ймовірність нижча за порогове значення, система може обрати продовжити моніторинг вузла для отримання додаткових подій перед прийняттям рішення.

Апостеріорна ймовірність відображає уточнену оцінку того, чи вузол вийшов з ладу, з урахуванням як початкового припущення (апріорного), так і нових подій.

Застосування теореми Байєса дозволяє системі приймати більш обґрунтовані, ймовірнісні рішення щодо збоїв вузлів, а не покладатися на бінарні, судження «все або нічого». Динамічно оновлюючи ймовірності збоїв, система може зменшити кількість хибнопозитивних рішень про відмови, тим самим підвищуючи надійність алгоритму консенсусу в розподіленому середовищі. Псевдо код запропонованого алгоритму визначення ймовірнісного виявлення відмов із використанням байєсівського висновку наведено на рисунку 2.2.

**Algorithm 1** Adaptive Probabilistic Failure Detection using Bayesian Inference

**Initialize** $P_{\text{failure}} \leftarrow$ INITIAL_FAILURE_PROBABILITY
**while** system_is_running **do**
    **Gather real-time evidence:**
    $E \leftarrow$ collect_evidence()  ▷ e.g., missed heartbeats, network delays
    **Calculate likelihood:**
    $P(E \mid \text{failure}) \leftarrow$ calculate_likelihood$(E, \text{failure} = \text{True})$
    $P(E \mid \neg\text{failure}) \leftarrow$ calculate_likelihood$(E, \text{failure} = \text{False})$
    **Calculate total probability of evidence:**
    $P(E) \leftarrow P(E \mid \text{failure}) \times P_{\text{failure}} + P(E \mid \neg\text{failure}) \times (1 - P_{\text{failure}})$
    **Update failure probability using Bayes' theorem:**
    $P_{\text{failure}|E} \leftarrow \frac{P(E|\text{failure}) \times P_{\text{failure}}}{P(E)}$
    **if** $P_{\text{failure}|E} >$ FAILURE_THRESHOLD **then**
        **Initiate leader re-election or failover:**
        initiate_leader_re_election()
    **else**
        **Continue monitoring:**
        continue_monitoring()
    **end if**
    **Optionally:** Adjust failure threshold based on context:
    adjust_threshold_based_on_context()
**end while**
**function** COLLECT_EVIDENCE
    **return** evidence data such as missed heartbeats, delayed responses, etc.
**end function**
**function** CALCULATE_LIKELIHOOD($E$, failure)
    **return** likelihood of the evidence $E$ given a failure or no failure
**end function**
**function** INITIATE_LEADER_RE_ELECTION
    Trigger the re-election or failover process
**end function**
**function** ADJUST_THRESHOLD_BASED_ON_CONTEXT
    Adjust the FAILURE_THRESHOLD dynamically based on current network conditions or historical performance
**end function**

Рис. 2.2. Псевдо код алгоритму визначення ймовірнісного виявлення відмов із використанням байєсівського висновку

У традиційних алгоритмах консенсусу часто використовуються статичні порогові значення для запуску процедур відновлення після відмов або переобрання лідера. Такі порогові значення, не адаптуються до змінних умов системи, що може призводити до проблем, наприклад, хибнопозитивних рішенньпро відмови вузлів. Наприклад, фіксований поріг може бути занадто малим під час мережевих перевантажень, через що система помилково визначає справні вузли як несправні [17-20,124-126].

Однак байєсівський підхід відкриває можливість динамічного порогового значення, де пороги налаштовуються в режимі реального часу на основі оновлених ймовірностей, отриманих за допомогою байєсівського висновку. Замість того, щоб покладатися на статичні умови для прийняття рішень, система може використовувати постійно оновлювані ймовірнісні оцінки для визначення моменту запуску конкретних дій. Ось як це працює:

*1. Оновлення ймовірності в реальному часі.* У міру надходження нових подій (наприклад, пропущених сигналів працездатності (heardbeat), затриманих відповідей) теорема Байєса дозволяє системі оновлювати ймовірність того, що вузол вийшов з ладу. Ця ймовірність не є фіксованою, а змінюється динамічно залежно від отриманих подій, що робить систему більш чутливою до поточних умов.

*2. Адаптивні порогові значення для прийняття рішень.* Завдяки байєсівським оновленням поріг для оголошення вузла таким що відмовив або запуску переобрання лідера може динамічно змінюватися. Наприклад, якщо апостеріорна ймовірність збою вузла залишається низькою, незважаючи на пропущені сигнали працездатності (heardbeat) (можливо, через відомі фактори, такі як перевантаження мережі), система може відкласти прийняття радикальних дій. Навпаки, якщо ймовірність значно зростає, система може реагувати більш агресивно.

*3. Контекстно-орієнтовані порогові значення.* Динамічні пороги дозволяють системі враховувати контекст, такий як історична продуктивність,

стан мережі або важливість вузла, що постраждав. Наприклад, якщо постраждалий вузол має історію стабільної роботи, система може встановити вищий поріг перед тим, як оголосити його несправним. Така контекстна обізнаність допомагає системі уникати хибнопозитивних спрацьовувань, адаптуючи свою реакцію до поточного середовища.

*4. Згладжування порогових значень.* Замість різких змін байєсівські оновлення дозволяють забезпечувати плавні переходи між різними станами прийняття рішень. У міру накопичення подій і зміни апостеріорної ймовірності порогові значення можуть коригуватися поступово, запобігаючи надмірній реакції системи на незначні або тимчасові аномалії.

Розглянемо сценарій, коли затримка в мережі тимчасово збільшується через високий трафік. Статичний тайм-аут у такій ситуації може призвести до хибнопозитивного визначення відмов, що спровокує непотрібні процедури відновлення (failovers). Однак завдяки динамічному пороговому значенню система може налаштовувати пороги виявлення відмовна основі байєсівських оновлень, які враховують поточні умови мережі, дозволяючи системі витримувати тимчасове збільшення затримки без помилкового трактування цього як відмови вузла [42].

Динамічні порогові значення, реалізовані через байєсівські оновлення, додають гнучкість і адаптивність у процес прийняття рішень в алгоритмах консенсусу. Постійно коригуючи порогові значення на основі подій у реальному часі та контексту, система може зменшити кількість хибнопозитивних спрацьовувань і точніше реагувати на реальні збої. Такий підхід робить розподілені сервісні системи більш стійкими та надійними, оскільки система адаптується до змінних умов, а не дотримується жорстких, наперед визначених правил, блок схема методу представелна на рисунку 2.3.

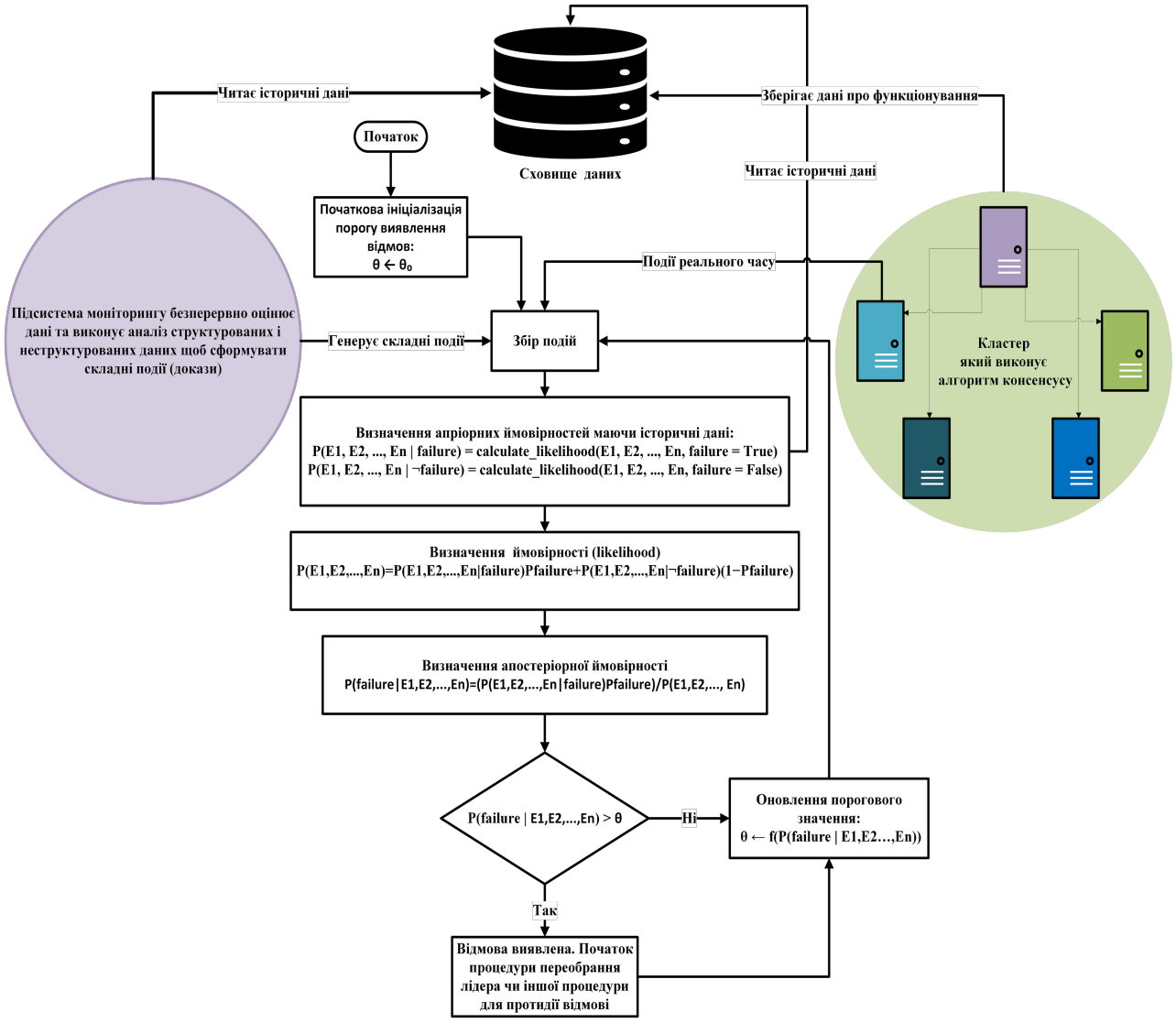



Рис. 2.3. Блок-схема алгоритму виявлення відмов у роботі РСС з використанням теореми Байєса

## 2.2. Зменшення впливу нестабільності мережевих з'єднань на функціонування алгоритмів консистентності

В умовах функціонування мереж чи окремо взятого кластеру мережі, безперебійна робота та ефективність алгоритмів консенсусу можуть суттєво погіршуватися через виклики, пов'язані з виявленням збоїв у мережі чи окремих вузлах. Виявлення цих збоїв зазвичай відбувається, коли вузол перевищує визначений поріг часу очікування на відповідь, що сигналізує про втрату зв'язку з іншим вузлом. У таких випадках запускається процес реконфігурації кластера, спрямований на збереження цілісності його функціонування [10,31,54].

Для мінімізації кількості реконфігурацій пропонуємо проводити часове прогнозування затримки часу відповіді в мережі $T_{prediction-delay-answer}$, та за допомогою функції $f$ визначати поріг виявлення відмов (де $f$ визначається адміністратором сиситеми). Основою прогнозування є статистичний аналіз даних, отриманих під час роботи системи. Кожен вузол у мережі регулярно збирає та аналізує інформацію про затримки, пов'язані з його з'єднаннями з іншими вузлами. Використовуючи історичні дані, вузли обчислюють очікувану затримку через часовий аналіз, що надає їм можливість прогнозувати потенційні відмови. Маючи набір значень прогнозованих затримок, кожне значення якого відповідає певному вузлу, вузол може ухвалити рішення щодо його несправності шляхом коригування порогу $T_{delay}$. Таке коригування має забезпечити виконання умови, наведеної у формулі:

$$T_{delay} < f(T_{prediction-delay-answer}), \tag{2.4}$$

та дозволить зменшити ймовірність помилкових висновків про несправність вузлів.


Рис. 2.4. Блок-схема алгоритму аналізу затримок

Такий підхід (зображений на рисунку 2.4) покращить здатність вузла ухвалювати рішення щодо несправності іншого вузла та дозволить уникнути можливих хибних висновків про його несправність. Це, своєю чергою, зменшить

час, необхідний для переналаштування кластеру та вибору лідера, а також підвищить відмовостійкість кластеру [24].

Обчислення значення $T_{prediction-delay-answer}$, необхідного кожному вузлу, може здійснюватися безпосередньо на самому вузлі (якщо вистачає обчислювальних ресурсів), або ж бути делеговане спеціальному вузлу в кластері, який має такі ресурси. Проте виконати такий часовий аналіз за великої кількості вузлів в кластері доволі складно. Саме тому пропонуємо використання SARIMA для визначення очікуваної затримки в мережах — це дозволить передбачити тривалість встановлення з'єднання між вузлами, оскільки SARIMA не є вимогливою до обчислювальних можливостей, ідея методу представлена на рисунку 2.5.

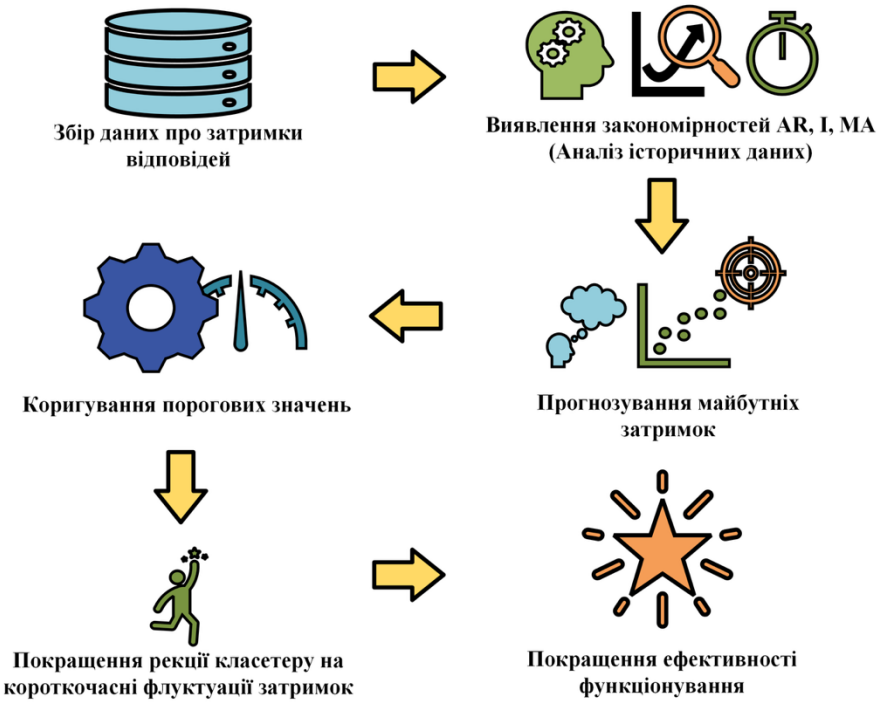



Рис. 2.5. Ідея прогнозування затримки часу відповіді в мережі із використанням моделі ШІ

Завдяки SARIMA вузли отримують можливість передбачати затримки ще до того, як вони стануть критичними, і адаптувати логіку виявлення збоїв.

Впровадження методу базується на такій послідовності дій:

1. Збір даних — кожен вузол у кластері фіксує затримки у зв’язку з іншими вузлами у вигляді часового ряду.
2. Попередня обробка — дані очищаються від аномалій, пропуски заповнюються, серії нормалізуються по часу.
3. Оцінка стаціонарності — застосовуються статистичні тести (наприклад, Дікі–Фуллера); при необхідності ряд інтегрується для досягнення стаціонарності.
4. Визначення параметрів моделі SARIMA — шляхом аналізу автокореляційної та часткової автокореляційної функцій встановлюються значення параметрів *p, d, q* (та їх сезонних аналогів *P, D, Q, m*).
5. Навчання моделі — *SARIMA* навчається на зібраних даних конкретної пари вузлів.
6. Прогноз затримок — модель генерує очікувані затримки на *k* кроків уперед.
7. Інтеграція у логіку виявлення збоїв — отримані прогнози використовуються для адаптивного налаштування часу очікування вузлів на відповідь, або інших параметрів що визначають спрацювання реконфігурації.
8. Оновлення моделі — у міру надходження нових даних модель періодично оновлюється для збереження точності.

Блок схема запропонованого методу представлена на рисинку 2.6.

Таким чином, запропонований підхід дозволяє забезпечити більш гнучке та точне реагування на коливання якості зв’язку в розподілених сервісних системах, зберігаючи при цьому узгодженість і стабільність кластерної структури.

### 2.2.1 Адаптивне визначення порогу відмов за рахунок впровадження моделі машинного навчання SARIMA

Навіть у великому та складному кластері проблему виявлення відмов можна звести до аналізу взаємодії вузлів. Для спрощення розглянемо невеликий кластер, що складається з $N$ вузлів, і проаналізуємо історію значень затримок між конкретною парою вузлів.

Щоб описати залежність поточного значення затримки у певний момент часу, між двома пристроями розподіленої системи, від одного або кількох попередніх значень, використовуватимемо авторегресійну ( $AR$ ) модель.

Авторегресійний компонент моделі $SARIMA$ представимо формулою 2.5, де параметр $p$ визначає кількість використаних лагів (попередніх значень) [127]. Формула має вигляд:

$$Y_t = c + \sum_{n=1}^{p} a_n y_{n-1} + e_t \tag{2.5}$$

Вищий параметр $p$ означає більший часовий інтервал. Авторегресійний процес $AR(1)$ — це процес, у якому поточне значення ґрунтується на безпосередньо попередньому значенні, тоді як процес $AR(2)$ — це процес, у якому поточне значення залежить від двох попередніх значень. Процес $AR(0)$ використовується для білого шуму і не має залежностей між членами ряду. Крім того, введемо в модель процес ковзного середнього порядку $q$, що позначимо як $MA(q)$ і описується формулою:

$$Y_t = c + \sum_{n=1}^{p} a_n y_{n-t} + e_t \tag{2.6}$$

Щоб описати залежність поточного значення затримки у заданий момент часу між двома пристроями від одного або кількох попередніх помилок прогнозу, використовується модель ковзного середнього ( $MA$ ).

Замість використання минулих значень змінної в регресійному прогнозуванні, модель ковзного середнього використовує попередні помилки прогнозу. У часових рядах модель ковзного середнього ($MA$), також відома як процес ковзного середнього, є поширеним підходом для моделювання однофакторних часових рядів. Модель ковзного середнього визначає, що вихідна змінна має взаємозв'язок із неідентичною випадковою змінною.

У статистичному аналізі часових рядів моделі авторегресії та ковзного середнього ($ARMA$) забезпечують економічний опис слабко стаціонарних стохастичних процесів у термінах двох поліномів: одного для авторегресії ($AR$) та іншого для ковзного середнього ($MA$).

Компонент $I$ (інтегрований) у свою чергу представляє різницю між необробленими спостереженнями, що робить часовий ряд стаціонарним (тобто дані замінюються різницею між їхніми значеннями та попередніми значеннями).

Властивість стаціонарності відіграє важливу роль, оскільки стаціонарний часовий ряд має властивості, що не залежать від часу спостереження. Таким чином, часові ряди з трендами або сезонністю не є стаціонарними – тренд і сезонність впливатимуть на значення часового ряду в різні моменти часу. Обробка таких даних дозволить забезпечити точніші прогнози значень.

Якщо значення затримки між двома вузлами демонструють чіткий сезонний патерн або сезонність (у більшості кластерів це має місце через особливості використання, наприклад, кластер активно використовується у будні дні), необхідно включити сезонний компонент у модель. Якщо значення затримки між двома вузлами не демонструють чіткого сезонного патерну або сезонності, то модель $SARIMA$ може бути непридатною для цього конкретного набору даних. Модель $SARIMA$ зазвичай використовується для роботи з часовими рядами, що демонструють повторювані патерни протягом фіксованих інтервалів.

Модель $SARIMA$ розширює модель $ARIMA$, додаючи до неї сезонність. Такі моделі записуються у вигляді $(p,d,q)\times(P,D,Q)_m$, де $(p,d,q)$ — це параметри для моделі $ARIMA$, а $(P,D,Q)_m$ представляють сезонні компоненти авторегресії,

інтеграції та ковзного середнього, причому період сезонності дорівнює $m$. Блок-схема визначення порогу відмов шляхом прогнозування часу очікування повідомлень за допомогою моделі машинного навчання SARIMA представлена на рисунку 2.6.

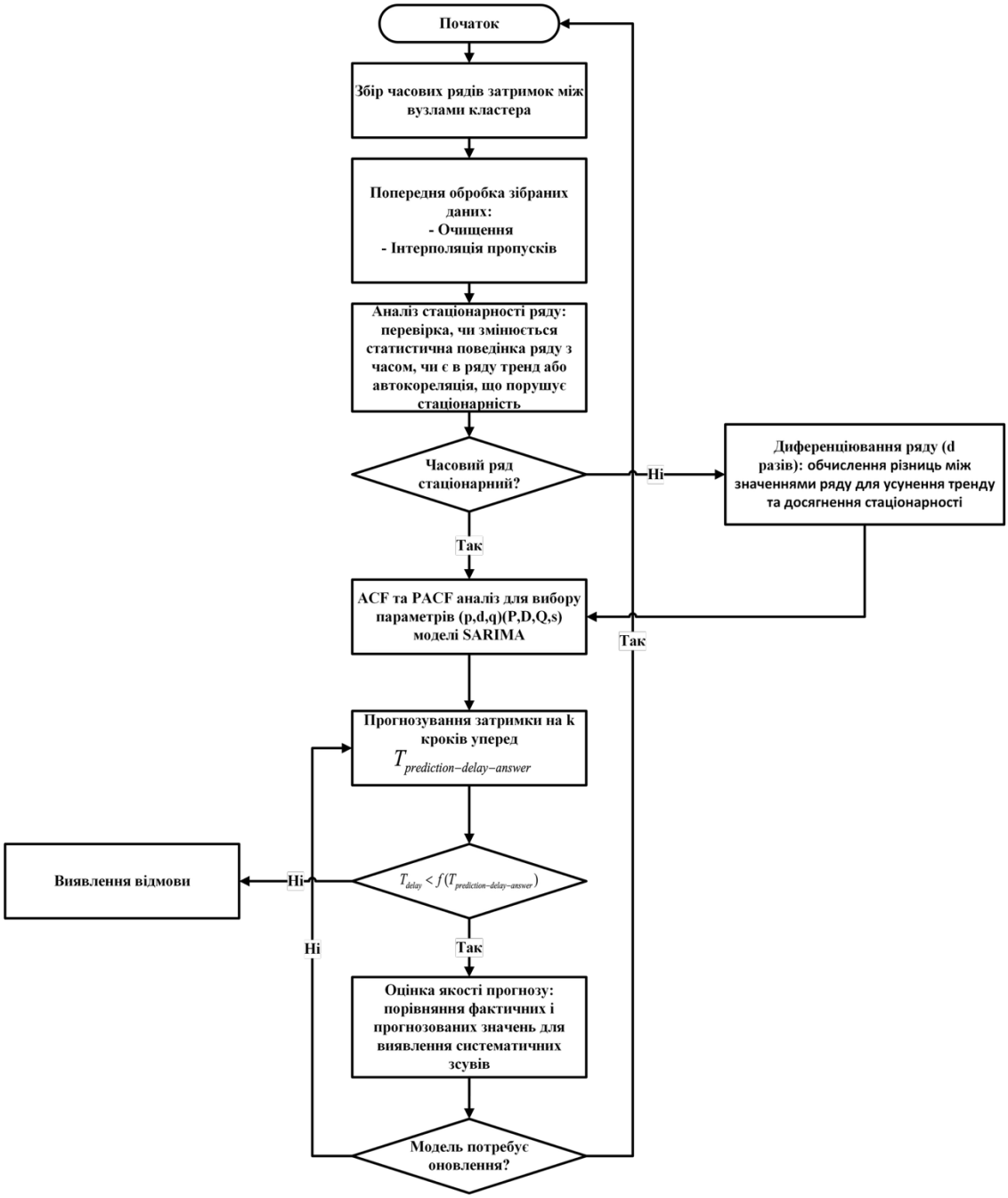



Рис. 2.6. Блок-схема алгоритму визначення порогу відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA

## 2.3. Розроблення автономного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем

У динамічному середовищі розподілених обчислень здатність вузлів мережі досягати консенсусу є фундаментальною для забезпечення цілісності та ефективності розподілених систем. Ці системи лежать в основі низки критичних застосувань — від блокчейн-технологій до розподілених баз даних, де узгодженість спільних даних має першочергове значення [1-2, 9-10]. Проте підтримка цієї узгодженості стає особливо складною в умовах нестабільних мережевих з’єднань — поширеної, але водночас серйозної проблеми, що може спричинити неконсистентність даних, зниження продуктивності системи та втрату її надійності [13-14, 43].

У контексті алгоритмів консенсусу, заснованих на лідері, таких як Raft, ідеальний сценарій є простим: найстійкіший кластер — це той, у якому лідер підтримує найбільш надійні мережеві з'єднання зі своїми підлеглими. Це передбачає мінімальні затримки між лідером і його послідовниками. Враховуючи принципи роботи алгоритму Raft, стає очевидним, що Raft не розроблений для визначення найоптимальнішого лідера в кластері; його основна мета полягає у вирішенні збоїв у системі та забезпеченні спільної роботи кластера, без особливого акценту на ефективність.

Проте такі алгоритми, як Raft, не мають вбудованих механізмів для вирішення реальних викликів, коли нестабільність мережі призводить до «замороженого» стану кластера через постійне переобрання лідера, викликане кількома проблемними мережевими з’єднаннями.

Вибір оптимального лідера в кластері є надзвичайно важливим для забезпечення ефективного управління даними та функціональності системи. Це вимагає детального аналізу атрибутів і умов, які сприяють вибору найбільш підходящого лідера серед вузлів.

У дискурсі про розподілені сервісні системи, особливо ті, що залежать від алгоритмів консенсусу, питання визначення найбільш оптимального лідера в

будь-який момент часу є ключовим. Це питання за своєю суттю є тимчасовим і залежить від поточної стабільності мережі, яка може змінюватися з часом. Таким чином, твердження про перевагу певного вузла як лідера може бути точним лише в контексті конкретного моменту часу, враховуючи швидкі зміни умов у мережі.

Для вирішення проблеми переобрання лідера розглянемо випадок. Нехай, маємо два вузли у гіпотетичному кластері мережі (рисунок 2.7).

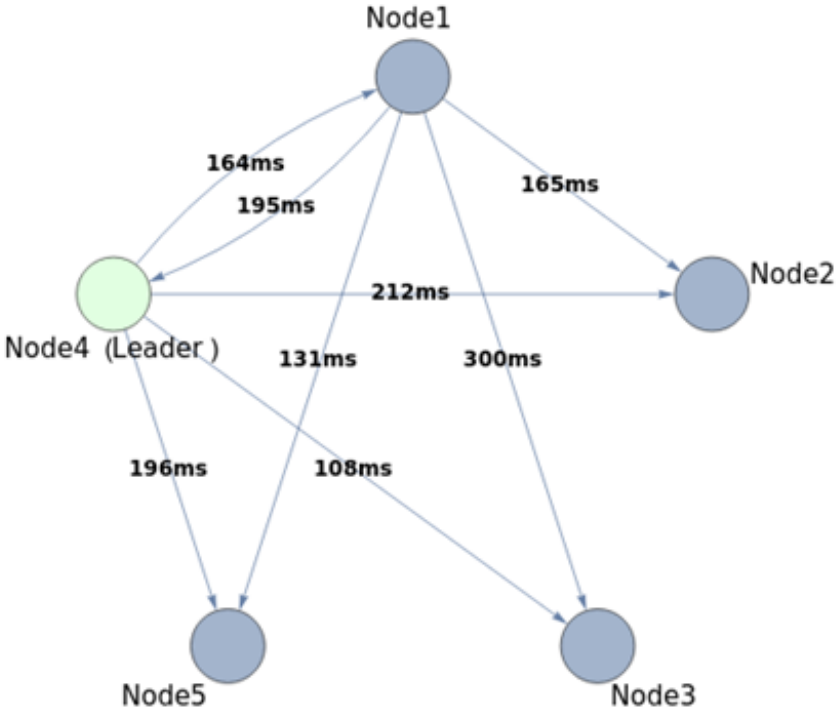



Рис. 2.7. Карта затримок у кластері для вибору лідера

Припустимо, що вузол 4 - чинний лідер, а вузол 1 - потенційний кандидат на роль лідера. Визначення його відповідності в порівнянні з вузлом 4 передбачає оцінку затримок у мережі до всіх інших вузлів у кластері. Цей процес зводиться до порівняльного аналізу затримок у каналах, що виникають від обох вузлів до всіх інших вузлів у мережі. Такий підхід підкреслює динамічність ефективності лідерства у розподілених системах, де оптимальний лідер не є статично визначеним, а відображає поточний, динаміно змінний стан роботи мережі.

Крізь цю призму пропонуємо обирати лідера, із адаптацією під динаміно змінний характер мережі. Щоб математично формалізувати цю концепцію, можна позначити $L(1, j)$ як затримку між вузлом $i$ та вузлом $j$. Для вузла 1,

який претендує на лідерство, його загальну затримку $T_1$ можна виразити як суму затримок між вузлом 1 та всіма іншими $n-1$ вузлами в мережі:

$$T_1 = \sum_{j=2}^{n} L(1, j). \tag{2.7}$$

Аналогічно, поточний лідер, вузол 4, має відповідну загальну затримку $T_4$

$$T_4 = \sum_{j=1, j\neq 4}^{n} L(4, j). \tag{2.8}$$

У цьому контексті вузол 1 буде більш підходящим лідером, якщо $T_1$ менше ніж $T_4$, що вказує на те, що вузол 1 має нижчу загальну затримку в комунікації з рештою мережі:

$$\text{Вузол 1 є більш стабільним якщо } T_1 < T_4. \tag{2.9}$$

Згідно з припущенням, вузол із меншою загальною затримкою є більш підходящим лідером. Порівнюючи $T_1$ та $T_4$, отримаємо:

$$T_1(835мс) > T_4(676мс). \tag{2.10}$$

Таким чином, вузол 4 із загальною затримкою 676 мс є більш підходящим лідером, ніж вузол 1 із загальною затримкою 835 мс, відповідно до заданої конфігурації.

Ця оцінка передбачає, що загалом нижча затримка безпосередньо корелює з кращою продуктивністю та придатністю для лідерства. Однак, при цьому, не враховуються сценарії, коли одне з'єднання має значно вищу затримку, ніж інші, що відіграє критичну роль у комплексній оцінці.

Покращимо надійність з'єднань від вузла 1 до інших вузлів, за винятком одного з'єднання, яке демонструє значно вищу затримку (див. рисунок 2.8).

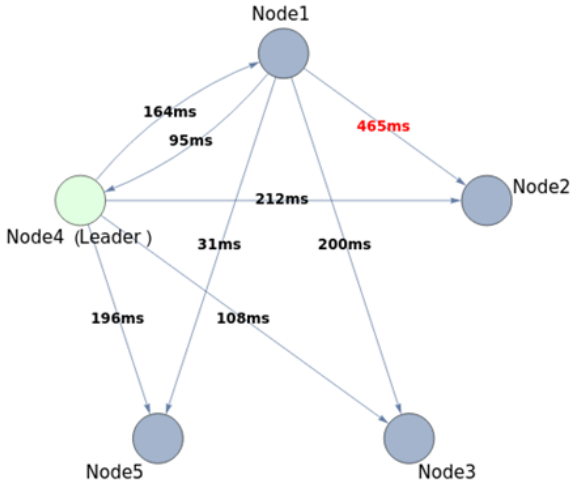



Рис. 2.8. Карта затримок у кластері з аномальним з'єднанням

Використання запропонованого вище підходу для визначення найбільш підходящого лідера може спочатку вказати на вузол 4 як на переважного кандидата. Однак така оцінка буде неправильною. Фактична продуктивність кластера фундаментально залежить від консенсусу більшості у кластері. Алгоритм консенсусу, заснований на лідері, вимагає, щоб більшість вузлів була працездатною та могла спілкуватися, щоб забезпечити прогрес. Більш конкретно, більшість можна виразити як:

$$M = \left\lfloor \frac{n}{2} \right\rfloor + 1 \qquad (2.11)$$

де $n$ – позначає кількість вузлів в кластреі, а дужки позначають операцію цілочисельного округлення вниз, яка округлює число до найбільшого цілого числа, меншого або рівного даному числу. Це забезпечує, що більшість завжди перевищує половину від загальної кількості вузлів.

Таким чином, визначення найбільш відповідного лідера потребує врахування принципу більшості у кластері. Запропонований метод визначає стратегію ідентифікації оптимального лідера в мережі, беручи до уваги правило більшості в кластері. Вона наголошує на виборі вузла, який використовує найшвидші канали зв’язку в межах кластера.

Першим кроком є визначення найменшої кількості вузлів, необхідної для формування більшості, яка керуватиме процесом прийняття рішень. Після цього

вузли впорядковуються за швидкістю їхньої комунікації — від найефективніших до найменшефективних. У межах цієї пріоритетної групи стратегія передбачає обчислення середньої швидкості, визначення медіани або ідентифікацію найповільнішого з'єднання. Завдяки цій оцінці обирається вузол, який найкраще підходить для виконання ролі лідера, щоб прискорити прийняття рішень і підвищити загальну ефективність роботи кластера.

Математичние представлення процесу ідентифікації найкращого лідера в кластері включає кілька кроків, його можна представити таким чином:

1. Обчислити більшість ($M$): Більшість ($M$), у свою чергу, визначається формулою. Якщо загальна кількість вузлів у кластері дорівнює $n$, тоді $M$. обчислюється за формулою 2.11.

2. Відсортувати затримки: Нехай $L = \{l_1, l_2, \ldots, l_n\}$ представляє множину затримок від одного вузла до всіх інших вузлів у кластері. Затримки в $L$ сортуються в порядку зростання для отримання $L_{sorted} = \{l_{(1)}, l_{(2)}, \ldots, l_{(n)}\}$, де $l_{(1)} \le l_{(2)} \le \ldots \le l_{(n)}$.

3. Вибрати перші затримок: Із відсортованої множини $L_{sorted}$ вибираються перші $M$ затримок, формуючи підмножину $L_M = \{l_{(1)}, l_{(2)}, \ldots, l_{(M)}\}$.

4. Обчислити метрику для визначення найкращого лідера: Використовується середнє значення, медіана або найповільніше з'єднання в межах кворуму (більшості).

Опишемо ці метрики математично. Середнє значення визначається за формулою:

$$\overline{l} = \frac{1}{M} \sum_{i=1}^{M} l_{(i)}. \qquad (2.12)$$

Медіана визначається як середнє значення у множині $L_M$, якщо $M$ непарне, або як середнє двох середніх значень, якщо $M$ парне:

$$M = l_{\left(\frac{M+1}{2}\right)} \left( якшо\, M\ \ непарне \right), \tag{2.13}$$

або

$$M = \frac{l_{\left(\frac{M}{2}\right)} + l_{\left(\frac{M}{2}+1\right)}}{2} \left( якшо\, M\ \ парне \right). \tag{2.14}$$

Найповільніше з'єднання в межах більшості визначаєть як:

$$\max(L_M) = \max(\{l_1, l_2, ..., l_M\}) \tag{2.15}$$

Ці значення слугують метрикою для оцінки придатності вузла на роль лідера, враховуючи його продуктивність у комунікації із більшістю вузлів у кластері.

Вибір оптимальної метрики для вибору лідера повинен ретельно враховувати вплив лідера на загальну пропускну здатність системи. У середовищах, де швидкість прийняття рішень є критично важливою, наприклад, у системах реального часу, де будь-яка затримка в комунікації вузлів може мати серйозні наслідки, орієнтація на найповільніше з'єднання може бути найбільш доцільним підходом. Ця метрика є особливо важливою, оскільки вона визначає граничний рівень продуктивності для всього кворуму; система не може працювати швидше, ніж її найповільніший вузол-учасник.

Отже, найповільніше з'єднання фактично задає темп процесу прийняття рішень більшістю. Для того, щоб обрана метрика відповідала вимогам системи та реальним умовам експлуатації, необхідно базувати вибір на емпіричних даних із реальних експериментів, які надають відчутний вимір продуктивності системи за типовими навантаженнями та умовами експлуатації. Це гарантує, що рішення приймаються лише тоді, коли вся більшість завершила свої завдання.

Зважаючи на ці міркування, підхід до вибору лідера у розподілених системах переосмислюється через алгоритм, орієнтований на кворум, який надає пріоритет стабільності найповільнішого з'єднання в мережі. Це забезпечує, що

обраний лідер є найбільш здатним підтримувати цілісність системи в періоди змін у мережевих затримках.

Таким чином, пропонуємо алгоритм, який не лише оцінює загальну затримку між потенційним лідером та іншими вузлами, але й враховує максимальну затримку, що впливає на більшість кластера. Його блок-схема наведена на рисунку 2.9 а псевдокод на рисунку 2.10 відповідно.

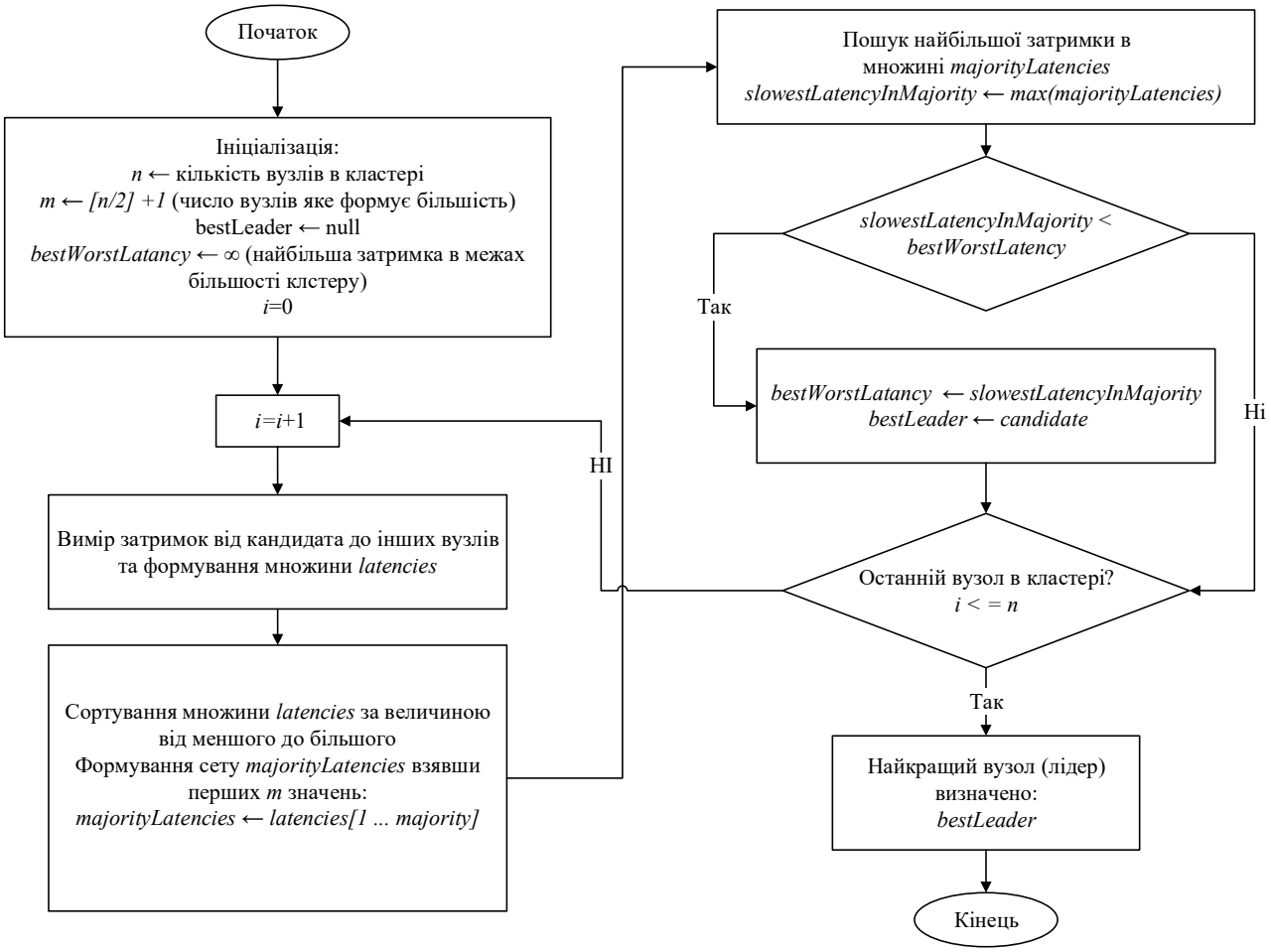



Рис. 2.9. Блок-схема алгоритму вибору лідера на основі найповільнішого з’єднання в межах кворуму

**Algorithm 1** Leader Selection Based on Slowest Link in Quorum

```
procedure FindBestLeaderBasedOnSlowestLink(nodes)
    n ← length(nodes)
    majority ← ⌊n/2⌋ + 1
    bestLeader ← null
    bestWorstLatency ← ∞
    for all candidate ∈ nodes do
        latencies ← GetLatenciesFromCandidateToOthers(candidate, nodes)
        sortedLatencies ← SortLatenciesAscending(latencies)
        majorityLatencies ← sortedLatencies[1 ... majority]
        slowestLatencyInMajority ← FindMaxLatency(majorityLatencies)
        if slowestLatencyInMajority < bestWorstLatency then
            bestWorstLatency ← slowestLatencyInMajority
            bestLeader ← candidate
        end if
    end for
    return bestLeader
end procedure
function GetLatenciesFromCandidateToOthers(candidate, nodes)
    // Retrieves all latencies from candidate to other nodes
    // Returns a list of latencies: [l1, l2, ..., ln]
end function
function SortLatenciesAscending(latencies)
    // Sorts latencies in ascending order
    // Returns the sorted list of latencies
end function
function FindMaxLatency(latencies)
    // Finds the maximum latency in the given list
    // Returns the maximum latency
end function
```

Рис. 2.10. Псевдокод алгоритму вибору лідера на основі найповільнішого з’єднання в межах кворуму

Опишемо процедури, які дозволять плавно інтегрувати процес вибору лідера в існуючу структуру кластера.

*Процес вибору*

Після обрання новий лідер починає передавати нові сигнали працездатності (heardbeat), щоб підтвердити свій контроль і забезпечити узгодженість журналу (log) у кластері. Цей механізм дозволяє системі швидко відновлюватися після збоїв лідера, підвищуючи її стійкість і забезпечуючи безперервну роботу.

Заздалегідь визначений час очікування для цих сигналів працездатності (heardbeat) задається випадковим чином у межах певного діапазону, наприклад, від 100 до 300 мілісекунд. Це забезпечує, що хоча б один вузол отримає тайм-аут

раніше за інші, сприяючи плавному переходу до нового лідера і фактично надаючи одному вузлу перевагу в часі для початку виборчого процесу.

Проблема виникає через те, що вибір нового лідера фактично є випадковим, оскільки тайм-аут для сигналів працездатності (heardbeat) встановлюється випадковим чином, тим самим ненавмисно надаючи одному вузлу перевагу. Такий підхід, який використовує випадкову затримку для вибору лідера, спрощує алгоритм Raft, роблячи його простим для реалізації. Тому зберегти концепцію тайм-аутів є доцільним.

Щоб удосконалити цей процес, стратегічно доцільно налаштувати систему так, щоб найбільш відповідний вузол мав коротший тайм-аут, що дозволить йому швидше взяти лідерство порівняно з іншими. Першим кроком є встановлення процесу, у межах якого всі вузли оцінюють свої затримки до всіх інших вузлів. Це вимірювання є ключовим для оцінки їхньої відповідності для виконання ролі лідера.

Кожен вузол $N_i$ періодично оцінює затримку $D_k(i,j)$ до кожного іншого вузла за розкладом, визначеним інтервалом $\Delta t$, заданим адміністратором. Це вимірювання затримок виконується у моменти часу $t_k = k\Delta t$ для кожного послідовного інтервалу $k = 1,2,3,\ldots$ Така конфігурація забезпечує, що вузли систематично та послідовно вимірюють і оновлюють свої метрики міжвузлових затримок відповідно до періодичності, встановленої $\Delta t$, сприяючи ефективному моніторингу та управлінню мережею.

Важливо, щоб адміністратор ретельно обирав інтервал $\Delta t$, враховуючи обчислювальні можливості кластера та поточний стан мережі. Оскільки створення запитів опитування (ping) є обчислювально недорогим, але все ж споживає ресурси процесора та збільшує трафік у мережі, $\Delta t$ слід оптимізувати для балансу між частотою вимірювань та можливим впливом на продуктивність системи. Надто часті вимірювання можуть надмірно навантажити мережу, тоді як занадто рідкісні можуть не фіксувати значних змін у затримках вузлів учасників кластера своєчасно.

Тому $\Delta t$ має бути вибрано розумно, щоб забезпечити ефективний моніторинг мережі без шкоди для загальної функціональності та чутливості кластера.

Для відносно стабільних мереж час тайм-ауту $\Delta t$ можна встановити на більш тривалі інтервали, щоб уникнути надто частого виклику оцінок. Крім того, цей процес можна вдосконалити, призначивши кожному вузлу унікальний зсув у розкладі оцінки, щоб уникнути одночасних вимірювань на всіх вузлах, та відповідно, зменшити ризик перевантаження мережі в один момент часу.

Щоб описати процес визначення значення тайм-ауту в алгоритмі консенсусу за умов змінної мережі, стає очевидним, що, хоча цей тайм-аут має враховувати затримку на найповільнішому з'єднанні в більшості, позначену як $max(L_M)$, його не слід встановлювати лише на цій основі. Такий підхід є недостатнім, оскільки точно розрахований тайм-аут також має бути обмежений певним максимальним значенням $T_{\max}$, щоб гарантувати реконфігурацію системи у разі фактичного збою вузла, а не лише через тимчасові затримки.

Це максимальне значення, $T_{\max}$, найкраще встановлюється адміністратором, який розуміє специфічні вимоги до використання та обмеження системи.

Формула для встановлення адаптивного тайм-ауту $T$, на даному етапі, може бути виражена наступним чином:

$$T = \left(max\left(L_M\right)/100\right) \times T_{\max}. \qquad (2.16)$$

Щоб вирішити проблему потенційно одночасних тайм-аутів у кількох вузлах кластера, коли затримки в мережі розподілені рівномірно, дійсно важливо додати елемент випадковості до налаштування тайм-ауту, як це запропоновано в протоколі консенсусу Raft. Ця випадковість допомагає зменшити ймовірність того, що всі вузли одночасно ініціюватимуть вибори лідера через тайм-аут у той самий момент. Модифікуємо формулу, щоб включити випадкову складову, зберігаючи при цьому основні принципи початкової конфігурації.

$$T_i = \left( \frac{max\left(L_M\right)_i + rand\left(a,b\right)}{100} \right) \times T_{max}, \tag{2.17}$$

де $rand\left(a,b\right)$ генерує випадкове число в діапазоні між $a$ та $b$ (значення $a$ та $b$ задаються адміністратором).

Розглянемо наступний сценарій із кластером із 5 вузлів (рисунок 2.11, де інші вузли припущені, але не показані). Припустимо, що вузол 2 краще підходить для лідерства, враховуючи його затримки, а максимальна затримка в межах кворуму менша, ніж у вузла 3: $max\left(L_M\right)_2 < max\left(L_M\right)_3$. Зважаючи на затримки в мережі, існує ймовірність, що вузол 3 досягне свого тайм-ауту раніше, ніж повідомлення від вузла 2 прибуде, що спонукатиме вузол 3 ініціювати нові вибори лідера.

Хоча вузол 2 має кращий потенціал для перемоги на виборах, цей сценарій підкреслює необхідність подальшого вдосконалення.

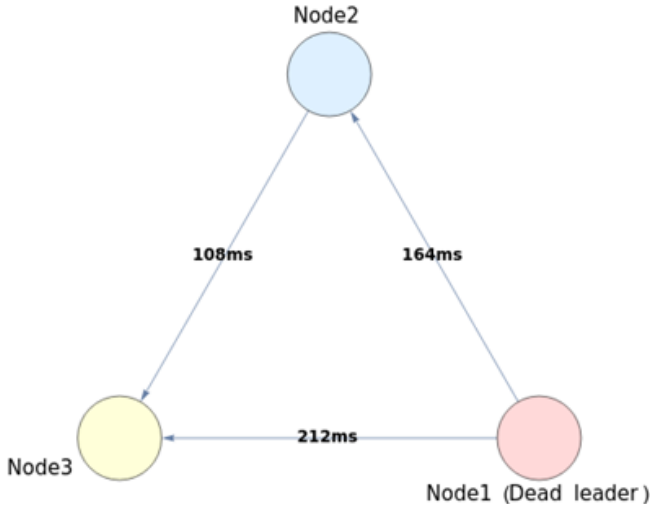



Рис. 2.11. Візуалізація затримок комунікації в кластері: карта затримок між вузлами

Опишемо спочатку кожен вузол має доступ до значень максимальної затримки інших вузлів $max(L_M)$, представлених у вигляді множини $\Theta$:

$$\Theta = \left\{ max(L_M)_1,\ \ max(L_M)_2,\ \ \ldots\ \ max(L_M)_{n-1} \right\}. \tag{2.18}$$

Кожен вузол може самостійно оцінити, чи є він найбільш підходящим кандидатом на роль лідера, використовуючи умову:

$$max(L_M)_i = min(\Theta) \tag{2.18}$$

Якщо вузол визначить, що він не є оптимальним вибором для лідерства, важливо, щоб він розрахував час очікування таким чином, щоб затримки не мали негативного впливу на процес виборів. Таким чином, вузол обчислює свій час очікування за формулою:

$$T_i = max\left(\left(\frac{max(L_M)_i}{max(\Theta)}\right) \times T_{\max}\ ,\ T_{\max} \times (1 - \partial_{me}\right) + rand(a,b) + \\ + \min\left(T_{dlbc} + T_{bccc} - T_{dlcc}\ ,\ \delta_c\right) \tag{2.20}$$

де $max(L_M)_i$ — затримка на найповільнішому з'єднанні в межах більшості для кандидата $i$; $rand(a,b)$ — функція, що вводить випадковість у межах діапазону $[a,b]$, заданого адміністратором; $T_{\max}$ — максимальне значення тайм-ауту, встановлене адміністратором; $\delta_c$ — значення в діапазоні від 0 до 1, яке вказує частку $T_{\max}$, що використовується як мінімальне значення $T_i$; $T_{dlbc}$ — тайм-аут від передбачуваного померлого лідера до найкращого кандидата; $T_{bccc}$ — тайм-аут від найкращого кандидата до поточного кандидата; $T_{dlcc}$ — тайм-аут від передбачуваного померлого лідера до поточного кандидата.

Вузол, який вважає себе оптимальним лідером, не використовуватиме другу частину формули. Таким чином, вираз, що охоплює обидва сценарії, виглядатиме наступним чином:

$$T_i = \begin{cases} max\left(\left(\frac{max(L_M)_i}{max(\Theta)}\right) \times T_{\max}\ ,\ T_{\max} \times (1 - \partial_{me}\right)) + rand(a,b) + \min\left(T_{dlbc} + T_{bccc} - T_{dlcc}\ ,\ \delta_c\right) & if\ max(L_M)_i \neq min(\Theta) \\ max\left(\left(\frac{max(L_M)_i}{max(\Theta)}\right) \times T_{\max}, T_{\max} \times \partial_{me}\right) + rand(a,b) & if\ max(L_M)_i = min(\Theta) \end{cases}. \tag{2.19}$$

Підсумовуючи, метод можна описати процесами обміну та обробки повідомленнями які необхідні для збору даних про затримки в кластері (рисунок 2.12) і процес розрахунку вузлом часу очікування на відповідь $T$.

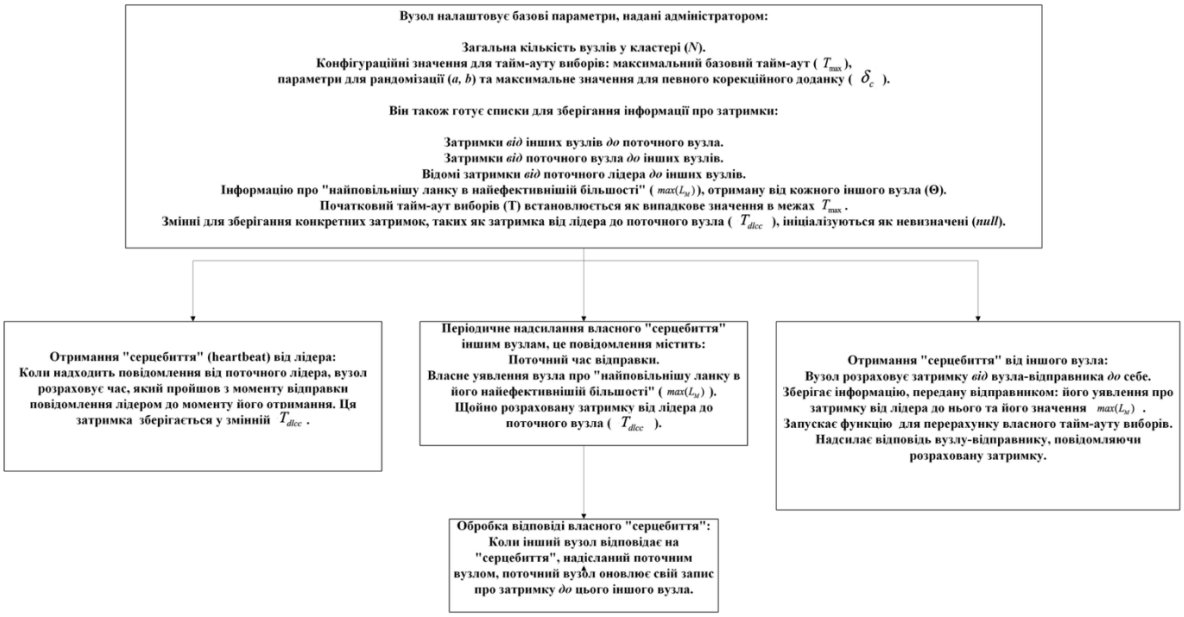



Рис. 2.12. Блок-схема яка описує роботу процесів heardbeat

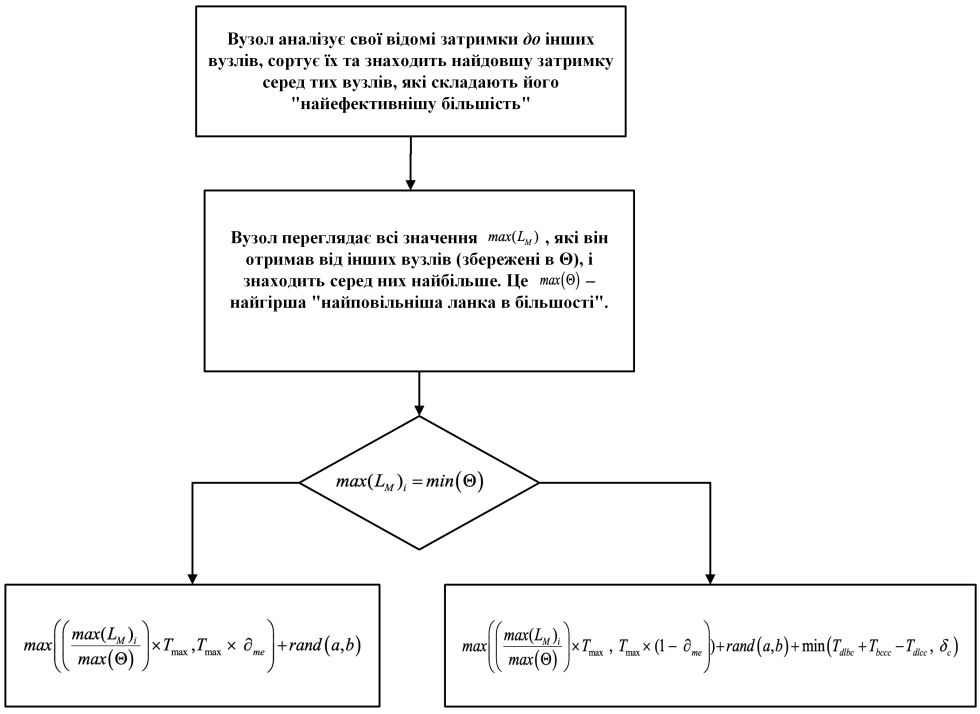



Рис. 2.13. Блок-схема розрахунку часу очікування на відповідь для вузла-послідовника

**Algorithm 1** Follower Heartbeat and Timeout Adjustment Process

$N \leftarrow$ provided by administrator ▷ Number of nodes within the cluster
$T_{\text{dlbc}}, T_{\text{bccc}}, T_{\text{dlcc}} \leftarrow$ null, null, null ▷ Delays between respective nodes
$T_{\max}, a, b, \delta_c, \partial_{me} \leftarrow$ administrator provided values ▷ Configuration values for leader election timeout, where $\partial_{me}$ is a value such that $0 \leq \partial_{me} \leq 1$
Nodes $\leftarrow \{x \in \mathbb{Z} \mid 1 \leq x \leq N-1\}$ ▷ Other nodes within the cluster
$L_{\text{latencies-from-node}} \leftarrow \{(node, \tau) \mid node \in \text{Nodes}, \tau \leftarrow \text{null}\}$ ▷ Node and delay-from-node pairs set
$L_{\text{latencies-to-node}} \leftarrow \{(node, \tau) \mid node \in \text{Nodes}, \tau \leftarrow \text{null}\}$ ▷ Node and delay-to-node pairs set
$L_{\text{leader-to-node}} \leftarrow \{(node, \tau) \mid node \in \text{Nodes}, \tau \leftarrow \text{null}\}$ ▷ Node and leader-to-node-delay pairs set
$maxL_M \leftarrow$ null ▷ The slowest link delay within the nodes' most efficient majority
$\Theta \leftarrow \{(node, maxL_M) \mid node \in \text{Nodes}, maxL_M \leftarrow \text{null}\}$ ▷ Node and slowest-link-delay-within-majority pairs set
$T \leftarrow \text{RANDOM}(T_{\max})$ ▷ Leader election timeout
**procedure** RECEIVELEADERSHEARTBEAT($message$)
    $T_{\text{dlcc}} \leftarrow \text{GETCURRENTTIME} - message.currentTime$
**end procedure**
**procedure** SENDFOLLOWERHEARTBEAT($node$)
    $currentTime \leftarrow \text{GETCURRENTTIME}$
    $latencyFromLeader \leftarrow T_{\text{dlcc}}$
    **for all** $otherNode \in$ Nodes **do**
        SENDMESSAGE(otherNode, $\{currentTime, maxL_M, latencyFromLeader\}$)
    **end for**
**end procedure**
**procedure** PROCESSFOLLOWERHEARTBEATRESPONSE($node$, $message$)
    $L_{\text{latencies-to-node}}[node] \leftarrow message.delayFromSender$
**end procedure**
**procedure** RECEIVEFOLLOWERHEARTBEAT($node$, $message$)
    $receivedAt \leftarrow \text{GETCURRENTTIME}$
    $delayFromSender \leftarrow receivedAt - message.currentTime$
    $L_{\text{latencies-from-node}}[node] \leftarrow delayFromSender$
    $L_{\text{latencies-from-leader}}[node] \leftarrow message.latencyFromLeader$
    $\Theta[node] \leftarrow message.maxL_M$
    CALCULATETIMEOUT
    SENDMESSAGE(node, $\{delayFromSender\}$)
**end procedure**
**function** CALCULATETIMEOUT
    $maxL_M \leftarrow \max\left\{\tau \mid (node, \tau) \in \text{sort}\left(L_{\text{latencies-to-node}}\right)\left[:\left\lceil\frac{|L_{\text{latencies-to-node}}|}{2}\right\rceil\right]\right\}$
    $max\Theta_M \leftarrow \left(\arg\max_{(node, maxL_M) \in \Theta} maxL_M\right).maxL_M$
    $(node, maxL_M) \leftarrow \arg\min_{(node, maxL_M) \in \Theta} maxL_M$
    $min\Theta_M \leftarrow (node, maxL_M).maxL_M$
    **if** $min\Theta_M = maxL_M$ **then**
        $T \leftarrow \max\left(\left(\frac{maxL_M}{max\Theta_M}\right) \times T_{\max}, T_{\max} \times \partial_{me}\right) + \text{RAND}(a, b)$
    **else**
        $bestCandidate \leftarrow (node, maxL_M).\text{node}$
        $\text{T}_{\text{dlbc}} \leftarrow L_{\text{leader-to-node}}[bestCandidate]$
        $\text{T}_{\text{bccc}} \leftarrow L_{\text{leader-from-node}}[bestCandidate]$
        $T \leftarrow \max\left(\left(\frac{maxL_M}{max\Theta_M}\right) \times T_{\max}, T_{\max} \times (1 - \partial_{me})\right) + \text{RAND}(a, b) + \min(T_{\text{dlbc}} + T_{\text{bccc}} - T_{\text{dlcc}}, \delta_c)$
    **end if**
**end function**

Рис. 2.14. Псевдокод який описує роботу процесів heardbeat та розрахунку часу очікування на відповідь вузла-послідовника

## 2.4. Висновки до другого розділу

1. У другому розділі увагу зосереджено на вирішенні проблеми підвищення ефективності фунціонування алгоритмів консенсусу, особливо в умовах нестабільності функціонування мережевих з'єднань в розподілених системах, шляхом розроблення нових методів та алгоритів. Дослідження показало, що нестабільність мережі, така як затримки, втрата пакетів і переривання зв'язку, має суттєвий вплив на узгодженість даних та ефективність роботи системи. Хибні спрацьовування, викликані короткочасними мережевими збоями, часто призводять до помилкових переобрань лідера, додаткових витрат ресурсів та загального зниження продуктивності.

2. Як наслідок, набув подальшого розвитку метод виявлення відмов в роботі розподілених сервісних систем, який на відміну від існуючих аналізує стан функціонування системи та мінімізує кількість помилкових рішень про відмови, як викликані короткочасними мережевими збоями, шляхом інтеграції ймовірнісного байєсівського підходу. Використання теореми Байєса для аналізу наявності збоїв, дозволяє системі приймати більш обґрунтовані, ймовірнісні рішення щодо збоїв вузлів. Динамічно оновлюючи ймовірності збоїв, система може зменшити кількість хибнопозитивних спрацьовувань, тим самим підвищуючи надійність алгоритму консенсусу та дасть змогу забезпечувати стабільне функціонування розподілених сервісних систем.

3. Важливу роль у стабільності функціонування системи та роботі алгоритмів консенсусу, відіграють налаштування параметрів часу очікування вузлом на відповідь та інтервалу повідомлень працездатності (heardbeat). Надто короткий час очікування збільшують чутливість до короткочасних флуктуацій мережевої затримки та збільшують кількість реконфігурацій кластеру, тоді як надто тривалі – можуть затримувати реакцію на реальні збої. Для мінімізації кількості реконфігурацій удосконалено метод адаптивного визначення порогу відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA.

Суть даного методу полягає у часовому прогнозуванні затримки часу відповіді в мережі, та визначенні порогу виявлення відмов. Основою прогнозування є статистичний аналіз даних, отриманих під час роботи системи. Кожен вузол у мережі регулярно збирає та аналізує інформацію про затримки, пов’язані з його з’єднаннями з іншими вузлами. Використовуючи історичні дані та модель машинного навчання SARIMA, вузли обчислюють очікувану затримку через часовий аналіз, що надає їм можливість прогнозувати потенційні відмови. Маючи набір значень прогнозованих затримок, кожне значення якого відповідає певному вузлу, вузол може ухвалити рішення щодо його несправності шляхом коригування порогу тайм-ауту.

4. Для пришвидшення процесу коригування порогу тайм-ауту вперше запропоновано автономний адаптивний метод підвищення продуктивності РСС, який на основі знань про лідер-орієнтовану структуру алгоритму консенсусу, дозволяє динамічно коригувати параметри затримки між вузлами кластеру, обирати лідера в кластері, та забезпечити максимальну стабільність і продуктивність роботи системи вцілому. Ключовою особливістю методу є використання мережевої затримки як основного критерію для визначення пріоритетності вузлів на роль лідера. Кожен вузол аналізує час затримки до інших вузлів більшості (кворуму), оцінює свою потенційну придатність до виконання ролі лідера та, залежно від цього, адаптує значення часу очікування на повідомлення працездатності (heardbeat). Таким чином, вузли з гіршими мережевими характеристиками штучно сповільнюють своє ініціювання виборів, поступаючись вузлам із кращими умовами з’єднання. Додатково у формування часу очікування інтегровано стохастичну компоненту, діапазон якої задається адміністративно та масштабується відповідно до поточних умов у мережі. Це дозволяє зменшити ймовірність одночасного спрацювання тайм-аутів у кількох вузлах, що є однією з основних проблем базових реалізацій таких алгоритмів як Raft.

# РОЗДІЛ 3. МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНИХ СИСТЕМ

## 3.1. Розрахунок ймовірності виявлення відмови вузла із використанням ймовірнісного байєсівського підходу

Байєсівський висновок надає надійну основу для оновлення ймовірностей збоїв вузлів у системах консенсусу шляхом постійної інтеграції нових подій. Однак його застосування в системах реального часу стикається з викликами через обчислювальну складність постійного перерахунку ймовірностей, особливо в середовищах із багатьма змінними. Ця складність може призводити до надмірного навантаження на ресурси та погіршення продуктивності в системах, які мають працювати в умовах жорстких обмежень реального часу.

Наприклад, для точного обчислення ймовірності збою система може одночасно враховувати різні фактори, такі як затримки в мережі, стан обладнання та історичні дані про продуктивність. Ця інтеграція, яка передбачає врахування всіх можливих комбінацій цих факторів, може стати надмірно витратною з точки зору обчислювальних ресурсів, особливо коли система повинна реагувати в режимі реального часу. Ситуація ускладнюється в системах із великою кількістю вузлів, де кожен вузол повинен часто оновлювати свої ймовірності збоїв, що багаторазово збільшує обчислювальне навантаження [156].

Нехай, в розподіленій системі кожен вузол регулярно надсилає повідомлення працездатності (heardbeat) (сигнал працездатності). Якщо повідмлення не надійшло, існує певна імовірність, що вузол дійсно вийшов із ладу, одночасно з тим присутня ймовірність що це тимчасова проблема з мережею.

Використаємо ймовірнісний метод щоб зробити правильний висновок про відмову вузла, сформуємо вхідні дані:

1. Апріорна ймовірність збою вузла.

Позначимо $P(Failure) = p$ та для прикладу вважатимемо:

$$p = P(Failure) = 0.01 \quad (1\%). \tag{3.1}$$

Тобто на підставі історичних даних вважаємо, що в середньому 1 вузол зі 100 може мати збій у довільний момент часу.

2. Імовірність пропущеного повідомлення працездатності (heardbeat)

Якщо вузол справний, але трапляється мережевий збій або затримка, подія пропущення повідомлення працездатності (heardbeat) може статися з імовірністю $\alpha$.

Якщо вузол дійсно вийшов з ладу, подія «пропущений heartbeat» трапляється з імовірністю $\beta$ де $\beta > \alpha$.

Нехай для прикладу:

$$\alpha = P(MissedHeartbeat \mid \neg Failure) = 0.05 \tag{3.2}$$

$$\beta = P(MissedHeartbeat \mid Failure) = 0.80 \quad (80\%) \tag{3.3}$$

Тобто $\alpha = 5\%$ означає, що навіть справний вузол іноді може не відповісти на повідомлення через звичайне перевантаження мережі. $\beta = 80\%$ означає, що якщо вузол дійсно вийшов з ладу, у $80\%$ випадків ми спостерігатимемо пропущене повідомлення “серцебиття” (у $20\%$ може бути так, що збій стався, але ми ще встигли отримати останнє повідомлення перед відмовою).

3. Подія $E$ — виявлено пропущене повідомлення працездатності (heardbeat).

*Крок 1. Обчислення повної імовірності події* ***E***

За формулою загальної ймовірності:

$$P(E) = P(E \mid Failure) \times P(Failure) + P(E \mid \neg Failure) \times P(\neg Failure) \tag{3.4}$$

Підставимо числові значення:

$$P(E) = 0.80 \times 0.01 + 0.05 \times 0.99 = 0.008 + 0.0495 = 0.0575 \tag{3.5}$$

Отже, імовірність того, що ми спостерігатимемо пропущене повідомлення працездатності (heardbeat) (враховуючи що вузол може бути як справним так і не справним) дорівнює $5.75\%$.

*Крок 2. Обчислення апостеріорної ймовірності збою вузла*

За теоремою Байєса:

$$P\left(Failure \mid E\right) = \frac{P\left(E \mid Failure\right) \times P\left(Failure\right)}{P\left(E\right)}. \tag{3.6}$$

Підставимо числа:

$$P\left(Failure \mid E\right) = \frac{0.80 \times 0.01}{0.0575} \approx \frac{0.008}{0.0575} \approx 0.139. \tag{3.7}$$

Отже, апостеріорна ймовірність того, що вузол вийшов з ладу за умови, що ми спостерігаємо пропущене повідомлення працездатності (heardbeat), становить близько $13.9\%$.

Без застосування методу виявлення відмов у роботі розподілених сервісних систем на основі ймовірнісного байєсівського підходу, «статичний» детектор, одразу б реагував на відсутність повідомлення працездатності (heardbeat) і оголошував вузол несправним (або негайно запускати процедуру переобрання лідера). Це часто призводить до хибних спрацьовувань, адже 5% пропусків для справних вузлів — це досить суттєвий відсоток.

Застосування байєсівського підходу показує, що лише за одного пропущеного повідомлення ймовірність реального збою дорівнює приблизно $14\%$, а це ще не підстава робити радикальний висновок про відмову вузла.

*Крок 3. «Накопичення» нових доказів та повторне обчислення*

Замість різкого «статичного» порогу (наприклад, один пропущене повідомлення працездатності (heardbeat) дорівнює відмові), байєсівський підхід дозволяє накопичувати кілька ознак (подій) і поступово підвищувати впевненість у відмові. Розгляньмо для прикладу 2 послідовно пропущені

повідомлення працездатності (heardbeat) тим самим вузлом. Позначимо таку комплексну подію як $E_2$ («пропущено 2 рази поспіль»).

Імовірність двох пропущених повідомлень поспіль при умові що вузол працездатиний та непрацездатний виразимо так:

Якщо вузол справний $\neg Failure$ :

$$P\left(E_2 \mid \neg Failure\right) = \alpha^2 = 0.05^2 = 0.0025 \quad \left(0.25\%\right) \tag{3.8}$$

Якщо вузол несправний ( $Failure$ ):

$$P\left(E_2 \mid Failure\right) = \beta^2 = 0.80^2 = 0.64 \quad \left(64\%\right) \tag{3.9}$$

Повна ймовірність події $E_2$ :

$$P\left(E_2\right) = P\left(E_2 \mid \text{Failure}\right) \times P\left(\text{Failure}\right) + P\left(E_2 \mid \neg\text{Failure}\right) \times P\left(\neg\text{Failure}\right) \tag{3.10}$$

$$P\left(E_2\right) = 0.64 \times 0.01 + 0.0025 \times 0.99 = 0.008875 \quad \left(\approx 0875\%\right) \tag{3.11}$$

Апостеріорна ймовірність збою за умови $E_2$ :

$$P\left(Failure \mid E_2\right) = \frac{P\left(E_2 \mid Failure\right) \times P\left(Failure\right)}{P\left(E_2\right)} = \frac{0.64 \times 0.01}{0.008875} \approx 0.72 \quad \left(72\%\right) \tag{3.12}$$

Таким чином, якщо вузол двічі поспіль не відповів на повідомлення працездатності (heardbeat), ймовірність того, що він дійсно вийшов з ладу, зростає вже до 72%.

Якщо встановимо поріг на рівні 0.5 (50%), то після одного пропуску повідомлення працездатності (heardbeat) (14%) система не вважатиме вузол несправним, а просто продовжить спостерігати.

Але після двох поспіль пропущених сигналів (72%) система уже з високою ймовірністю (вище порогу у 50%) оголосить вузол несправним і ініціює процедури відновлення чи переобрання лідера.

Такий підхід різко знижує частку хибнопозитивних спрацьовувань у тих ситуаціях, коли пропущені сигнали були наслідком короткочасних мережевих затримок, а не реальних відмов.

Тепер розглянемо ймовірність двох послідовних пропусків повідомлення працездатності (heardbeat) для справного та несправного вузла, використовуючи вже визначе параметри:

$\alpha = 0.05$ – ймовірність пропущення повідомлення працездатності (heardbeat) для справного вузла.

$\beta = 0.8$ – ймовірність пропущення повідомлення працездатності (heardbeat) для несправного вузла.

Ймовірність серії $k$ пропусків обчислюється за формулою:

$$P(E_k \mid Failure) = \beta^k, \quad P(E_k \mid \neg Failure) = \alpha^k. \tag{3.13}$$

Враховуючи дані можна розрахувати відповідні ймовірності. Результати розрахунків наведено у таблиці 3.1

Таблиця 3.1

Розраховані значення ймовірності послідовних пропусків повідомлення працездатності (heardbeat)

| $k$ | $P(E_k \mid Failure)$ | $P(E_k \mid \neg Failure)$ |
|---|---|---|
| 1 | 0.05 | 0.8 |
| 2 | $0.05^2 = 0.0025$ | $0.8^2 = 0.64$ |
| 3 | $0.05^3 = 0.000125$ | $0.8^3 = 0.512$ |
| 4 | $0.05^4 = 0.00000625$ | $0.8^4 = 0.4096$ |
| 5 | $0.05^5 = 0.0000003125$ | $0.8^5 = 0.32768$ |
| 6 | $0.05^6 = 0.000000015625$ | $0.8^6 = 0.262144$ |
| 7 | $0.05^7 = 0.00000000078125$ | $0.8^7 = 0.2097152$ |
| 8 | $0.05^8 = 0.0000000000390625$ | $0.8^8 = 0.16777216$ |
| 9 | $0.05^9 = 0.000000000001953125$ | $0.8^9 = 0.134217728$ |
| 10 | $0.05^{10} = 0.00000000000009765625$ | $0.8^{10} = 0.1073741824$ |
| 11 | $0.05^{11} = 0.000000000000000048828125$ | $0.8^{11} = 0.08589934592$ |
| 12 | $0.05^{12} = 0.000000000000000000244140625$ | $0.8^{12} = 0.068719476736$ |
| 13 | $0.05^{13} = 0.000000000000000000001220703125$ | $0.8^{13} = 0.0549755813888$ |
| 14 | $0.05^{14} = 0.000000000000000000006103515625$ | $0.8^{14} = 0.04398046511104$ |

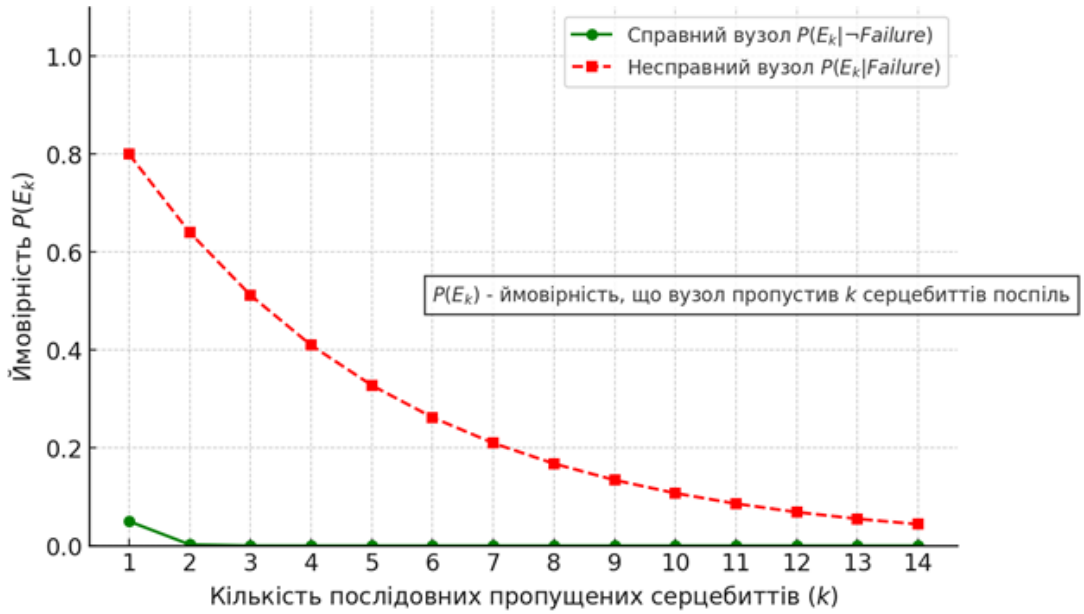



Рис. 3.1. Зміна ймовірності послідовних пропусків повідомлення працездатності (heardbeat) для справного та несправного вузла

Завершивши обрахунки по аналогії з $E_1$ та $E_2$, побудуємо графік залежності ймовірності збою вузла $P(Failure|E)$ від кількості послідовних пропущених серцебиттів $k$.

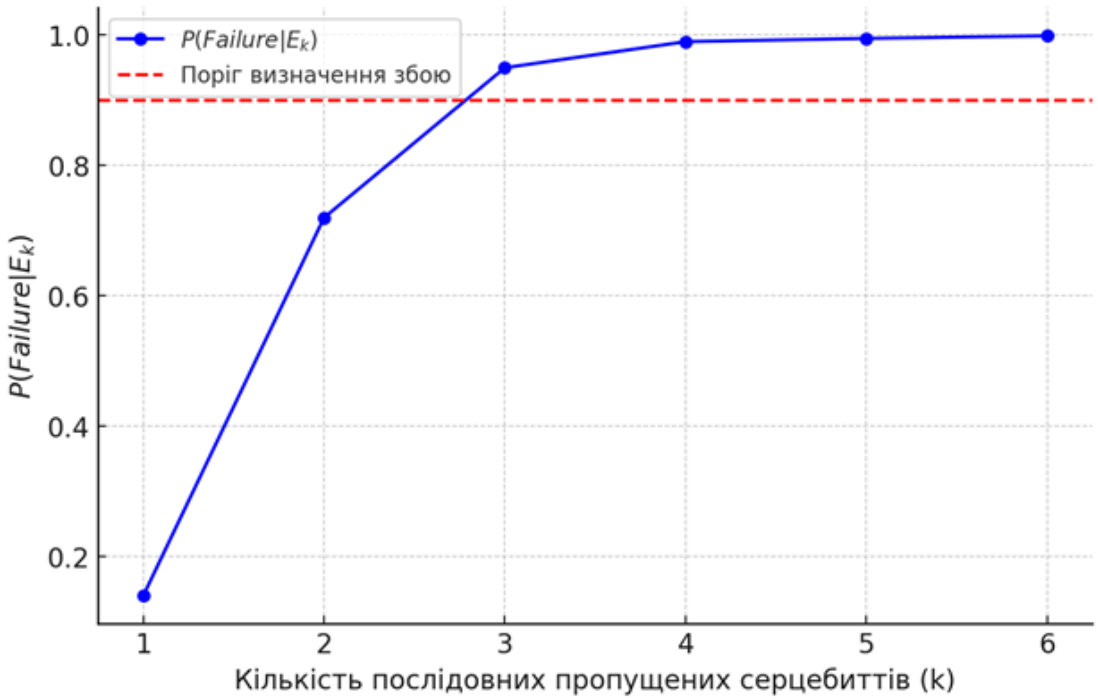



Рис. 3.2. Ймовірність збою вузла в залежності від кількості послідовних пропущених серцебиттів

Як видно з отриманих результатів, при малих значеннях $k$ ймовірність збою вузла відносно низька, але з кожним наступним пропуском повідомлення працездатності (heardbeat) вона швидко зростає. При $k=2$ вона досягає $70\%$, а при $k=3$ перевищує 90%. Вже при $k=4$ і більше значення ймовірності практично наближаються до $1$. На рисунку 3.2 червона пунктирна лінія позначає поріг визначення збою (наприклад, $90\%$. ). Якщо ймовірність перевищує цей поріг, система визнає вузол несправним.

Ключовою особливістю методу виявлення відмов на основі ймовірнісного байєсівського підходу є його здатсть оцінювати ймовірність збою, враховуючи кілька факторів. Розглянемо приклад, у якому система враховує дві події:

I. Пропущене "серцебиття"($E_1$) — вузол не надіслав повідомлення працездатності (heardbeat).

II. Висока затримка відповіді ($E_2$) — вузол відповів із великою затримкою.

Якщо розглядати лише одне пропущене повідомлення, ймовірність збою була обчислена раніше і становить:

$$P(Failure \mid E_1) \approx 0.139 \ . \tag{3.14}$$

Тепер додамо другу подію: значне збільшення затримки відповіді. Припустимо:

$P(E_2 \mid Failure) = 0.7$ - при збої вузол повільно відповідає в 70% випадків.

$P(E_2 \mid \neg Failure) = 0.1$ - повністю працездатний вузол рідко відповідає із затримкою, лише в 10% випадків.

Обчислимо апостеріорну ймовірність збою вузла після спостереження обох подій за формулою Байєса:

$$P(Failure \mid E_1, E_2) = \frac{P(E_2 \mid Failure) \times P(Failure \mid E_1)}{P(E_2 \mid Failure) \times P(Failure \mid E_1) + P(E_2 \mid \neg Failure) \times P(\neg Failure \mid E_1)} \tag{3.15}$$

Підставимо значення:

$$P\left(Failure \mid E_1, E_2\right) = \frac{0.7 \times 0.139}{\left(0.7 \times 0.139\right) + \left(0.1 \times 0.861\right)}, \tag{3.16}$$

$$P\left(Failure \mid E_1, E_2\right) = \frac{0.0973}{0.0973 + 0.0861} = \frac{0.0973}{0.1834} \approx 0.53. \tag{3.17}$$

Таким чином, якщо спостерігається не лише пропущене повідомлення, але й підвищена затримка відповіді, ймовірність збою вузла зростає до 53%. Це значно підвищує впевненість у реальному збої, що дозволяє уникати передчасних реакцій та знижує частку хибнопозитивних спрацьовувань.

Тож якщо припустити, що реальні збої у системі трапляються з частотою близько 1% (тобто 1 вузол зі 100 дійсно виходить з ладу у певний проміжок часу), то з урахуванням типового рівня мережевих пропусків (наприклад, 5%), класичний статичний підхід з порогом “один пропуск = відмова” може призводити до хибних спрацьовувань у кожному двадцятому випадку, тобто близько 5%. Байєсівський підхід дозволяє зменшити цю частку до приблизно 0,5%, оскільки рішення приймається лише при послідовних пропусках або наявності кількох підозрілих ознак, а не після одиничного інциденту.

Приведений розрахунок підтверджує коректність і доцільність розвитку методу виявлення відмов: застосовуючи ймовірнісний байєсівський підхід, система переходить від «жорстких» однокрокових рішень до гнучкого визначення відмови вузла, що мінімізує помилкові реакції на тимчасові проблеми й покращує загальну надійність розподілених сервісних систем. А загальне покращення у зменшенні хибнопозитивних спрацьовувань становить близько 4–5%. Це покращення досягається завдяки більш гнучкому, накопичувальному аналізу подій та адаптивному порогу, що враховує історію поведінки вузла і додаткові ознаки збою.

## 3.2. Дослідження ефективності методу адаптивного визначення порогу відмов у мережах IoT з використанням моделі машинного навчання SARIMA на основі розробленої імітаційної моделі

Проведемо дослідження ефективності удосконаленого методу адаптивного визначення порогу відмов. Зосередимося на аналізі та прогнозуванні часу відгуку у невеликому кластері IoT мережі. Реалізуємо це за допомогою мови програмування Python, а саме програмним пакетом Pandas. Pandas — це швидкий, потужний, гнучкий і зручний у використанні інструмент, що дасть можливість не лише сконфігурувати мережу, а й здійснювати відкритий аналіз та обробку параметрів функціонування.

Конфігурація кластера складалася з десяти пристроїв IoT, з'єднаних у локальну мережу, причому кожен пристрій відповідає за виконання певних завдань. Час відповіді фіксувався через регулярні інтервали в 10 секунд між послідовними запитами. Дані, які передавалися, сформували масив, що складається з вимірювань часу відповіді, отриманих із конкретної конфігурації кластера IoT протягом одного тижня. Обраний набір даних візуалізовано на рисунку 3.3, що дає змогу оцінити часові зміни та потенційні аномалії в даних про час відповіді [156].

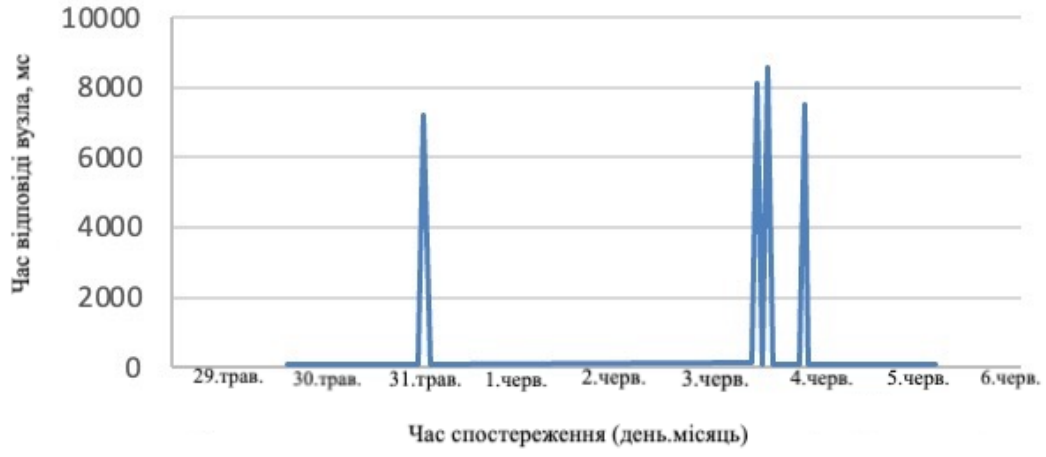



Рис. 3.3. Часові зміни та потенційні аномалії в даних про час відповіді.

Як видно з рисунку 3.3, дані моніторигу містять значні сплески часу відповіді, що чітко вказує на несправність обладнання в ці моменти. Нас цікавить

затримка в IoT-мережі. Відфільтруємо значення, які значно перевищують норму, і для спрощення моделювання обробимо дані, обчисливши медіанне значення часу відповіді протягом години. Після обробки отримаємо розподіл, представлений на рисунку 3.4

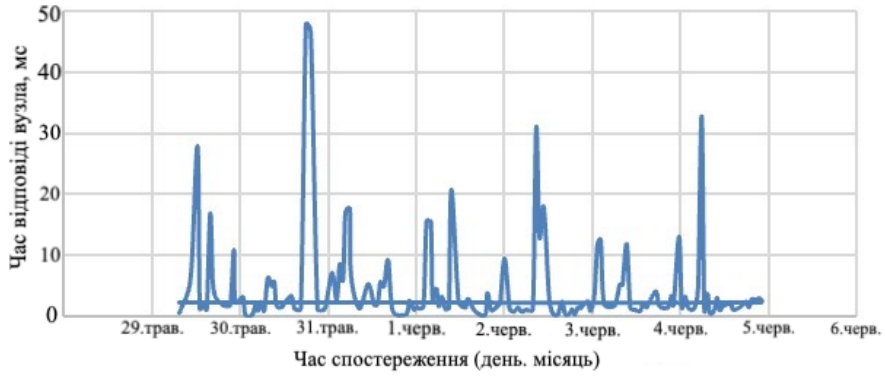



Рис. 3.4. Відфільтровані значення часу відповіді вузла IoT у період з 29 травня по 5 червня

Після фільтрації набору даних ми можемо почати застосовувати модель машинного навчання. Щоб визначити авторегресійну частину нашої моделі, скористаємося графіком автокореляційної функції. Цей графік допоможе проаналізувати кореляції між змінною та її лаговими значеннями, що сприятиме визначенню авторегресійних параметрів моделі.

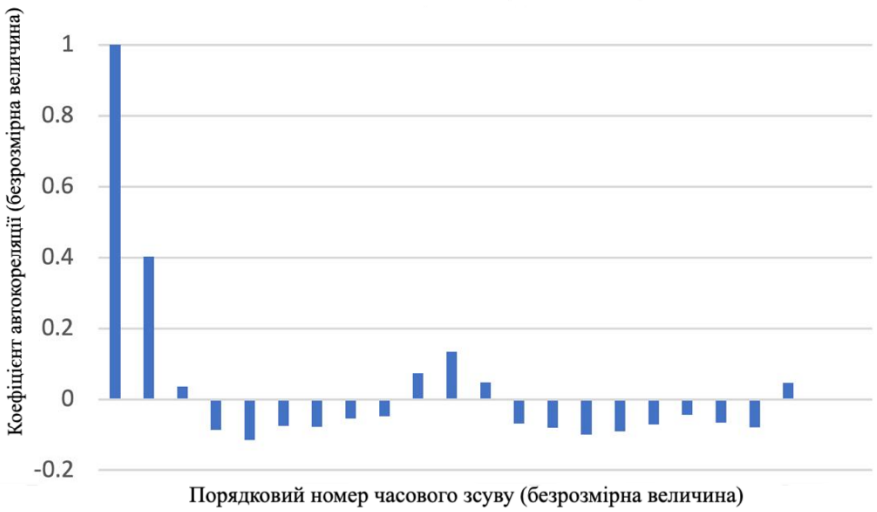



Рис. 3.5. Значення автокореляційної функції для розподілу часу відповіді IoT вузла

Функція автокореляції при зсуві 0 (поточне значення затримки) є кореляцією ряду затримок із самим собою, тому результатом буде кореляція, рівна 1.

Як видно з графіка (рисунок 3.5), існує сильна залежність при лагу 1, після чого залежність різко зменшується. Звідси випливає, що компонент ковзного середнього ( $MA$ ) у нашій моделі може мати значення 1. Однак для більш точного моделювання використаємо $MA(2)$ оскільки значення автокореляції ( $ACF$ ) залишаються досить високими на цих точках.

На цьому етапі модель можна представити формулою:

$$Y_t = c + e_t + \theta_1 e_{t-1} + \ldots + \theta_{14} e_{t-14}, \tag{3.18}$$

де $Y_t$ — значення часових рядів у момент часу *t*; $c$ — константний член або інтерцепт моделі. Він представляє середнє значення часових рядів і часто включається для усунення тренду або упередженості в даних; $e_t$ — залишковий член у момент часу *t*. Він представляє різницю між фактичним значенням часових рядів і його передбаченим значенням; $\theta_1$, $\theta_2$, ..., $\theta_{14}$ — коефіцієнти лагових залишкових членів.

Лагові залишкові члени — білі шуми, тобто випадкові помилки прогнозування, тобто помилки на попередніх етапах часу $i-1$, $i-2$, ..., $i-14$, відповідно. Коефіцієнти вказують на силу та напрямок впливу минулих помилок на поточне значення часових рядів. У моделі $MA$ ці коефіцієнти відомі як параметри ковзного середнього.

Щоб визначити порядок компоненти, побудуємо графік часткової функції автокореляції ( $PACF$ ).


Рис. 3.6. Значення часткової автокореляційної функції ($PACF$) для розподілу часу відповіді IoT вузла

Часткова функція автокореляції ($PACF$) схожа на автокореляційну функцію ($ACF$), але вона відображає лише кореляцію між двома точками спостереженнями, яку не можна пояснити часовими зсувами (лагами) між цими спостереженнями. Наприклад, часткова автокореляція для лагу 3 враховує лише кореляцію, яку не пояснюють лаги 1 та 2. Іншими словами, часткова кореляція для кожного лагу — це унікальна кореляція між цими двома спостереженнями після часткового усунення проміжних кореляцій.

Аналізуючи графік $PACF$, можна зробити висновок, що значення у точках 1, 2, 3 і 4 залишаються на високому рівні. Таким чином, для побудови авторегресійної ($AR$) частини ці значення є достатніми, тобто $AR(4)$. Однак для більш точного прогнозування, як і у випадку з $ACF$, будемо використовувати ці лаги.

$$Y_t = c + e_t + \theta_1 e_{t-1} + \ldots + \theta_{14} e_{t-14} + a_1 y_{t-1} + \ldots + a_{14} y_{t-14}, \qquad (3.19)$$

де $Y_t$ – значення часової серії у момент часу $t$; $c$ – константний член або перехоплення моделі, що представляє середнє значення часової серії та часто використовується для усунення тренду або зміщення в даних; $e_t$ – похибка у

момент часу $i$, яка представляє різницю між фактичним значенням часової серії та її прогнозованим значенням; $\theta_1$, $\theta_2$, ..., $\theta_{14}$ – коефіцієнти запізнених похибок, які є параметрами моделі, що оцінюються на основі даних. Запізнені похибки – це похибки на попередніх часових кроках $t-1$, $t-2$, ..., $t-14$; ці коефіцієнти вказують на силу та напрямок впливу минулих похибок на поточне значення часової серії. У моделі $MA$ (Moving Average) ці коефіцієнти відомі як параметри ковзного середнього; $a_1$, $a_2$, ..., $a_{14}$ – коефіцієнти авторегресійних термінів, які представляють зв'язок між поточним значенням часової серії та її минулими значеннями, також їх називають коефіцієнтами лагів; $y_{t-1}$, $y_{t-2}$, …, $y_{t-14}$ – значення часової серії на попередньому кроці $t-1$, $t-2$, ..., $t-14$.

Введемо інтеграцію даних для отримання стаціонарного розподілу значень з метою покращення прогнозування.

$$\Delta Y_t = c + e_t + \theta_1 e_{t-1} + \ldots + \theta_{14} e_{t-14} + a_1 \Delta y_{t-1} + \ldots + a_{14} \Delta y_{t-14}, \qquad (3.20)$$

Де — значення часового ряду у момент часу $t$; $c$ — стала величина або точка перетину моделі, яка представляє середнє значення часового ряду й часто включається для усунення тренду або зміщення у даних; $e_t$ — похибка в момент часу $t$, що відображає різницю між фактичним значенням часового ряду та його прогнозованим значенням; $\theta_1$, $\theta_2$, ..., $\theta_{14}$ — коефіцієнти заиманих похибок, які є параметрами моделі, що оцінюються за даними. Затримані похибки — це похибки на попередніх часових кроках $t-1$, $t-2$, ..., $t-14$ відповідно. Коефіцієнти вказують на силу та напрямок впливу минулих похибок на поточне значення часового ряду. У моделі $MA$ (Moving Average) ці коефієнти як параметри ковзного середнього; $a_1$, $a_2$, ..., $a_{14}$— коефіцієнти авторегресивних членів, які представляють залежність між поточним значенням часового ряду та його попередніми значеннями і також називаються коефіцієнтами затримки; $\Delta y_{t-1}$, $\Delta y_{t-2}$, …, $\Delta y_{t-14}$ — перша різниця значень часового ряду на попередніх кроках $t-1$, $t-2$, ..., $t-14$.

Розглядаючи вхідний набір значень, можна побачити сезонну складову даних, тобто залежність зміни функції протягом доби, хоча це не завжди чітко виражено. Це означає, що можна удосконалити модель визначення порогу відмов, додавши сезонність, тобто використовуючи модель сезонного авторегресивного інтегрованого ковзного середнього (Seasonal Auto Regressive Integrated Moving Average). Вибрати сезонність досить просто, оскільки вона відповідає сезонному патерну протягом доби, що становить 24 години. Сезонність також можна відстежити за допомогою графіка автокореляційної функції ( $ACF$ ).

Усі описані вище параметри будуть вхідними даними для моделі машинного навчання *SARIMA*. Саме завдяки її інтеграції матимемо можливість прогнозувати значення часу відповіді та встановити його граничне значення, що дасть можливість при нестабільності мережевих з'єднань, визначати чи вузол просто завантажений обробкою запитів чи відбувся збій у його функціонуванні. У результаті аналізу даних і функцій $ACF$ та $PACF$ визначено модель і параметрами *SARIMA*, яка представлятиметься у вигляді $S(1,1,1,24)\,AR(14)\,I(1)\,MA(14)$, де *1,1,1* — це значення $AR$, $I$ та $MA$ у сезонній складовій відповідно.

$$\begin{aligned}\Delta Y_t = c + \ & e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_{14} e_{t-14} + a_1 \Delta y_{t-1} \\ & + a_2 \Delta y_{t-2} + \ldots + a_{14} \Delta y_{t-14} + \psi_1 \varepsilon_{t-1} + \psi_2 \varepsilon_{t-2} + \ldots + \psi_{24} \varepsilon_{t-24}\end{aligned}, \qquad (3.17)$$

де $\Delta Y_t$ — значення часового ряду у момент часу $t$ ; $c$ — стала величина або точка перетину моделі, яка представляє середнє значення часового ряду й часто використовується для усунення тренду або упереджеі у даних; $e_t$ — похибка у момент часу *t*, що відображає різницю між фактичним значенням часового ряду та його прогнозованим значенням; $\theta_1$, $\theta_2$ . , ..., $\theta_{14}$ — коефіцієнти затриманих похибок, які є параметрами моделі, що оцінюються на основі даних. Затримані похибки — це похибки на попередніх часових кроках ($t-1$, $t-2$, ..., $t-14$ відповідно). Коефіцієнти показують силу та напрямок впливу минулих похибок

на поточне значення часового ряду. У моделі $MA$ ці коефіцієнти відомі як параметри ковзного середнього. $a_1$, $a_2$, ..., $a_{14}$ — це коефієнорегресивних членів, які представляють зв'язок між поточним значенням часового ряду та його попередніми значеннями, також відомі як коефіцієнти затримки. $\Delta y_{t-1}$, $\Delta y_{t-2}$, …, $\Delta y_{t-14}$ — перша різниця значень часового ряду на попередніх кроках $t-1$, $t-2$, ..., $t-14$. $\psi_1$, $\psi_2$,..., $\psi_3$— коефіцієнти сезонної $MA$ — компоненти, які представляють вплив затриманих похибок на поточне значення $Y$ після різницювання і видалення сезонної компоненти.

Для тренування і тестування розділимо вхідний набір даних на дві частини: одну для навчання і другу для тестування. Дані з 29 травня по 3 червня використовуються для навчання моделі, а дані з 3 червня по 4 червня зарезервовані для тестування моделі, тобто для порівняння з передбаченими значеннями. Тестовий набір даних представлено на рисунку 3.7.


Рис. 3.7. Розподіл тестового часу відповіді вузла IoT без використання моделі машинного навчання SARIMA

Використовуючи модель машинного навчання *SARIMA* та сезонного авторегресивного інтегрованого ковзного середнього, отримано ймовірнісний розподіл затримки відповіді вузла IoT протягом доби (рисунок 3.8).


Рис. 3.8. Прогнозований розподіл часу відповіді вузла IoT із використанням моделі машинного навчання SARIMA

Проаналізуймо детальніше тестовий та прогнозований набори даних (рисунок 3.9).


Рис. 3.9. Спрогнозований (синя крива) та тестовий (червона крива) розподіл часу відповіді вузла IoT

Враховуючи, що спрогнозований і тестовий розподіл фактично співпадає, оцінемо, наскільки впровадження методу адаптивного визначення порогу відмов

з використанням моделі машинного навчання *SARIMA* ефективне. Припустимо, що адміністратор виставив статичний поріг у 10 мс (без використання методу). У такому випадку протягом доби система зможе коректно працювати, витримавши 3 з 5 піків погіршення працездатності мережі (затримки відповіді), хоча всі ці піки насправді є лише тимчасовими коливаннями (у випадку порівняння оранжевої кривої з статичним порогом). З іншого боку, використовуючи прогноз затримки (тестовий розподіл на рис. 3.9), можна розрахувати динамічний поріг шляхом додавання до прогнозу сталої (наприклад, 2 мс — визначеної адміністратором), щоб забезпечити розташування порогу трохи вище фактичної затримки. Це дозволить краще враховувати зміни навантаження та зменшує кількість хибних спрацювань.

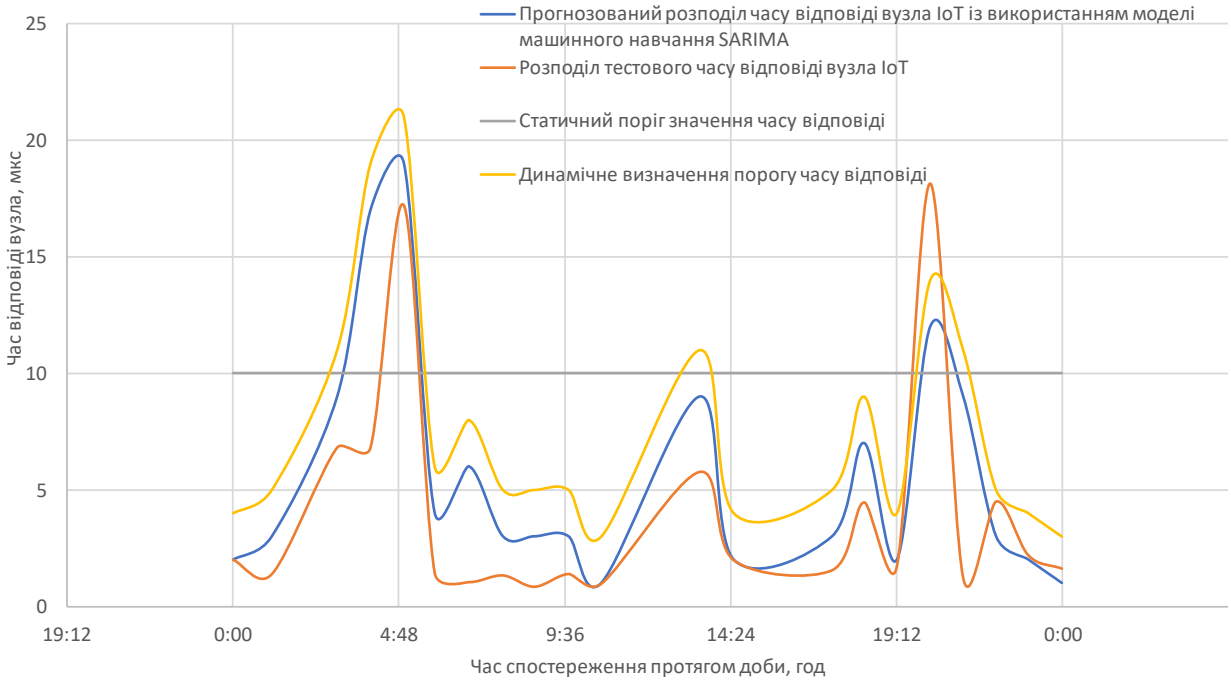



Рис. 3.10. Розподіл чісу відповіді вузла із використанням адаптивного визначення порогу відмов

Аналізуючи результати на рис. 3.10, маємо п’ять піків затримки о 4.48, ~13.00, ~19.00, ~19.30, ~20.00 (оранжева лінія). При впровадженні методу адаптивного визначення порогу відмов з використанням моделі машинного навчання *SARIMA* кластер витримує 4 з 5 піків навантаження. Таким чином, динамічний підхід демонструє покращення точності виявлення приблизно на

20% порівняно зі статичним методом, забезпечуючи більш адаптивну та ефективну реакцію системи на нестабільні умови мережі.

## 3.3. Розроблення імітаційної моделі кластеру розподіленої сервісної системи для оцінки ефективності визначення потенційного лідера з використанням автономного адаптивного методому

### 3.3.1. Особливості реалізації імітаційної моделі кластеру для оцінювання працездатності алгоритмів консенсусу за умов мережевих флуктуацій

Для дослідження впливу нестабільних метрежевих з’єднань на явище безперервного переобрання лідера при роботі алгоритму консенсусу розроблено імітаційну модель кластеру розподіленої сервісної системи для оцінки ефективності визначення потенційного лідера автономним адаптивним методом (запропонованого у п. 2.3), яка відтворює кластер, що складається з п'яти вузлів. Кожен вузол у цьому кластері поєднаний з кожним іншим вузлом, формуючи повністю взаємопов'язану мережу. Ця структура висвітлена на рисунку 3.11, що представляє основу для дослідження. Така структура дозволяє детально вивчати динаміку переобрання лідера в контрольованому середовищі, надаючи уявлення про основні механізми, які сприяють або перешкоджають процесу безперервної ротації лідерства [157].

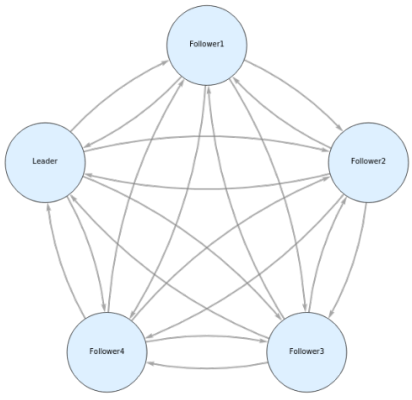



Рис. 3.11. Конфігурація кластера

В контексті алгоритмів консенсусу, заснованих на роботі лідера, особливо Raft, концепція heardbeat виступає як ключовий компонент. Цей концепт включає в себе відправлення лідером сигналів вузлам-послідовникам через встановлені інтервали [7]. Функціонуючи подібно до регуляторного імпульсу, ці сигнали працездатності (heardbeat) є інструментом у підтримці авторитету лідера і запобіганні незапланованих виборів. Вони служать ознакою активного статусу лідера та відіграють важливу роль у забезпеченні синхронізації кластера, хоча й без обміну даними. Модель використовує генератор розподілу з логарифмічно нормальним законом для моделювання затримок між вузлами. Цей вибір базується на реалістичній поведінці мережевих затримок, яким властиво бути невід'ємними та виявляти асиметричний розподіл із схильністю до "довгих хвостів" [43]. Такий розподіл точно відображає природні мережеві затримки, що характеризуються переважно короткими інтервалами зі спорадичними тривалими затримками.

Ядро моделі це запуск кроку алгоритму працездатності (heardbeat), критичного механізму для оцінки поточного стану системи та визначення необхідності реконфігурації у відповідь на модельовані затримки. Процес працездатності (heardbeat) функціонує як діагностичний інструмент, що безперервно моніторить здоров'я системи і ініціює протоколи реконфігурації за потреби.

Іншим центральним елементом цієї моделі є застосування "Закону великих чисел" (LLN), який пояснюється формулою 3.18:

$$X_n = \frac{1}{n}\left(X_1 + \ldots + X_n\right) \quad X_n \to \mu \; n \to \infty, \tag{3.18}$$

де $X_n$ означає середню затримку, спостережену протягом $n$ експериментів, де $\mu$. представляє справжню середню затримку або ймовірність реконфігурації системи у випадку визначеного часу, він має право ініціювати процес виборів тобто реконфігурації кластера з метою призначення нового лідера. Цей механізм

стоїть на варті операційної стабільності та надійності кластера, захищаючи його від потенційних перерв у роботі. затримок. Цей статистичний принцип надає основу для передбачення поведінки системи в широкому спектрі модельованих сценаріїв, тим самим оцінюючи її стабільність та надійність.

Використання закону великих чисел у цьому контексті має на меті через масштабне моделювання виявити двійков результат того, чи вимагає система реконфігурації відповідно до алгоритму консенсусу у відповідь на модельовані затримки. Основна мета полягає в тому, щоб визначити ймовірність спрацьовування реконфігурації через численні модельовані інтервали працездатності (heardbeat), тим самим надаючи всеосяжне розуміння поведінки алгоритму в умовах нормальної експлуатації. Цей метод також служить для оцінки стійкості системи та її адаптивності до мінливої динаміки мережевого зв'язку, вносячи цінну інформацію про стабільність розподіленої системи під різними модельованими умовами.

Застосування закону великих чисел до удосконалення алгоритмів у розподілених системах базується на принципі ймовірнісної стабільності. Це подібно до механізму зворотного зв'язку в системах керування, де продуктивність алгоритму реконфігурації оцінюється через численні ітерації, щоб виявити закономірності у його поведінці. Якщо алгоритм демонструє схильність ініціювати реконфігурації з надмірною частотою або рідкістю, закон великих чисел служить статистичною основою для механізму налаштування вхідних параметрів. У академічному розумінні, закон великих чисел надає основу для емпіричної оцінки. Він кількісно визначає очікувану поведінку на основі обширного набору даних, зменшуючи стохастичні аномалії та забезпечуючи відповідність реакції розподіленої системи на модельовані мережеві умови теоретичними прогнозам. Спостерігаючи збіжність частоти реконфігурацій до стабільного середнього значення, архітектори систем можуть детально налаштувати алгоритм, щоб підтримувати баланс між оперативністю та

стабільністю, що є незамінним для надійності та ефективності будь-якої розподіленої системи.

У простих математичних термінах модель описується наступним чином:

Позначимо $N = 3000$ як загальну кількість проведених експериментів. У кожному експерименті симульовано передачу сигналів працездатності (heardbeat) від лідера до 4 вузлів-послідовників у кластері, враховуючи мережеві затримки.

Визначено бінарний результат для кожного експерименту:

1. 1, якщо відбувається переобрання лідера (тобто, якщо будь-який з послідовників не отримує сигнал працездатності (heardbeat) протягом визначеного часу очікування);
2. 0 - в іншому випадку.

Нехай $X_i$ позначено як результат i-го експерименту, з $i = 1,2,...,N$. Ймовірність ініціації повторних виборів у будь-якому експерименті, базується на мережевій затримці та задається за допомогою: $P(X_i = 1)$. Середня кількість перевиборів, що відбуваються протягом *N* експериментів, обчислюється як:

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{3.19}$$

Значення $\bar{X}$, у свою чергу, оцінює ймовірність $P$ реконфігурації кластера через пропущений сигнал працездатності (heardbeat), коли $N$ зростає. Математично це виражається так:

Значення $\bar{X}$ оцінює ймовірність *P* ініціації реконфігурації кластера внаслідок пропущеного сигналу працездатності (heardbeat) при зростанні кількості спостережень *N*. Математично це виражається як:

$$P = \lim_{N\to\infty} \bar{X} = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} X_i \tag{3.20}$$

Це формулювання базується на законі великих чисел, який стверджує, що середнє значення результатів, отриманих з великої кількості спроб, прямує до математичного сподівання ймовірності події при нескінченному *N*.

Блок-схеми складових розробленої імітаційної моделі кластеру РСС представлені на рисунках 3.12 та 3.13.

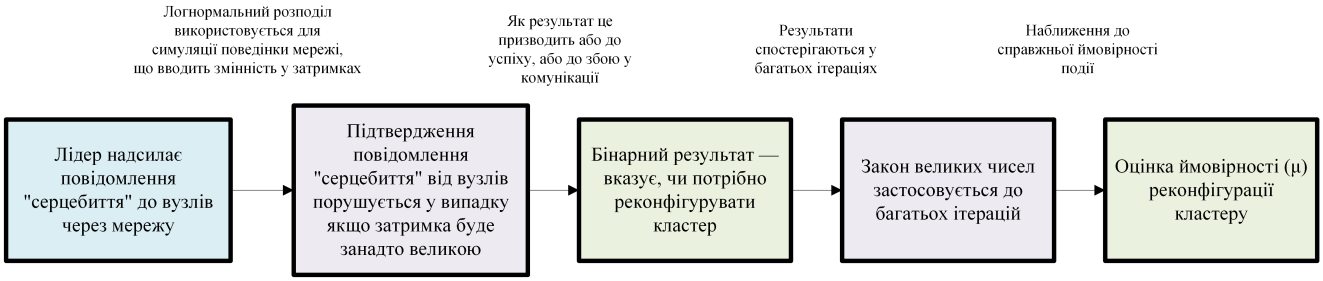



Рис. 3.12 Блок-схема оцінки ймовірності переконфігурації кластера через затримку мережі

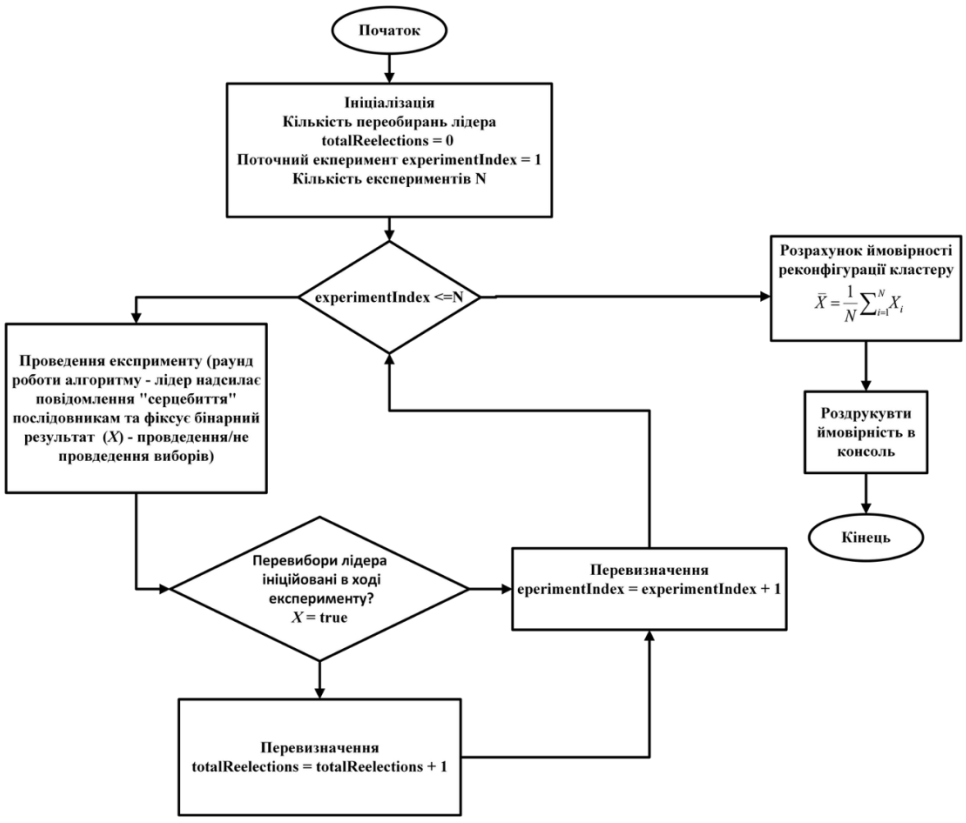



Рис. 3.13. Блок-схема ймовірнісності реконфігурації кластеру РСС

Калібрування параметрів моделі відіграє ключову роль у симуляції розподілених систем. Ця дискусія розпочинається з акцентування на важливості

вибору відповідних вхідних даних, зокрема тайм-ауту виборів та стабільності мережі, для точного оцінювання продуктивності системи.

Вибір значення тайм-ауту у розподілених системах, особливо у контексті мережевих затримок, потребує практичного підходу, який враховує варіабельність і непередбачуваність роботи мережі. Метою є встановлення такого тайм-ауту, який був би достатньо довгим, щоб уникнути зайвих виборів лідера через короткочасні мережеві збої, але водночас достатньо коротким для забезпечення своєчасного перемикання у разі реальних збоїв лідера. Ця задача є специфічною для кожної системи та стану мережі на даний момент і потребує постійної уваги адміністратора системи.

Рекомендований підхід полягає у встановленні тайм-ауту виборів значно вище максимального спостережуваного затримання мережі, з урахуванням буфера для компенсації змінності та несподіваних сплесків. Цей буфер дозволяє уникнути проблем з виборами лідера через тимчасові збої мережі, які можуть дестабілізувати кластер і вплинути на його продуктивність. Проте навіть цей буфер не унеможливлює необхідності переформування системи, оскільки стабільність мережі з часом може змінюватися.

У симуляції ми досліджуємо, як система функціонує на межі своїх можливостей. Для цього тайм-аут виборів встановлено як постійну величину (в нашому випадку 3475 мс, що відповідає географічно розподіленому кластеру), а тайм-аут для переформування — на рівні 0,9 від верхньої межі затримання мережі. У таких умовах мінімальні та максимальні межі затримок у симуляції становлять відповідно 700 мс та 3475 мс. Експеримент проводився 3000 разів для перевірки стабільності моделі, а результати представлені на рисунку 3.14.

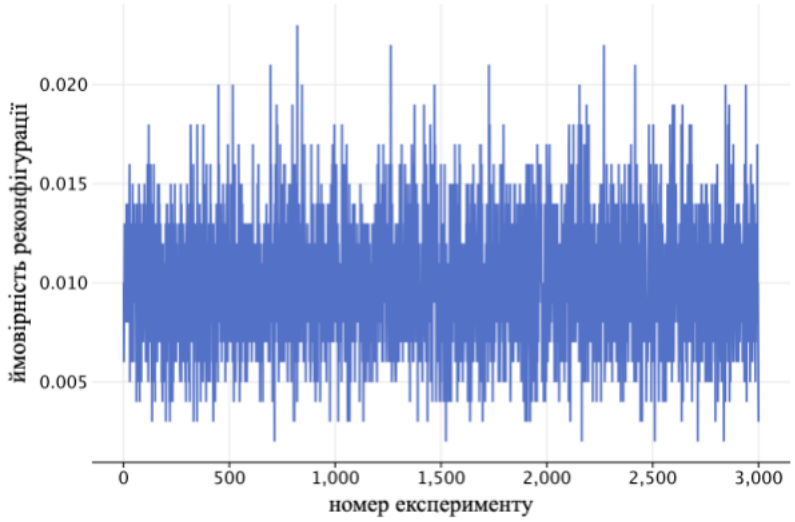



Рис. 3.14. Оцінка ймовірності переконфігурації кластера за результатами 3000 експериментальних випробувань

Як видно, динаміка відсутня, а значення сходиться до 0,01, що означає, що ймовірність переформування кластера становить 1% при кожному надсиланні сигналу працездатності (heardbeat) до вузлів. Таким чином, тепер можливо передбачити, як кластер буде поводитися за конкретних мережевих умов і параметрів кластера.

Існує нагальна потреба у всебічному дослідженні впливу змін у поведінці мережі на продуктивність і надійність алгоритмів консенсусу. Імітаційна модель, представлена у цьому дослідженні, є інструментом для таких досліджень, який дозволяє детально відтворювати різноманітні мережеві умови та їх вплив на процеси консенсусу. Завдяки використанню цієї моделі можемо аналізувати тонкі динамічні взаємозв'язки між нестабільністю мережі та поведінкою алгоритмів консенсусу, отримуючи цінну інформацію для підвищення стійкості розподілених систем.

Розроблена імітаційна модель призначена для дослідження ефективності алгоритмів консенсусу в розподілених системах в умовах мережевих флуктуацій. Вона побудована на основі подієво-орієнтованого підходу, що дозволяє моделювати ключові аспекти роботи розподілених систем, включаючи затримки зв’язків та реконфігурацію кластеру. Практична цінність моделі полягає у

можливості моделювання сценаріїв мережевої нестабільності та оцінки їхнього впливу на стабільність системи.

У моделі, події, такі як надсилання повідомлень працездатності (heardbeat), реконфігурація кластеру та оновлення станів вузлів, визначають потік роботи системи. Такий підхід дає змогу детально симулювати взаємодію між вузлами, відслідковувати затримки сигналів та оцінювати тригерні події, що ініціюють зміну лідера або інші реакції системи.

Кластер вузлів складається з групи вузлів, де один вузол виконує роль лідера, а решта діють як послідовники. Управління кластером здійснюється за допомогою методів, які дозволяють ініціалізувати лідера та виконувати реконфігурацію кластеру у разі втрати зв'язку з лідером. Вузли обробляють повідомленння працездатності (heardbeat) для перевірки стану лідера, а також можуть створювати двосторонні зв'язки між собою для забезпечення комунікації.

Мережеві з'єднання в моделі реалізовані у вигляді двосторонніх зв'язків, які моделюють затримки з використанням генераторів випадкових чисел. Генератори затримок працюють на основі нормального та логнормального розподілів, що дозволяє створювати реалістичні умови мережевих флуктуацій. Фабрика генераторів дозволяє налаштовувати тип і параметри затримок залежно від потреб моделювання.

Алгоритм роботи моделі починається з ініціалізації, під час якої створюється кластер з п'яти вузлів, призначається лідер та встановлюються зв'язки між вузлами. Під час ітерацій моделювання оновлюються затримки зв'язків та перевіряється стан сигналів від лідера. У разі відсутності сигналу від лідера виконується реконфігурація кластеру, і вузли обирають нового лідера. Результати симуляції використовуються для статистичної оцінки ймовірності реконфігурацій відповідно до закону великих чисел.

Код реалізовано мовою Kotlin із застосуванням об'єктно-орієнтованого підходу. Основні елементи реалізації включають ініціалізацію вузлів, створення

зв’язків із використанням генераторів затримок та запуск симуляції. У файлі Simulation.kt налаштовуються сценарії мережевих флуктуацій, включаючи параметри затримок та кількість пошкоджених зв’язків, що дозволяє моделювати різноманітні умови роботи системи [128].

Основні компоненти включають клас Cluster, який моделює групу вузлів із визначенням лідера, клас Node, що представляє вузли системи, і клас Link, який забезпечує двосторонні мережеві зв’язки між вузлами. Для моделювання мережевих затримок використані генератори затримок (Normal та Log Normal), реалізовані через інтерфейс DelaySimulator, що дозволяє динамічно налаштовувати параметри затримок. Зв’язки можуть моделювати різноманітні сценарії мережевих флуктуацій завдяки варіації розподілів затримок, а вузли автоматично обчислюють латентність та ініціюють реконфігурацію кластеру у разі втрати сигналу від лідера. Основний робочий процес включає оновлення стану зв’язків та вузлів із урахуванням затримок, що дозволяє оцінювати стабільність системи за різних умов роботи мережі.

Для аналізу результатів симуляції були інтегровані статистичні методи та інструменти візуалізації. Наприклад, при використанні бібліотеки Kandy (JetBrains Kotlin Data Visualization), побудова лінійних графіків дозволяє простежити тренди реконфігурацій у часі.

Для оцінки розподілів затримок використовувалися гістограми та графіки, що відображають нормальні та логнормальні розподіли. Це дало змогу аналізувати особливості деградації мережі при різних умовах, підтверджуючи отримані значення на основі законів великих чисел.

Для інтерактивного налаштування сценаріїв симуляції та тестування було використано Jupyter Notebook, інтегрований із Kotlin. Це дозволило гнучко налаштовувати параметри симуляції (напр., затримки, кількість вузлів та зв’язків), запускати окремі етапи симуляції та візуалізувати результати в реальному часі.

Параметри, такі як "від початкових до кінцевих значень затримок" (from, until) або кількість деградованих зв’язків (numberOfBrokenLinks), задавалися динамічно через код.

Стислий огляд використаних технологій

Мова програмування: Kotlin (велика модульність, об'єктно-орієнтований підхід, інтеграція з Jupyter Notebook через “%use” мітки).

Бібліотеки для симуляцій: кастомні класи та фабрики (DelaySimulator, Cluster, Node).

Бібліотеки для візуалізації: Kandy (JetBrains) для аналізу даних і побудови графіків.

Інтерактивне тестування: Jupyter Notebook для повторюваних ітерацій налаштувань параметрів і миттєвого спостереження за результатами.

Таким чином, модель базується на сучасних програмних засобах і має модульну архітектуру, що дозволяє ефективно симулювати розподілені сервісні системи в умовах мережевих флуктуацій, оцінюючи ефективність алгоритмів консенсусу.

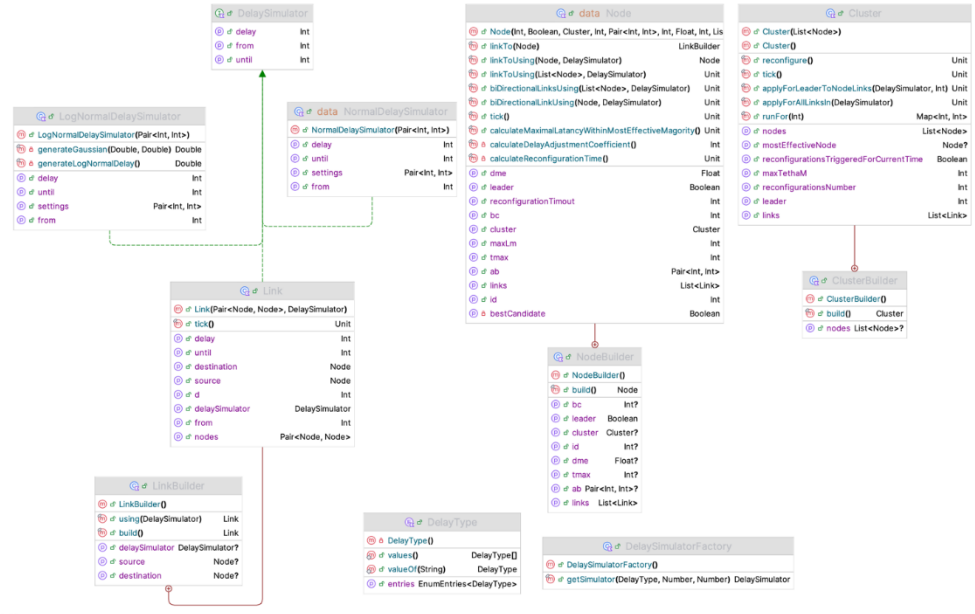



Рис. 3.15. UML діаграма розробленої моделі симуляції для оцінки автономного адаптивного методу підвищення ефективності розподіленої сервісної системи

Підсумовуючи можна сказати ще ретельне створення та аналіз моделей симуляції, поєднані з застосуванням статистичних принципів, як-от закон великих чисел, надають безцінні уявлення про динаміку розподілених систем, особливо у розумінні явища повторних виборів лідера. Симулюючи різні сценарії та уважно вибираючи параметри моделі, такі як час затримки виборів та стабільність мережі, архітектори систем можуть удосконалити алгоритми консенсусу, щоб знайти баланс між швидкістю реагування та стабільністю. Цей ітеративний процес моделювання, аналізу та налаштування параметрів є незамінним для забезпечення надійності та ефективності розподілених систем у реальних застосуваннях, де умови мережі можуть бути непередбачуваними та динамічними. Постійна пильність та адаптація є ключовими у підтримці оптимальної продуктивності системи та стійкості проти потенційних перебоїв.

### 3.3.2. Моделювання стану кластерної системи в режимі реального часу функціонування мережі

Змоделюємо кластер з п’яти вузлів (рисунок 3.16), де вузол 1 призначемо лідером. За основу візьмемо повнозв’язну структуру взаємозв’язків вузлів міжсобою. Задамо первинні параметри: $T_{\max}$=300, $a$=1, $b$=10, $\delta_c$=50, $\partial_{me}$=0.4

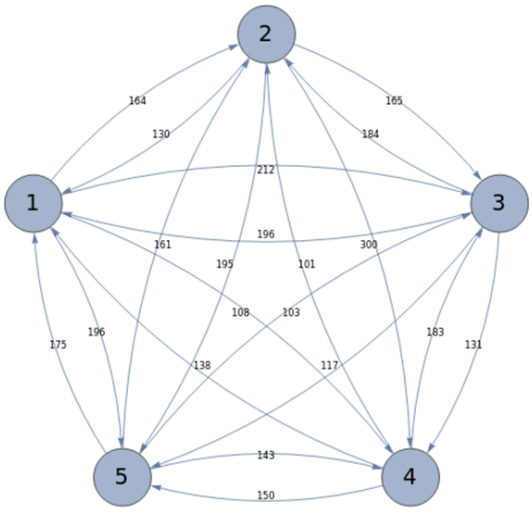



Рис. 3.16. Топологічна структура змодельованого кластеру

Змоделюємо розподіл затримок від кожного вузла в середині кластера відповідно до лог нормального закону розподілу. Отримані значення наведені у таблиці 3.1, а їх розподіл на рис. 3.17.

Таблиця 3.1

Затримки в мережі всередині кластера

| - | Вузол 1 | Вузол 2 | Вузол 3 | Вузол 4 | Вузол 5 |
|---|---|---|---|---|---|
| Вузол 1 | - | 164 | 212 | 108 | 196 |
| Вузол 2 | 130 | - | 165 | 300 | 195 |
| Вузол 3 | 196 | 184 | - | 131 | 117 |
| Вузол 4 | 138 | 101 | 131 | - | 150 |
| Вузол 5 | 175 | 161 | 103 | 143 | - |

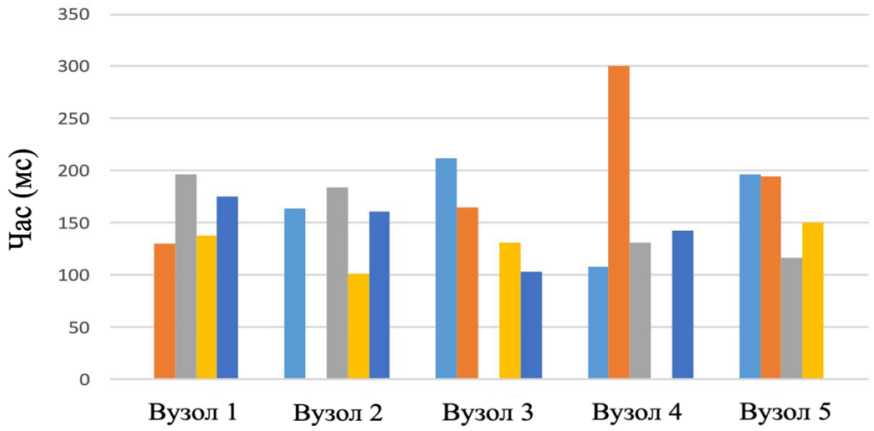



Рис. 3.17. Розподіл затримок від кожного вузла в середині кластера

Припустимо, що кластер працює тривалий час, і всі вузли добре обізнані про стан кластера завдяки процесу обміну повідомлями працездатності (heardbeat) між послідовниками (follower heartbeat). У разі раптового збою лідера (вузла 1) кожен вузол використовуватиме визначений у п. 2.3 алгоритм для обчислення тайм-ауту (рис. 2.9) [83].

З використання формули 2.18, що наведена в п.2.3 визначимо, який із вузлів найкраще підходить для виконання ролі лідера. Ці розрахунки допоможуть визначити вузол з оптимальними характеристиками для лідерства, враховуючи поточні умови мережі та час відгуку вузлів. Результати розрахунків наведені в таблиці 3.2.

Таблиця 3.2.

Властивості затримок у кластері згідно з запропонованим алгоритмом

| Вузол | $\{l_1, l_2, ..., l_{N-1}\}$ | $sort\left(\{l_1, l_2, \ldots l_M\}\right)$ | $max(L_M)$ |
|---|---|---|---|
| 1 | $\{164, 212, 108, 196\}$ | $\{108, 164, 196\}$ | 196 |
| 2 | $\{130, 165, 300, 195\}$ | $\{165, 195, 300\}$ | 300 |
| 3 | $\{196, 184, 131, 117\}$ | $\{117, 131, 184\}$ | 184 |
| 4 | $\{138, 101, 131, 150\}$ | $\{101, 131, 150\}$ | 150 |
| 5 | $\{171, 161, 103, 143\}$ | $\{103, 143, 161\}$ | 161 |

Вузол 4 має мінімальне значення $max(L_M) = 150$ і повинен розглядатися як найкращий кандидат для виконання ролі лідера кластера. Використаємо формулу 2.19 для розрахунку тайм-аутів для кожного вузла в кластері. Результати та основні проміжні обчислення представлені в таблиці 3.3

Таблиця 3.3.

Розрахунок значень тайм-аутів для кожного вузла в кластері

| Вузол | $max(\Theta)$ | $T_{dlbc}$ | $T_{bccc}$ | $T_{dlcc}$ | $rand(a,b)$ | $T$ |
|---|---|---|---|---|---|---|
| 1 | 300 | 108 | 138 | 0 | | |
| 2 | | | 101 | 164 | 8 | 353 |
| 3 | | | 131 | 212 | 3 | 214 |
| 4 | | | 0 | 108 | 4 | 154 |
| 5 | | | 150 | 196 | 1 | 212 |

Як видно з отриманих розрахунків, вузол 4 має значення тайм-ауту 154 мілісекунди. У порівнянні із затримкою між поточним лідером (вузлом 1) і найкращим кандидатом (вузлом 4), яка становить 108 мілісекунд, значення тайм-ауту ідеально відповідає вимогам. Воно є адаптивним і оптимально коротким, що дозволяє реагувати швидше за інших та ініціювати переобрання у випадку аномалій у мережевих затримках. Крім того, це значення є достатньо довгим, щоб врахувати поточний стан мережі, що допомагає запобігти непотрібним переобранням через коливання у мережі.

### 3.4. Висновки до третього розділу

1. Для оцінки ефективності функціонування розподіленої системи в умовах динамічного середовища та непередбачуваних мережевих затримок на основі методу виявлення відмов з використанням ймовірнісного байєсівського підходу проведено розрахунок апостеріорної ймовірності відбою вузла. Проведені розрахунки доводять, що навіть за умов наявності окремих ознак потенційної відмови, таких як пропущені повідомлення працездатності (heardbeat) або підвищена затримка, метод не схильний до передчасних висновків, а натомість дозволяє зважено формувати рішення на основі комбінації факторів. Такий підхід значно зменшує кількість хибнопозитивних спрацьовувань приблизно на 4–5%. Це покращення досягається завдяки більш гнучкому, накопичувальному аналізу подій та адаптивному порогу, що враховує історію поведінки вузла і додаткові ознаки збою.

2. Для оцінки ефективності виявлення відмови в розподіленій сервісній системі системи, що функціонує на основі методу адаптивного визначення порогу відмов з використанням моделі машинного навчання SARIMA розроблено імітаційну модель структури IoT кластера, як частини РСС. В результаті роботи моделі отримано залежність, аналіз якої свідчить, що при прогнозуванні тривалостей часу відповіді за допомогою моделі машинного навчання SARIMA вдається динамічно коригувати поріг відмов та чітко розмежовувати пікові моменти функціонування системи та відмови в обслуговуванні. Це дає змогу краще враховувати зміни навантаження та зменшує кількість хибних спрацювань. Після проведення моделювання із використанням методу адаптивного визначення порогу відмов та моделі машинного навчання SARIMA вдалося підвищити точність виявлення збоїв на 20%, забезпечуючи при цьому адаптивну та ефективну реакцію системи на нестабільні умови функціонування мережі.

3. Імітаційне моделювання кластеру розподіленої сервісної системи для визначення потенційного лідера з використанням запропонованого автономного

адаптивного методу засвідчило його ефективність у динамічному мережевому середовищі. Результати моделювання продемонстрували здатність алгоритму визначення тайм-ауту точно і гнучко визначати вузол, який має найсприятливіші умови для виконання ролі лідера, з урахуванням фактичних характеристик мережі. Обчислені значення тайм-аутів показали адаптивність підходу до поточних умов: мінімізація часу реакції на відмову чинного лідера, водночас із достатнім буфером часу для уникнення хибних спрацювань в умовах флуктуацій затримок. Для повноцінної валідації методу у наступному розділі буде проведено подальше моделювання та практична впровадження за різноманітних сценаріїв та умов функціонування.

# РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ФУНКЦІОНУВАННЯ РОЗРОБЛЕНИХ МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНИХ СЕРВІСНИХ СИСТЕМ

Розподілені сервісні системи є основою сучасних обчислювальних платформ, забезпечуючи масштабованість, високу доступність і стабільність у різних мережевих умовах. Однак ефективне функціонування таких систем ускладнюється через постійні зміни мережевого середовища, можливі затримки у зв'язку та збої окремих компонентів. Ці виклики вимагають впровадження адаптивних методів, які можуть динамічно реагувати на зовнішні зміни, зберігаючи стабільність і продуктивність системи.

У цьому розділі проведена практична реалізація розробленого автономного адаптивного методу покращення функціонування розподілених сервісних систем. Основна увага приділена оцінці ефективності методів в умовах зміни працездатності мережі. Показано, як автономний адаптивний підхід, представлений в попередньому розділі, допомагає мінімізувати кількість реконфігурацій кластеру, знижуючи витрати обчислювальних ресурсів і підвищуючи загальну стійкість системи.

Детально аналізується вплив автономного адаптивного методу на процеси вибору лідера та розподілу ролей у кластері, а також його здатність забезпечувати стабільність навіть за умов критичних мережевих збоїв. Результати досліджень демонструють універсальність і високу ефективність запропонованого підходу в умовах постійних змін середовища.

## 4.1 Дослідження ефективності застосування автомномного адаптивного методу та його вплив на функціонування розподіленої сервісної системи

Для доведення ефективності впровадження автономного адаптивного методу, запропонованого у п. 2.3. використаємо розроблену у п. 3.3. імітаційну модель кластеру РСС. Метрикою підвищення ефективності функціонування

кластеру буде зменшення ймовірності реконфігурації кластеру в умовах нестабільності структури мережі та нерівномірного розподілу затримок між вузлами.

### 4.1.1. Статистична оцінка мережевих характеристик для верифікації параметрів імітаційної моделі кластеру РСС

Щоб точно симулювати поведінку алгоритмів консенсусу, важливо будувати моделі на основі реальних динамік мережі. Така основа є вирішальною для створення симуляцій, що максимально наближені до реальних мережевих умов, покращуючи їх надійність і прогностичну точність.

Для точного моделювання поведінки алгоритму консенсусу основою є реальні динамічні характеристики мережі. Це забезпечує, що симуляції максимально наближені до реальних мережевих умов, покращуючи їх надійність і прогностичну точність. На початковому етапі варто пріоритизувати дослідження природи затримок у мережі.

У цьому дослідженні використано комплексний набір даних із детального аналізу стабільності мережі, ретельно задокументований у Kaggle зошиті Гарі Стаффорда [159]. Цей набір даних спеціально розроблений для захоплення та візуалізації варіабельності часу в з'єднаннях LAN-мереж з Інтернетом, що сприяє глибшому розумінню розподілу затримок у таких мережах. За допомогою часу відгуку на запити ping, зібраного з інтервалом у 10 секунд з різних IoT-пристроїв збору даних, набір даних забезпечує подвійний погляд, охоплюючи як бездротові мережі 2.4 ГГц, так і дротові мережі Ethernet зі швидкістю 100 Мбіт/с. Ці виміри часу, зафіксовані на шляху до локального Інтернет-маршрутизатора і далі до першого сервера у Інтернеті, слугують важливою основою для моделювання поведінки мережі. Наявність прогалин у цих часових рядах надає унікальне уявлення про випадки відключень, які можуть виникати або в самій LAN-мережі, або стосовно здатності маршрутизатора підтримувати підключення

до Інтернету. Використання датаграм протоколу ICMP ECHO_REQUEST та ECHO_RESPONSE забезпечує точність і надійність цих вимірів.

Набір даних відіграє вирішальну роль у моделюванні затримок у моделі. Самі дані представлені на рисунку 4.1. Після аналізу даних, поданих на рисунку, стає очевидним, що очищення даних є необхідним. Таким чином, оновлений набір даних, де видалені викиди, представлений на рисунку 4.2.

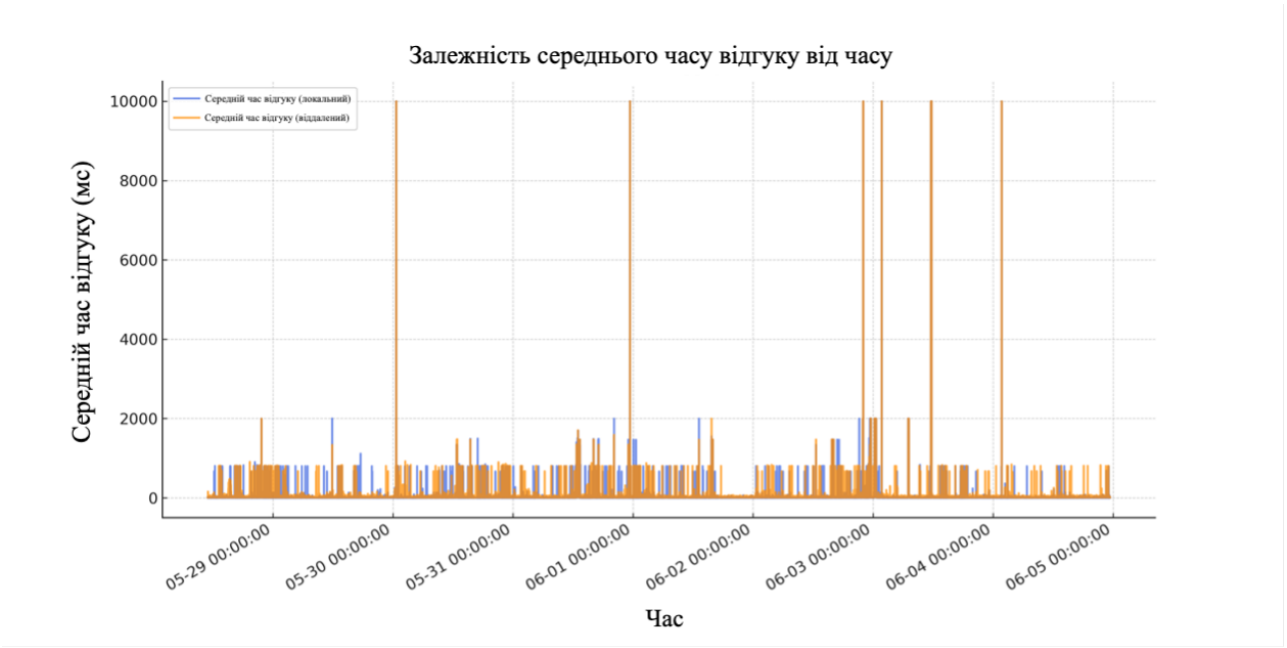



Рис. 4.1. Залежність середнього часу відгуку (затримки) від часу

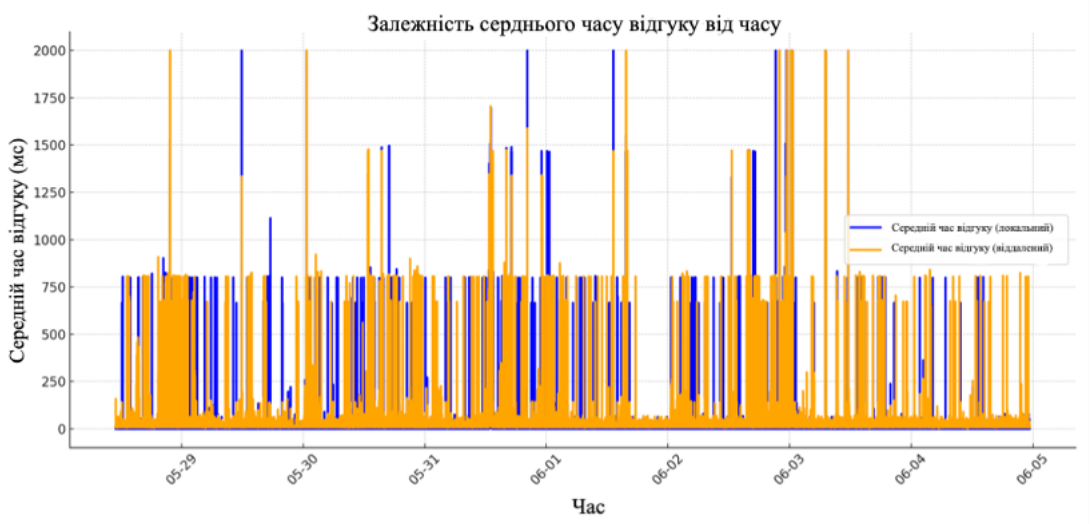



Рис. 4.2. Залежність середнього часу відгуку (затримки) від часу (очищені дані)

Для захоплення основних характеристик даних необхідна їх трансформація. Рисунок 4.3 — це комбінована ілюстрація, яка використовує гістограми та діаграми розмаху (boxplot) для відображення розподілу набору даних. Перша частина цього діаграмного зображення демонструє "Гістограму середнього локального часу відгуку" у порівнянні з "Діаграмою розмаху середнього локального часу відгуку"; друга частина показує "Гістограму середнього віддаленого часу відгуку" та "Діаграму розмаху середнього віддаленого часу відгуку". Гістограми наочно показують розподіл частот даних за певними інтервалами, тоді як діаграми розмаху надають графічне представлення квартилів даних, медіани та будь-яких викидів.

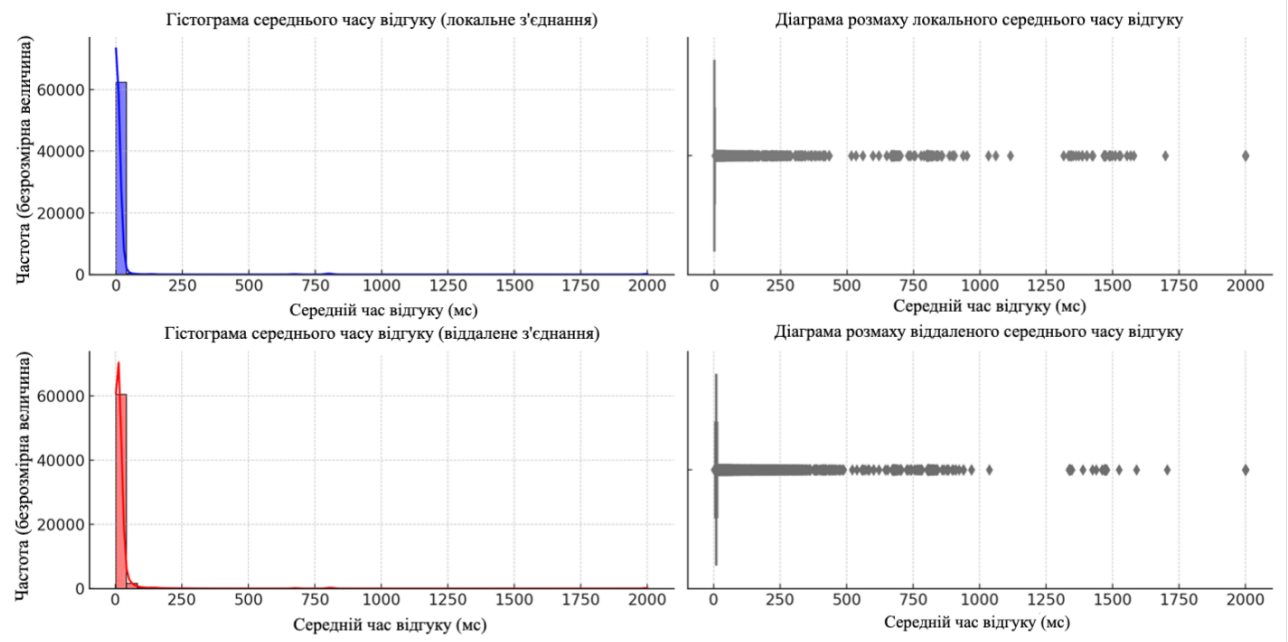



Рис. 4.3. Порівняльний аналіз середнього часу пінгу для локальних та віддалених з'єднань

Рисунок 4.3 також надає важливі уявлення, демонструючи оцінку щільності розподілу даних, що показує ймовірнісну щільність часу відгуку. Ширина форми на діаграмі змінюється відповідно до частоти спостережень: ширші області вказують на вищу концентрацію даних (вищу ймовірнісну щільність), а вужчі — на меншу.


Рис. 4.4. Графіки розподілу середнього часу пінгу для локальних та віддалених з'єднань

Після початкового аналізу стає зрозуміло, що дані демонструють концентрований діапазон нижчих часів пінгу як для локальних, так і для віддалених з'єднань, з наявністю викидів, що сягають вищих значень. Графіки показують різкий пік на нижчих мілісекундах, що вказує на швидке зниження частоти зі збільшенням часу пінгу. Однак досі не зрозуміло, якому розподілу відповідають дані; необхідно провести додаткові тести.

Для моделювання поведінки мережі потрібно зрозуміти розподіл даних; отже, необхідно провести більш детальний статистичний аналіз.

Наш аналіз розпочався з вилучення даних про мережеві затримки, зокрема середніх локальних (local_avg) та віддалених (remote_avg) затримок. Набір даних складався з вимірювань з численних вузлів у розподіленій мережі, що дало змогу отримати повне уявлення про варіації затримок. Для аналізу розподілу цих показників затримок ми застосували логарифмічне перетворення (рисунок 4.5), техніку, яка полегшує порівняння емпіричних розподілів даних з логнормальною моделлю.

Перетворені дані були візуалізовані за допомогою гістограм та графіків квантиль-квантиль (Q-Q), де перші ілюструють частотний розподіл, а другі дозволяють оцінити відповідність теоретичному нормальному розподілу. Як

дані про локальний середній час відгуку, так і віддалений середній час відгуку затримки після перетворення проявили характеристики, схожі на логнормальний розподіл. Зокрема, гістограми показали асиметрію, яка після перетворення більш гармонійно узгоджувалася із симетрією, очікуваною від нормального розподілу. Відповідно, графіки Q-Q підтвердили це спостереження, де точки даних здебільшого дотримувалися лінійності, що є ознакою нормального розподілу у логарифмічному просторі.

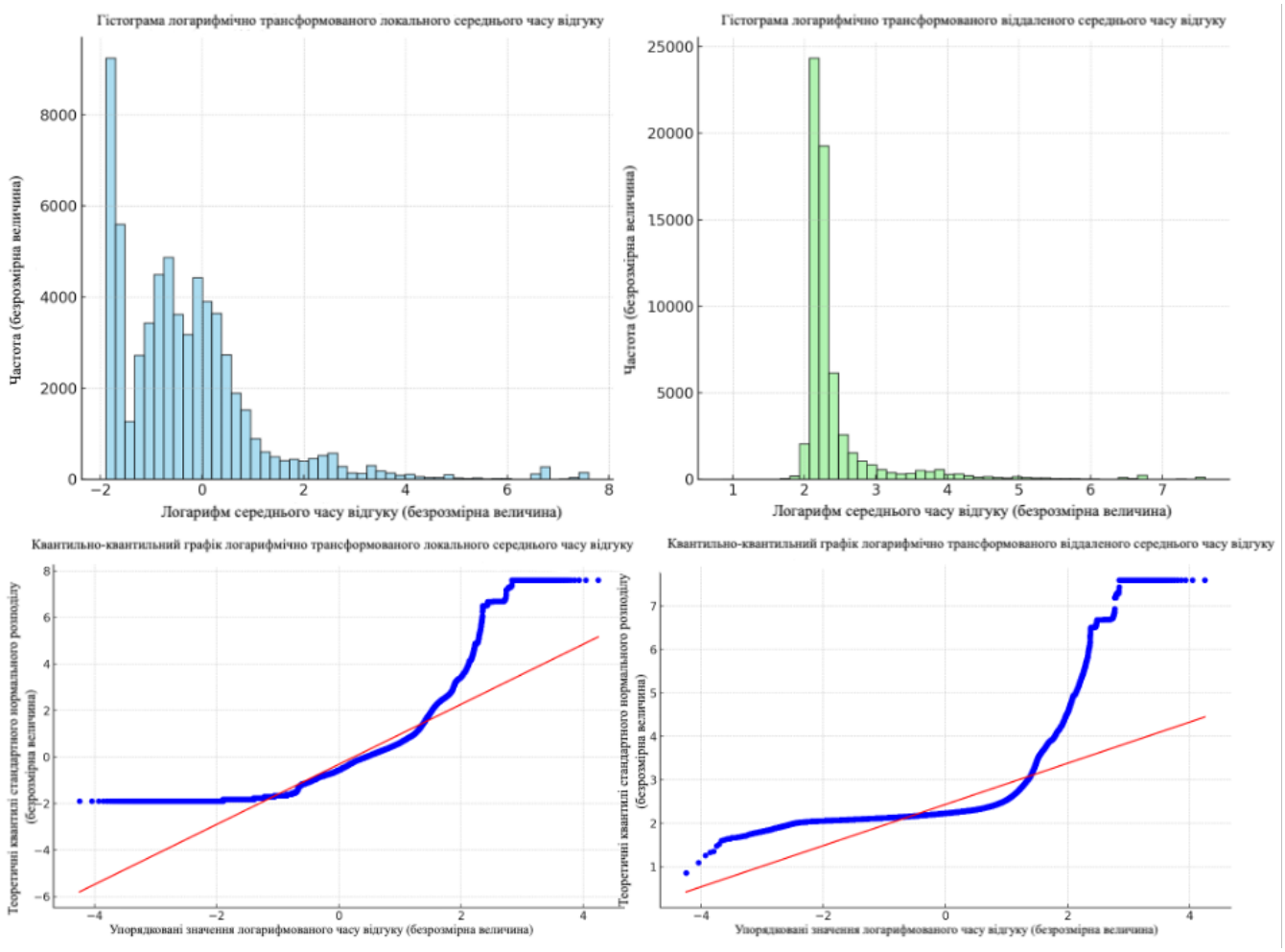



Рис. 4.5. Порівняльний статистичний аналіз логарифмованих значень часу пінгу для локальних та віддалених з'єднань: гістограми та квантильно-квантильні графіки

При оцінці нормальності розподілів даних у нашому аналізі тест Шапіро-Вілка є важливим статистичним інструментом. Цей тест оцінює, чи є вибірка з

нормально розподіленої популяції, використовуючи статистику тесту W, визначену формулою 4.1.

$$W = \frac{\left(\sum_{i=1}^{n} a_i x(i)\right)^2}{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2}, \tag{4.1}$$

де $n$ — це обсяг вибірки, $x_i$ позначає впорядковані значення вибірки, представляє середнє вибірки, а $n$ — коефіцієнти, отримані з очікуваних значень порядкової статистики для нормального розподілу. Значення статистики $X_i$, наближене до 1, свідчить про те, що дані добре апроксимуються нормальним розподілом, тоді як значення, значно менші за 1, вказують на відхилення від нормальності.

Застосувавши тест Шапіро-Вілка до набору даних, зокрема аналізуючи локальний середній час відгуку та віддалений середній час відгуку, тест показав значення статистики $W$, які суттєво відрізняються від значення, характерного для нормального розподілу. Для локального середнього часу відгуку статистика $i = 1, 2, ..., N$ вказує на значне відхилення від нормальності, підтверджуючи ненормальний характер розподілу даних. Аналогічно, статистика $P(X_i = 1)$ для віддаленого середнього часу відгуку підтверджує цей висновок, додатково вказуючи на ненормальний характер розподілу.

Незважаючи на суворе відхилення нормальності у логарифмічно перетворених даних, виявлене статистичними тестами, загальний аналіз свідчить про доцільність використання логнормального розподілу для апроксимації мережевої затримки у моделях алгоритмів консенсусу. Емпіричні дані, зокрема візуальний аналіз, підкреслюють потенціал використання параметрів логнормального розподілу для моделювання затримки в мережі. Ця апроксимація може значно підвищити точність моделі, пропонуючи більш реалістичне відображення мережевої динаміки у розподілених системах.

Включення логнормальної апроксимації мережевої затримки в моделі симуляцій може забезпечити дослідників та практиків надійним інструментом для прогнозування та аналізу ефективності алгоритмів консенсусу за різних мережевих умов. Майбутні дослідження можуть дослідити можливості удосконалення цих апроксимацій та їх вплив на ефективність консенсусу та стійкість системи.

Це дослідження прокладає шлях до глибшого розуміння динаміки мережевих затримок, закликаючи до статистично обґрунтованого підходу до моделювання в галузі алгоритмів розподіленого консенсусу.

### 4.1.2. Оцінка впливу параметру тайм-ауту на процес реконфігурації кластерів в розподіленій системі

По-перше, вибір відповідних вхідних параметрів моделі, таких як тайм-аут і стабільність мережі між вузлами, є критично важливим для точного оцінювання продуктивності системи. Визначення відповідного значення тайм-ауту у розподілених системах, особливо з урахуванням затримок у мережі, потребує прагматичного підходу, який враховує мінливість і непередбачуваність продуктивності мережі [130-136]. Мета полягає у встановленні тривалості тайм-ауту, яка б збалансовувала необхідність запобігання непотрібним виборам лідера через тимчасові мережеві збої з потребою у швидкому переключенні у разі дійсного виходу лідера з ладу. Це завдання вимагає постійної уваги від системних адміністраторів, оскільки воно залежить від конкретного стану системи та мережі в будь-який момент часу.

Загальноприйнятою стратегією для встановлення часу очікування вузлом повідомлення є визначення цього часу як максимальної спостережуваної мережевої затримки з додаванням резерву для компенсації нестабільностей та неочікуваних сплесків [137-142]. Цей буфер допомагає запобігти тимчасовим проблемам мережі, які можуть ініціювати перевибори лідера, що могло б дестабілізувати кластер і перешкоджати продуктивності. Однак, навіть із таким

буфером, система залишається вразливою до реконфігурації, оскільки стабільність мережі може змінюватися з часом [143-148]. У нашій моделі важливо оцінити, як система поводиться, коли продуктивність мережі поступово погіршується. Для дослідження цього встановимо час затримки для виборів як постійне значення 3475 мілісекунд (припускаючи, що кластер географічно розподілений) і виконуємо сценарій, де продуктивність мережі поступово знижується.

У кожній ітерації експерименту поступово збільшуватимемо верхню межу тривалості мережевих затримок, які відповідають логарифмічно-нормальному розподілу, на 1 мс, залишаючи нижню межу незмінною. Наприклад, якщо на першому кроці затримки варіюються від 1 до X мс, то на наступному — від 1 до *X*+1 мс, потім — від 1 до *X*+2 мс, і так далі. Це дозволяє моделювати поступове погіршення умов у мережі без зміни мінімального рівня затримки.

Такий підхід дає змогу спостерігати за реакцією системи та визначити критичну точку, коли починає відбуватися реконфігурація (наприклад, вибір нового лідера). Для кожного кроку експеримент буде повторено кілька разів з метою статистичної оцінки ймовірності реконфігурації.

У дослідженні, проведеному з метою розуміння динаміки поведінки кластера за певних умов мережі, була використана модель для симуляції відповіді кластера на змінні затримки мережі, зокрема, коли час очікування реконфігурації калібрується як 90% від максимальної межі затримки мережі. Обрані параметри для затримки мережі варіювалися від мінімуму 700 мс до максимуму 3475 мс. За результатами серії з 3000 експериментів, результати вказують на збіжність до значення 0,01, що свідчить про 1% ймовірності реконфігурації кластера з кожним повідомленням, відправленим між вузлами. Це відкриття закладає основу для прогностичного моделювання поведінки кластера.

Вже встановлено, що існує шанс (1%) реконфігурації кластера, коли час очікування реконфігурації встановлено на 0,9 від максимального ліміту

затримки мережі. Наступним кроком є дослідження поведінки кластера при коригуванні часу очікування реконфігурації в ширшому діапазоні, зокрема від 0,01 до 1 від максимального ліміту затримки мережі.

Дослідження впливу часу затримки реконфігурації на поведінку кластера передбачає детальний аналіз, особливо коли ці параметри встановлені як відсоток від максимально можливого часу затримки мережі. Це дослідження графічно представлено на рисунку 4, який слугує важливим інструментом для розуміння ймовірнісних наслідків реконфігурацій кластера. Графік використовує ось x для демонстрації діапазону множників часу очікування, від 0,01 (що вказує на мінімальну затримку) до 1 (що відображає повний обсяг максимальної затримки мережі), використаних для встановлення часу очікування реконфігурації. Ось y, у свою чергу, показує ймовірність виникнення реконфігурації кластера як прямий наслідок. Це візуальне представлення надає безцінні уявлення про те, як різні налаштування часу затримки реконфігурації відносно затримки мережі впливають на ймовірність реконфігурації кластера, пропонуючи глибше розуміння динаміки кластера у відповідь на мінливість мережі.

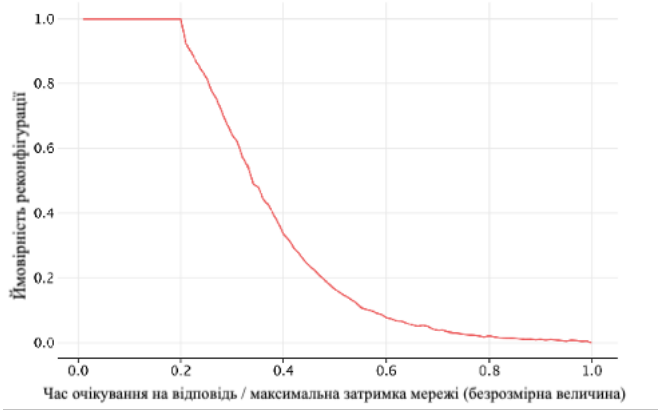



Рис. 4.11. Крива стабільності кластера з 5 вузлами

З кривої, яка показує різке зниження, а потім стабілізацію при збільшенні множника часу зартримки, можна зробити кілька висновків:

Висока чутливість при низьких множниках: Коли час затримки реконфігурації встановлено на дуже малу частку (близько 0,01) максимального

верхнього обмеження затримки мережі, ймовірність спричинення переконфігурації кластера становить майже 1 (або 100%). Це свідчить про те, що дуже короткі тайм-аути відносно затримки мережі дають високу ймовірність спричинити переконфігурацію.

Зменшення ймовірності зі збільшенням тайм-ауту: При збільшенні тайм-ауту переконфігурації, ймовірність переконфігурації кластера різко зменшується. Це очікувано, оскільки більш тривалі затримки дають кластеру можливість дочекатися повідомлення працездатності (heardbeat) від лідера.

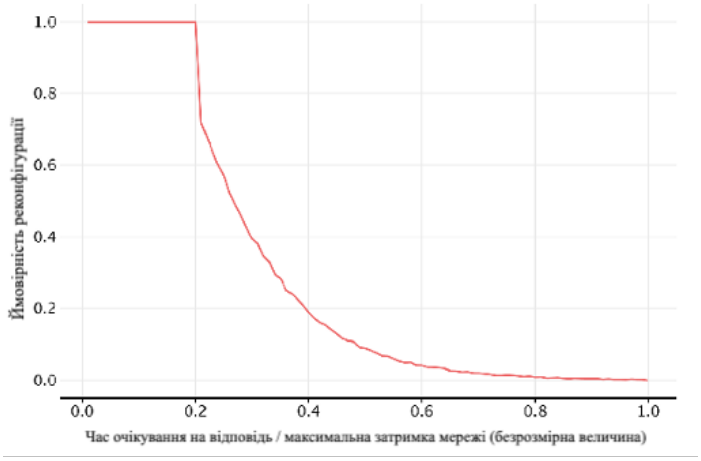
Стабілізація ймовірності:  Приблизно на тій точці, коли час затримки дорівнює максимальній верхній межі затримки мережі (множник 1), ймовірність переконфігурації стабілізується на дуже низькому значенні (близько 0,01 або 1%). Це свідчить про те, що коли час затримки встановлено на максимально очікувану затримку мережі, ймовірність непотрібних переконфігурацій є мінімальною.

У підсумку, графік показує, що встановлення занадто низького часу переконфігурації відносно затримки мережі може призвести до високої ймовірності переконфігурацій кластера, що може призвести до нестабільності. Навпаки, встановлення часу очікування близько до фактичної максимальної затримки мережі значно знижує ризик непотрібних переконфігурацій, що призводить до більш стабільної роботи кластера. Ця рівновага є критичною для забезпечення доступності та продуктивності кластера, особливо в розподілених системах, таких як бази даних або мікросервісні архітектури.

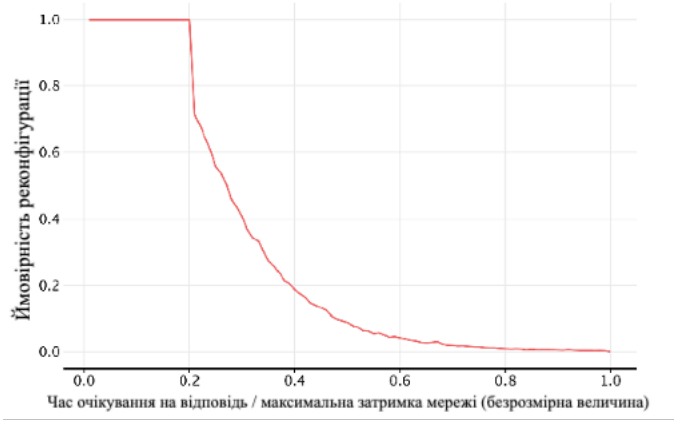
Для розширення розуміння динаміки, що відбувається в розподілених системах, корисно повторити вищезазначений експеримент на кластерах різних розмірів. Зокрема, дослідження кластерів з 4 та 3 вузлами відповідно (рисунок 4.12) дозволить отримати уявлення про масштабованість та надійність системи за різних конфігурацій.

Проведення аналізу на цих менших кластерах дозволить оцінити вплив часу переконфігурації відносно розміру кластера. Ймовірність переконфігурації

кластера та її чутливість до налаштувань часу очікування можуть змінюватися зі зміною кількості вузлів у кластері.




а)




б)

Рис. 4.12. Крива стійкості кластера з 4 вузлами (а) та 3 вузлами (б)

Початковий спад ймовірності все ще є різким, але менш крутим порівняно з кластером з 5 вузлами (рисунок 4), що може свідчити про те, що 4-вузловий кластер трохи менш чутливий до дуже коротких затримок.

Ймовірність переконфігурації також зменшується зі збільшенням часу затримки, аналогічно до 5-вузлового кластера.

Крива для 3-вузлового кластера показує менш крутий початковий спад порівняно з 5-вузловим та 4-вузловими кластерами, що свідчить про те, що 3-вузловий кластер ще менш чутливий до дуже коротких затримок. Існує

поступове зменшення ймовірності переконфігурації зі збільшенням множника часу затримки, слідуючи тому самому шаблону, що й інші графіки.

Для розуміння впливу змінності продуктивності мережі на стабільність кластера потрібен аналіз різних сценаріїв, в яких змінюються параметри мережі. Ці зміни відображаються на рисунку 4.13. Наступна візуалізація, надана моделлю, дозволяє оцінити стійкість кластера до умов мережі, які змінюються з часом. Вона моделює три сценарії деградації мережі, кожен зображений окремою кольоровою лінією. Ці лінії відображають зміну часу затримки повідомлень — від нижньої межі *x* до верхньої межі *y*. У процесі симуляції ці межі поступово зміщуються: у деяких сценаріях зростає лише верхня межа (погіршення найгірших умов), в інших — лише нижня межа (погіршення базової якості зв'язку), або обидві межі змінюються синхронно (глобальне погіршення каналу). Це дозволяє проаналізувати реакцію системи консенсусу на різні моделі деградації продуктивності мережі. Подальший аналіз траєкторії кожної лінії надає уявлення про стійкість конфігурацій кластера в змінних мережевих умовах.

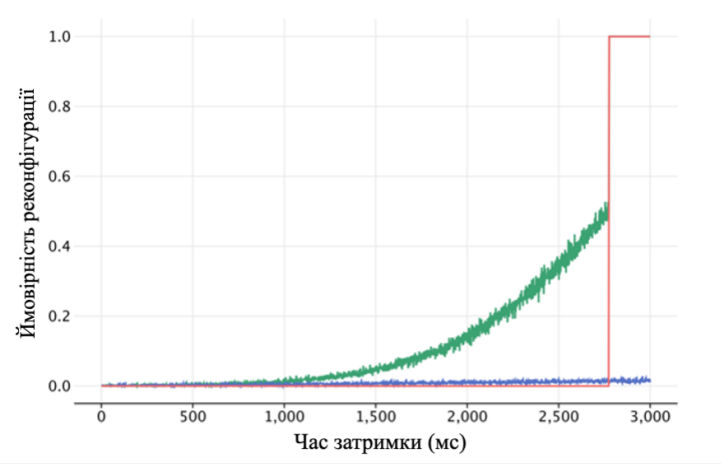



Рис. 4.13. Вплив змінності обмежень продуктивності мережі на ймовірність переконфігурації кластера, де *а* (зелена лінія) - погіршення обох обмежень, *б* (синя лінія) - погіршення верхнього обмеження, *в* (червона лінія) - погіршення нижнього обмеження

На 4.13 зелена лінія зображає сценарій, де як верхнє, так і нижнє обмеження продуктивності мережі погіршуються. Синя лінія представляє ситуацію, коли погіршується лише верхнє обмеження, а червона лінія вказує на випадок, коли погіршується лише нижнє обмеження. Ця візуалізація є ключовою для розуміння впливу змінності мережі на стабільність кластера.

Погіршення обох обмежень. Цей випадок представляє сценарій, коли як верхнє, так і нижнє обмеження продуктивності мережі погіршуються. Протягом значної тривалості ймовірність переконфігурації залишається низькою, що свідчить про стійкість кластера до погіршення продуктивності мережі. Однак, з погіршенням стабільності мережі відбувається драматичне зростання ймовірності переконфігурації. Цей раптовий ріст свідчить про те, що коли погіршення мережі досягає певної критичної точки, воно сильно впливає на кластер, можливо через нездатність ефективно спілкуватися, що призводить до ймовірної переконфігурації.

Погіршення верхньої межі. У цьому випадку вказується сценарій, коли лише верхня межа продуктивності мережі погіршується. Протягом графіку синя лінія залишається близькою до горизонтальної вісі, що свідчить про дуже низьку ймовірність переконфігурації. Це свідчить про те, що кластер досить терпимий до погіршення верхньої межі продуктивності, що відповідає збільшеним затримкам або зниженій пропускній здатності, які не мають безпосереднього впливу на роботу кластера.

Погіршення нижньої межі. Цей випадок представляє ситуацію, коли тільки нижня межа продуктивності мережі погіршується. На відміну від зеленої лінії, червона лінія показує різке зростання ймовірності реконфігурації в останній момент, яке є майже вертикальним. Це вказує на те, що кластер працює надійно до раптового падіння стабільності. Цей сценарій є цілком очікуваним, оскільки коли затримка в мережі перевищує час затримки реконфігурації, реконфігурація кластера стає неминучою. Тим не менше, порівняння результату з погіршенням

верхньої межі показує, що кластер більш стійкий до погіршення нижньої межі, хоча це і не є вирішальним.

Рисунок 4.13 ілюструє, що кластер може в значній мірі впоратися з індивідуальним зниженням продуктивності мережі. Однак кластер вразливий до одночасного зниження як верхньої, так і нижньої меж продуктивності мережі, що призводить до високої ймовірності реконфігурації.

Дослідимо складні закономірності ймовірностей реконфігурації кластерів, коли мережі зіштовхуються зі зростаючою кількістю нестабільних з’єднань. Дослідження прагне виявити конкретні точки, в яких ймовірність самостійної реконфігурації кластерів різко зростає, підкреслюючи складні взаємозв'язки.

На рисунку 4.14 показано поведінку кластера при збільшенні кількості зв'язків які мають велику затримку, зелена крива представляє кластер, що постраждав від одного компрометованого зв'язку. Ця крива показує зниження продуктивності мережі як у верхній, так і у нижній межах. Світло-зелена крива позначає кластер з двома компрометованими зв'язками. Жовта крива представляє кластер з трьома пошкодженими зв'язками. Фіолетова крива представляє кластер з чотирма несправними зв'язками. Нарешті, червона крива зображає сценарій, де кожен зв'язок у кластері є дефектним, що впливає не тільки на комунікацію від лідера до підлеглих, але на всю комункіцію загалом.

Аналізуючи графік, можна виділити наступні ключові висновки щодо ймовірності реконфігурації кластера, коли кількість скомпрометованих зв'язків зростає:

Стійкість до початкових відмов. Кластер проявляє певну стійкість до початкового погіршення мережі. На початкових етапах, навіть з одним або двома пошкодженими зв'язками (зелена та світло-зелена крива), ймовірність переконфігурації залишається низькою, що свідчить про здатність витримувати часткові проблеми мережі без негайної переконфігурації.

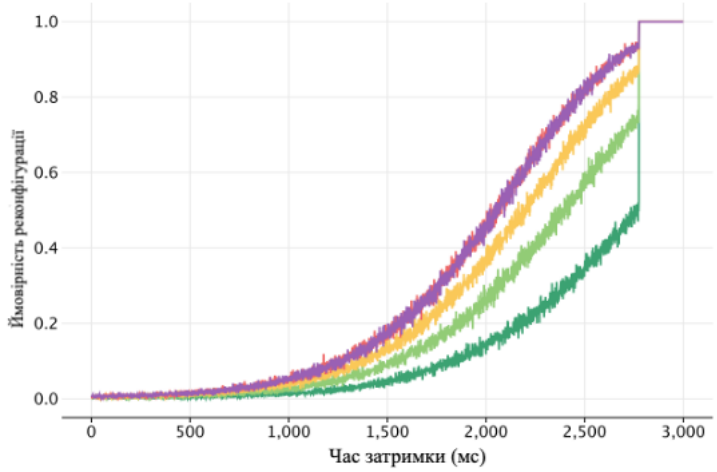



Рис. 4.14. Криві стабільності кластера, де а (зелена крива) - один пошкоджений зв'язок, б (світло-зелена крива) - два пошкоджених зв'язки, в (жовта крива) - три пошкоджені зв'язки, г (фіолетова крива) - чотири пошкоджені зв'язки, д (червона крива) – всі звяки пошкоджені.

Нелінійне збільшення ймовірності переконфігурації. Кожна крива представляє додатковий збій зв'язку та зростання ймовірності реконфігурації, однак темп зростання не є сталим. Початковий збій (зелена крива) спричиняє значнішу зміну в ймовірності, ніж наступні (від світло-зеленого до фіолетового), що свідчить про сповільнювальний ефект із накопиченням кількості ушкоджених зв'язків. Це вказує на зменшення впливу кожного додаткового несправного зв'язку на схильність мережі до реконфігурації, підкреслюючи більш складну та різнопланову взаємодію між збоями зв'язків та стабільністю кластера, ніж простий лінійний чи прискорений приріст.

Поріг терпимості. Існує пороговий рівень цілісності мережі, нижче якого ймовірність реконфігурації швидко наближається до одинці.

Відкладена неминучість. Незважаючи на кількість розірваних зв'язків, існує стійкий шаблон, за яким висока ймовірність реконфігурації відкладається.

Ці висновки підкреслюють критичну важливість підтримання цілісності мережі в розподілених системах. Вони також демонструють важливість проектування кластерів з міцними стратегіями відмовостійкості та

реконфігурації для управління збільшеною ймовірністю реконфігурації, коли посилюються проблеми в мережі.

Загальне дослідження впливу нестабільних мережевих з'єднань на кластери, що виконують алгоритми консенсусу, привело до кількох ключових висновків:

Час затримки виборів проти затримок мережі. Ймовірність непотрібних виборів лідера значно залежить від налаштування часу очікування виборів у порівнянні з мережевими затримками. Короткі інтервали часу затримки, у порівнянні з варіативністю мережевих затримок, призводять до високої ймовірності спричинення реконфігурацій кластера. Навпаки, узгодження часу очікування виборів більш точно з максимальною спостережуваною мережевою затримкою знижує ймовірність зайвих перевиборів лідера, стабілізуючи систему.

Чутливість стабільності мережі. Ефективність роботи кластера високо чутлива до змін у стабільності мережі. Хоча система демонструє стійкість до ізольованих проблем з продуктивністю мережі, нелінійне зростання ймовірності реконфігурації кластера відбувається, коли нестабільність мережі стає більш явною. Це вказує на критичний поріг продуктивності мережі, нижче якого стабільність системи суттєво підірвана.

Вплив скомпрометованих мережевих зв'язків. Реакція системи на скомпрометовані мережеві зв'язки виявляє міцну толерантність до окремих випадків погіршення (верхня або нижня межа мережевих затримок). Однак, одночасне погіршення на обох верхніх та нижніх меж продуктивності помітно збільшує ймовірність реконфігурації кластера. Це підкреслює критичну роль підтримки продуктивності мережевих зв'язків для стабільності кластера.

Розмір кластера проти ймовірності реконфігурації. Експерименти також показали, що кількість вузлів впливає на загальну стабільність системи, де менша кількість вузлів демонструє знижену чутливість до дуже коротких затримок реконфігурації. Точна ймовірність реконфігурації варіюється залежно

від розміру кластера, що вказує на те, що оптимальні налаштування часу затримки можуть відрізнятися залежно від кількості вузлів.

Нелінійний зв'язок між деградацією мережі та ймовірністю переконфігурації. Зі зростанням кількості компрометованих мережевих зв'язків збільшується ймовірність переконфігурації, хоча кожна зкомпрометована ланка мережі має менший вплив на стабільність системи. Ці висновки підкреслюють необхідність ретельного врахування мережевих умов та конфігурації кластера при проектуванні та експлуатації розподілених систем, які використовують алгоритми консенсусу. Розуміння балансу між налаштуванням часу затримки виборів та змінністю затримок мережі, разом з впливом розміру кластера та продуктивності мережевих зв'язків, є важливим для забезпечення стабільності та ефективності системи.

Розмір кластера також впливає на його чутливість до нестабільностей мережі та загальну стабільність. Менші кластери менш чутливі до коротких інтервалів очікування виборів порівняно з більшими кластерами, що вказує на те, що складність досягнення консенсусу між вузлами впливає на реакцію системи на затримки в мережі. Однак, невідповідні налаштування інтервалів очікування можуть дестабілізувати кластери будь-якого розміру.

Реакція на порушення меж затримок у мережі підкреслює значну толерантність системи до окремих випадків погіршення. Проте одночасне погіршення показників продуктивності помітно збільшує ймовірність переконфігурації кластера.

### 4.1.3. Дослідження ефективності функціонування розподіленої сервісної системи в умовах змінної працездатності мережі з використанням автономного адаптивного методу

Автономний адаптивний метод, представлений у підрозділі 2.3 представлений тут на основі розробленої імімтаційної моделі, по суті є автономною системою моніторингу відмов, яка потребує лише початкового

налаштування з боку адміністратора. Після цього вона функціонує без додаткового втручання, автоматично реагуючи на зміни показників мережі відповідно до визначених параметрів. Це дозволяє зменшити обчислювальне навантаження, оскільки система працює виключно з відомими значеннями затримок у кластері, що робить її ефективною навіть за обмежених ресурсів.

Ключовою особливістю цього методу є незалежність вузлів, що значно підвищує стійкість системи до збоїв. У разі виникнення відмови окремих компонентів система продовжує функціонувати без втрати загальної працездатності. Більше того, кластер має здатність до самонавчання й адаптації, поступово оптимізуючи свою роботу для досягнення максимальної продуктивності.

Запропонований підхід також забезпечує гнучке управління розподілом лідерських ролей між вузлами. Система надає пріоритет тим вузлам, які демонструють найвищу продуктивність на поточний момент, враховуючи стан з'єднань у кластері. Це створює додаткові переваги для ефективного використання ресурсів та сприяє динамічному балансуванню навантаження в розподіленій системі.

Загалом, метод дозволяє досягти високого рівня надійності, ефективності та адаптивності в умовах постійних змін і можливих збоїв у розподілених мережевих середовищах.

У попередньому пункті дисертаційного дослідження проаналізовано роботу кластера без застосування адаптивного підходу (рисунок. 4.15). На графіку показано зміну поведінки кластера у випадках, коли зростає кількість зв’язків із високою затримкою. Наприклад, зелена крива ілюструє кластер, що постраждав від одного компрометованого зв’язку, тоді як світло-зелена крива демонструє вплив двох таких зв’язків, і так далі.

Аналіз показує, що з кожним новим компрометованим зв’язком продуктивність і стабільність кластера суттєво знижуються. Це зумовлено тим, що кожен додатковий повільний зв’язок підвищує імовірність реконфігурації

кластеру. Унаслідок цього зменшується загальна ефективність роботи кластера та зростає час виконання операцій, що є критичним у контексті розподілених систем.

Результати такого аналізу підкреслюють необхідність впровадження адаптивного підходу, здатного ефективно реагувати на зростаючу кількість зв’язків із високою затримкою, мінімізуючи їхній вплив на продуктивність системи а також своєчасно реагувати на важливі збої в кластері та своєчасно реагувати на критичні зміни затримки в мережі [85].

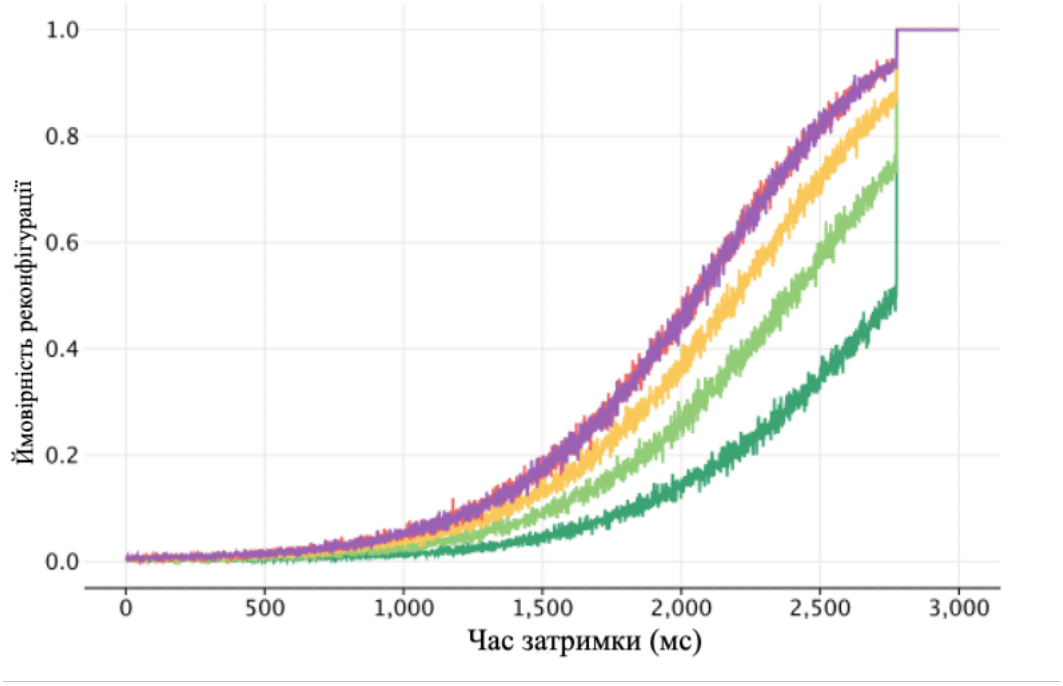



Рис. 4.15. Криві стабільності кластера, де а (зелена крива) - один пошкоджений зв'язок, б (світло-зелена крива) - два пошкоджених зв'язки, в (жовта крива) - три пошкоджені зв'язки, г (фіолетова крива) - чотири пошкоджені зв'язки, д (червона крива) – всі звяки пошкоджені.

Попри наведені спостереження, можна окреслити ключові обмеження традиційного підходу. Зокрема, у всіх випадках час очікування вузла на отримання повідомлення працездатності (heardbeat) залишається статичним і задається адміністратором лише один раз під час налаштування. Протягом подальшої роботи кластера це значення не змінюється, що значно обмежує здатність системи адаптуватися до змін у мережевих умовах.

Наприклад, якщо одне зі з’єднань на короткий проміжок часу, навіть на декілька мілісекунд, перевищить встановлений поріг затримки, це неминуче спричинить переобрання нового лідера в кластері. Враховуючи, що такі перевищення затримки часто є тимчасовими, подібна ситуація може суттєво вплинути на загальну продуктивність системи. Одне ненадійне з’єднання здатне викликати каскадні наслідки для працездатності всього кластера.

У контексті великих розподілених систем, які можуть складатися з тисяч вузлів і з’єднань, така ситуація є неприпустимою. Це зумовлює необхідність розробки адаптивних механізмів, що дозволяють змінювати параметри роботи системи в реальному часі, враховуючи поточний стан мережі, та уникати надмірного реагування на короткочасні збої.

Другою суттєвою проблемою є непередбачуваність у процесі вибору лідера. Традиційні підходи надають усім вузлам рівні шанси бути обраними новим лідером, не враховуючи поточного стану мережі та продуктивності кожного з вузлів.

Такий підхід може призводити до ситуацій, коли лідером стає вузол із менш стабільним або повільним з’єднанням, що, у свою чергу, негативно впливає на загальну продуктивність і стійкість системи. Наприклад, якщо новий лідер має високі затримки у зв’язку з іншими вузлами, це може створити вузьке місце в обробці запитів і уповільнити роботу всього кластера.

Ця проблема підкреслює важливість впровадження механізму вибору лідера, який враховує поточний стан мережі, затримки між вузлами та їхню здатність ефективно обробляти запити. Такий підхід забезпечив би більшу надійність і оптимальність роботи системи, дозволяючи кластеру швидко адаптуватися до змін і підтримувати стабільність навіть у складних умовах.

Тепер розглянемо схожу конфігурацію кластеру але з застосуванням розробленого підходу, де аналогічно до попереднього рисунку, зелена крва – представляє один пошкоджений зв'язок в кластері, світло-зелена – два, жовта – три, червона - всі звязки в кластері.

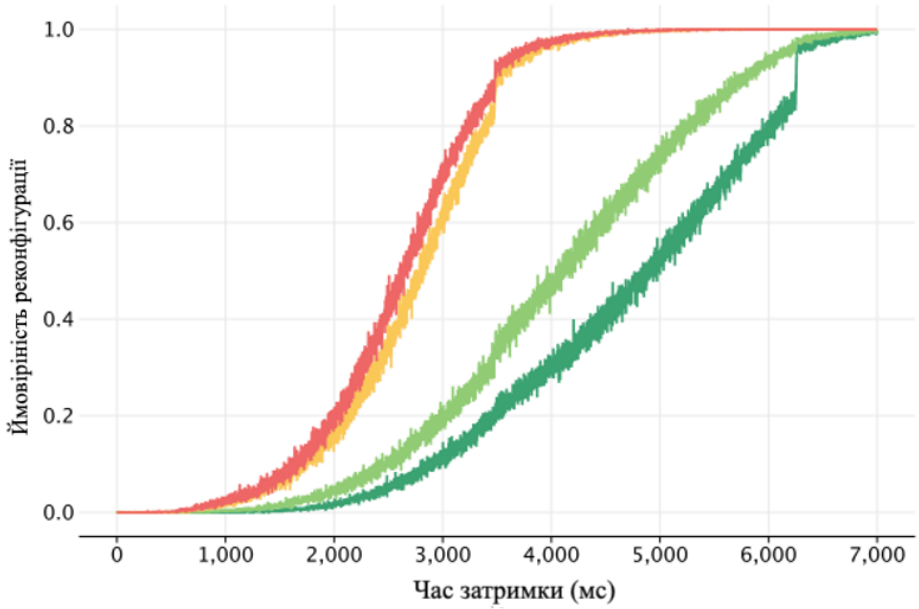



Рис. 4.16. Криві стабільності кластера,який використовує адаптивний метод визначення затримки, де а (зелена крива) - один пошкоджений зв'язок, б (світло-зелена крива) - два пошкоджених зв'язки, в (жовта крива) - три пошкоджені зв'язки, г (фіолетова крива) - чотири пошкоджені зв'язки, д (червона крива) – всі звяки пошкоджені.

Як видно з рисунку ситуація кардинально інша, і перше варто відмітити це вісь *x*, яка позначає час затримки, в цьому випдку вона значно перевизщує попередньу і обмежена занченням $T_{max}$ що встановлює верхній ліміт часу очікування вузла на повідомлення працездатності (heardbeat) від лідеру, більше того це значення може бути встановлене до безмежності у випадку якщо адміністаратор хоче зменшити вплив поодиноких обривів звязку (тут варто відмітити що у цьому випадку система зупиняється доти доки зв'язок не буде відновлено) [109].

Розглянемо це детальніше, з рисунку видно що зелена та світло зелена крива є більш горизонтальною що свідчить про те що у випадку відмов 1 чи 2 звязків у кластері це не суттєво впливає на імовірність реконфігурації.

Число цих кривих $k$, що відображає максимальну кількість зв'язків, які можуть відмовити без істотного впливу на ймовірність реконфігурації, їх можна описати формулою:

$$k = n - q \tag{4.2}$$

де $n$ — загальна кількість вузлів у кластері, $q$ — мінімальна кількість вузлів, що формують кворум для підтримки стабільності системи.

Для формального визначення $k$, необхідно звернутися до принципу обчислення кількості вузлів, що формують кворум. Кворум визначається як мінімальна кількість вузлів, необхідна для забезпечення більшості в системі. Іншими словами, $q$ — це кількість вузлів, що гарантує здатність системи досягати консенсусу, навіть за наявності відмов окремих компонентів та визначається формулою:

$$q = \left\lfloor \frac{n}{2} \right\rfloor + 1 \tag{4.3}$$

Виходячи з попердньої формули $k$ можна виразити як:

$$k = n - \left( \left\lfloor \frac{n}{2} \right\rfloor + 1 \right) \tag{4.4}$$

Або у спрощеній формі:

$$k = \frac{n-1}{2} \tag{4.5}$$

У цьому контексті зелена та світло-зелена крива на графіку демонструють, що навіть при відмовах $k = 1$ чи $k = 2$ звязків, система все ще здатна уникати реконфігурацій.

Щоб підтвердити цю гіпотезу збільшимо кластер до 8 вузлів та сформуємо новий графік.

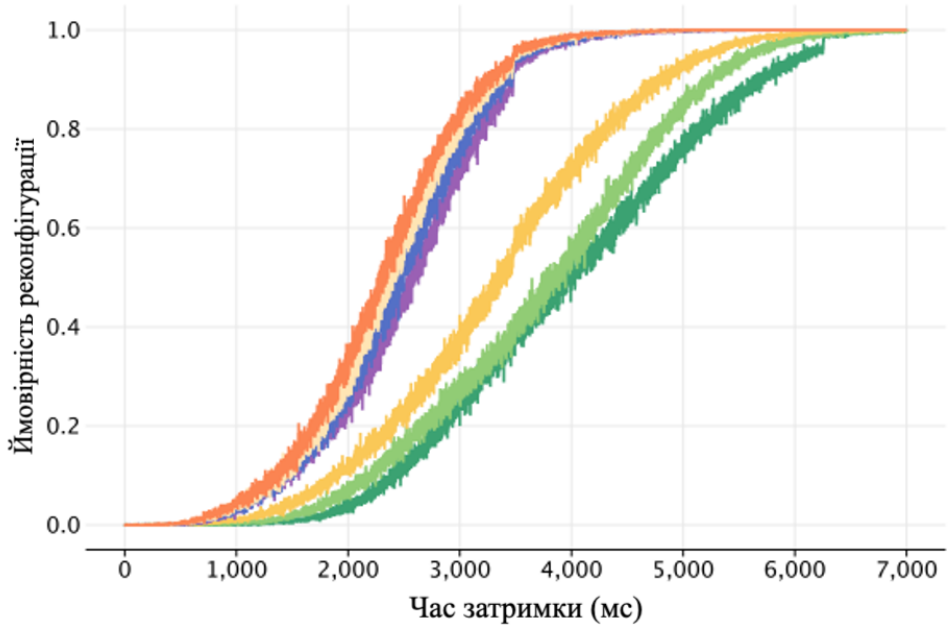



Рис. 4.17. Криві стабільності кластера з 8 вузлами, який використовує адаптивний метод визначення затримки

Рисунок 4.17 показує, що збільшення кластеру підвищує стійкість до відмов згідно з формулою 4.5. Зелені та світло-зелені криві демонструють незначний вплив відмов одного або двох з'єднань, що підтверджує високу стійкість системи. Жовта крива, хоча й вказує на дещо більший вплив на стабільність, все одно залишаються в межах прийнятних показників затримки, що дозволяє уникнути реконфігурації.

Відповідно до формули 4.5, для 8 вузлів система допускає відмову до трьох з'єднань ($k$=3) без втрати стабільності. Це підтверджується більш пологим характером кривих навіть при зростаючій кількості пошкоджених з'єднань.

Оцінивши покращення в процентах, порівнюючи допустиму кількість збоїв до та після впровадження адаптивного алгоритму можна констатувати покращення на 200%:

$$ПокращенняСтійкості = \left( \frac{k_{новий} - k_{старий}}{k_{новий}} \right) \times 100\% = \left( \frac{3-1}{1} \right) \times 100\% = 200\% \qquad (4.6)$$

Візуалізуючи формулу 4.5, відслідковується її лінійність з ростом кластеру.

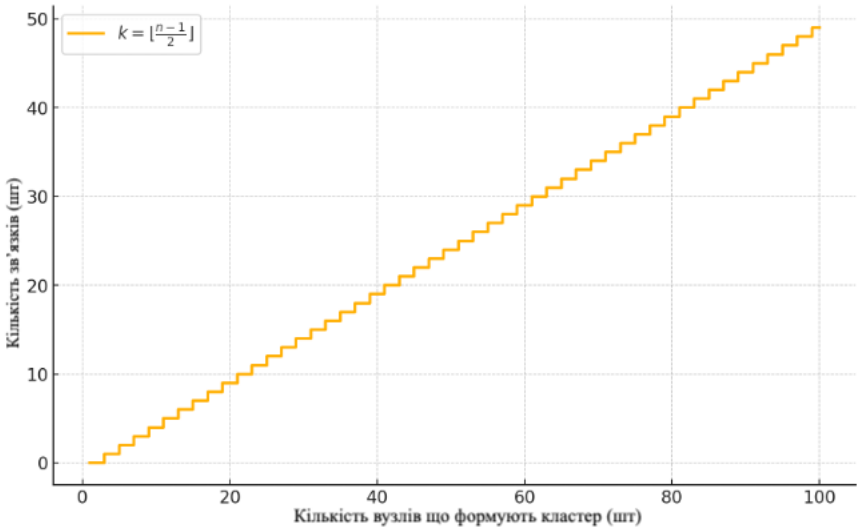



Рис. 4.18. Графік залежності $k$ (кількість зв'язків у кластері, які можуть вийти з ладу без втрати його працездатності) від $n$ (кількість вузлів, що формують кластер).

Таким чином, адаптивний підхід значно підвищує стійкість кластеру до мережевих збоїв, дозволяючи системі налаштовуватися на оптимальну роботу, мінімізуючи кількість лідерських переобрань і враховуючи стан кожного з'єднання. Це дозволяє не тільки уникати надлишкових реконфігурацій, але й збільшувати загальну ефективність системи.

Також ключовим моментом є значення $T_{max}$, що представляє верхній ліміт часу очікування вузла на отримання повідомлення працездатності (heardbeat) від лідера. Збільшення $T_{max}$ дозволяє системі бути більш стійкою до короткочасних затримок чи обривів зв'язку, оскільки зменшується ймовірність помилкових реконфігурацій через тимчасові збої в комунікації.

- Якщо $T_{max}$ встановлено занадто малим, навіть короткочасні затримки можуть викликати хибні спрацьовування механізмів вибору нового лідера.
- Якщо $T_{max}$ встановлено великим (аж до теоретичного безмежного значення), кластер стає більш толерантним до зв'язкових збоїв, але зростає час реакції на реальні відмови лідера.

Зелена і світло-зелена крива на графіку свідчать, що для малих $k$ (1 чи 2 відмови) система демонструє низьку ймовірність реконфігурації навіть за значного $T_{max}$.

Також просте порівнявши значення ймовірностей реконфігурації при у фіксованій точці часу 2500 мс (при наявності 1 погано функціонуючого звязку) можна побачити що ймовірність реконфігурації в адаптивному випадку значно нища та становить ~0.05 в порівнянні з ~0.38 для неадаптивного методу що наочно показує перевагу адаптивного методу та можливість ігнорувати незначні збої.

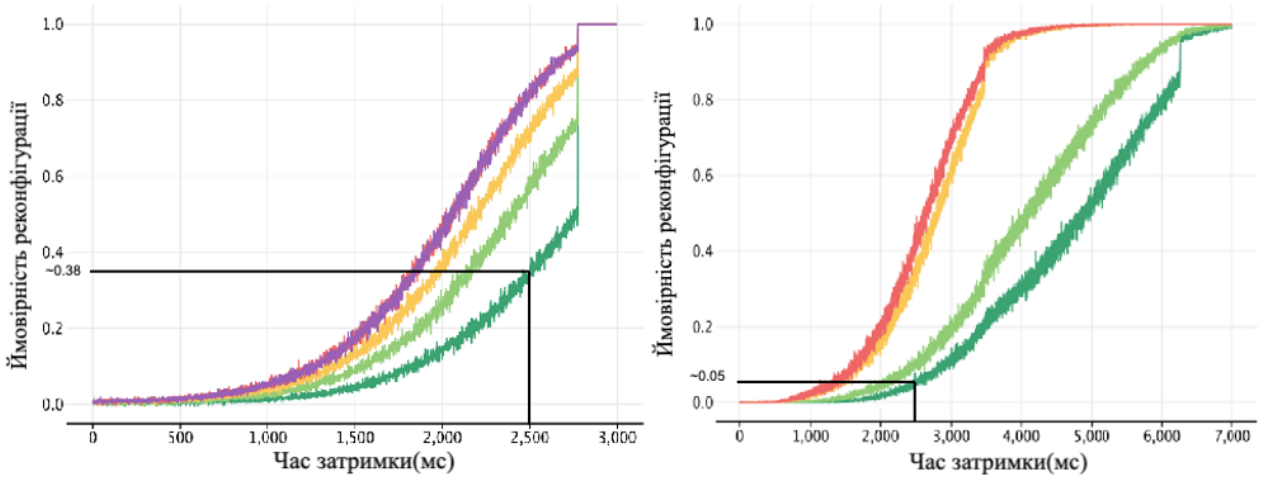



Рис. 4.19. Криві стабільності кластера до впроводження адаптивного методу (лівий грфік) та після впровадження (правий графік), де вісь $y$ представляє ймовірність реконфігурації сласетру а вісь $x$ – час затримки.

Неадаптивний метод має фіксований час очікування перед оголошенням вузла мертвим, що може призводити до зайвих реконфігурацій. Натомість адаптивний метод регулює цей час залежно від стану системи: при некритичних затримках відповіді він відкладає реконфігурацію, а при надмірному збільшенні часу очікування та кількості погано функціонуючих з'єднань — різко її активує. Тому важливо аналітично підтвердити цей факт.

Для оцінки ефективності методів використовується площа під кривою (AUC, Area Under Curve), яка визначає сумарну ймовірність реконфігурації залежно від часу затримки:

$$AUC = \int_{X_{\min}}^{X_{\max}} P(x)\,dx \tag{4.7}$$

Ця формула описує інтеграл від функції ймовірності реконфігурації P(x) прадесталеної на рисунку 4.19 в межах певного діапазону затримок [$x_{min}$, $x_{max}$].

Оскільки методи можуть працювати з різними діапазонами затримки, вводиться нормалізоване значенн $AUC_{norm}$:

$$AUC_{norm} = \frac{AUC}{x_{\max} - x_{\min}} \tag{4.8}$$

Ця метрика дозволить коректно порівнювати загальний вплив методів незалежно від тривалості процесу.

Результати числового аналізу демонструють (таблиця 4.1), що адаптивний метод значно зменшує ймовірність реконфігурації при малій кількості непрацюючих зєднань, але здійснює різкий стрибок при досягненні критичного значення кількості погано працюючих зєднань.

Під критичною кількістю погано працюючих зєднань мається на увазі така мережева ситуація, коли кількість скомпрометованих з'єднань досягає або перевищує поріг (визначений формулою 4.5), достатній для втрати кворуму — тобто, коли система вже не може гарантувати узгодженість або виконання консенсусу без реконфігурації (наприклад, у кластері з 5 вузлів — при виході з ладу 3 або більше з них).

Таблиця 4.1

Сумарне значення ймовірності реконфігурації залежно від часу затримки

| Випадок | *AUC* (загальна площа) | Нормалізований *AUC* |
|---|---|---|
| Неадаптивний, 2 вузли | 1031.36 | 0.385 |
| Адаптивний, 2 вузли | 3460.04 | 0.495 |
| Неадаптивний, 3 вузли | 1213.50 | 0.441 |
| Адаптивний, 3 вузли | 3476.64 | 0.497 |

Щоб оцінити, наскільки ефективно адаптивний метод реагує при переході від некритичного до критичного стану, вводиться коефіцієнт порогової

адаптивності $A_T$ (формула 4.9). Перехід до критичного стану є умовою, за якої реконфігурація дійсно необхідна, тому важливо, щоб метод точно розпізнавав цей момент і не реагував передчасно.

$$A_T = \frac{\Delta P_{adaptive}}{\Delta P_{non-adaptive}} \tag{4.9}$$

Де $\Delta P$ — стрибок ймовірності реконфігурації між двома сусідніми станами:

$$\Delta P_{adaptive} = P_{critical}^{adaptive} - P_{pre-critical}^{adaptive} \tag{4.10}$$

$$\Delta P_{non-adaptive} = P_{critical}^{non-adaptive} - P_{critical}^{non-adaptive} \tag{4.11}$$

Розрахунки показують, що:

$$P_{pre-critical}^{non-adaptive} = 0.42996, \quad P_{critical}^{non-adaptive} = 0.44075 \tag{4.12}$$

$$P_{pre-critical}^{adaptive} = 0.10198, \quad P_{critical}^{adaptive} = 0.4966 \tag{4.13}$$

$$\Delta P_{non-adaptive} = 0.44075 - 0.42996 = 0.01079 \tag{4.14}$$

$$A_T = \frac{0.39462}{0.01079} \approx 0.3657 \tag{4.15}$$

Це означає, що адаптивний метод змінює свою поведінку у 36.5 разів сильніше при досягненні критичної кількості відмов в кластері, ніж неадаптивний.

Для кількісного оцінювання ефективності адаптивного методу як у стабільному, так і в критичному режимах, розглянемо абсолютну різницю ймовірностей реконфігурації у відповідних станах. У некритичних умовах — тобто коли кількість скомпрометованих з’єднань менша за критичну — адаптивний метод демонструє значне зменшення ймовірності передчасної реконфігурації:

$$\Delta P_{pre-critical} = P_{non-adaptive}^{pre-critical} - P_{adaptive}^{pre-critical} = 0.42996 - 0.10198 = 0.32798 \tag{4.16}$$

Де $\Delta P_{pre-critical}$ показує зменшення ймовірності на приблизно на 33%. Така поведінка свідчить про високу стійкість системи до незначних або короткочасних збоїв.

У критичному режимі, навпаки, адаптивний метод демонструє кращу чутливість до реальної загрози, підвищуючи ймовірність реконфігурації:

$$\Delta P_{critical} = P_{adaptive}^{critical} - P_{non-adaptive}^{critical} = 0.4966 - 0.44075 = 0.05585 \quad (4.17)$$

Де $\Delta P_{critical}$ показує покращення на 5.6% у порівнянні з неадаптивним підходом. Таким чином, метод не лише утримується від надмірної реактивності за нормальних умов, але й своєчасно активується у момент загрози, що робить його поведінку ефективно збалансованою.

Тож адаптивний метод демонструє покращення поведінки системи як у стабільних, так і в критичних мережевих умовах. Зокрема, у некритичному стані ймовірність передчасної реконфігурації зменшується на приблизно 33% порівняно з неадаптивним підходом, що дозволяє уникати зайвих втручань при короткочасних збоях. Водночас у критичному стані, коли кількість скомпрометованих з'єднань досягає порогу втрати кворуму, адаптивний метод забезпечує додаткове підвищення ймовірності реконфігурації на 5.6%, своєчасно реагуючи на загрозу і підтримуючи надійність кластера.

Адаптивний метод ефективніший, оскільки мінімізує зайві реконфігурації при малих збоях, але швидко активується при небезпечному рівні відмов. Нормалізовані значення AUC підтверджують, що він забезпечує стабільну ефективність навіть при збільшенні кількості відмов. Це підтверджує, що адаптивний метод є оптимальним підходом до реконфігурації кластерних систем.

Тепер доведемо математично, що алгоритм вирішує проблему непередбачуваності вибору лідера, можна виконати шляхом формального аналізу його властивостей.

Для демонстрації того, що запропонований алгоритм вирішує проблему непередбачуваності вибору лідера, необхідно виконати формальний аналіз його

роботи, а також показати залежність результатів від стану мережі. Основною метою є доведення, що алгоритм забезпечує пріоритетність для вузлів із кращими характеристиками та мінімізує ймовірність вибору вузлів із низькими мережевими показниками.

Алгоритм передбачає, що кожен вузол $i$ має визначений час очікування $T_i$, після якого він ініціює процедуру вибору лідера. Чим менше це значення, тим раніше вузол почне ініціацію, і відповідно — має вищий шанс бути обраним. Це формалізується через співвідношення:

$$P_i \propto \frac{1}{T_i} \tag{4.18}$$

де $P_i$ — ймовірність того, що саме вузол $i$ стане лідером.

Для вузлів із гіршими характеристиками значення $T_k$ буде більшим, і може бути виражене формулою:

$$T_k \approx T_{\max} \times (1 - \partial_{me}) + rand(a,b) \Rightarrow P_k \propto \frac{1}{T_k} \tag{4.19}$$

де $P_k$ — ймовірність вибору вузла з вищим тайм-аутом. Очевидно, що при однакових умовах $P_k < P_i$ , якщо $T_k > T_i$.

Щоб оцінити, наскільки ефективно алгоритм надає перевагу найкращому вузлу (тобто тому, хто має найменший $T_i$), можна скористатись використати ймовірность вибору найкращого вузла $P_{optimal}$:

$$P_{optimal} = \frac{1}{1 + \sum_{k \neq i} \frac{T_i}{T_k}} \tag{4.20}$$

Тут видно, що якщо всі $T_k > T_i$ , то всі частки $T_i / T_k < 1$, а отже $P_{optimal} \to 1$. Це означає, що алгоритм майже гарантовано обере найкращий вузол, якщо його характеристики значно кращі.

Також для загального оцінювання розподілу лідерства між вузлами використовується частота вибору вузла $i$:

$$f_{\text{leader}}(i) = \frac{1}{\sum_{k=1}^{n} \frac{T_k}{T_i}} \tag{4.21}$$

Це співвідношення дозволяє аналітично описати, як часто вузол *i* буде обраний лідером порівняно з іншими. Якщо $T_i << T_k$ для більшості $k \neq i$, то частота $f_{leader}(i) \to 1$, тобто саме цей вузол домінує у виборах.

Отже, описані математичні залежності формалізують поведінку алгоритму та доводять, що він забезпечує справедливий, чутливий до мережевих умов і пріоритетний вибір лідера.

Також доцільно формально підтвердити що метод запобігає ситуації коли всі вузли почнуть вибори одночасно, додатковоа випадкова компонента $rand(a,b)$ запобігає цій ситуації, коли декілька вузлів із подібними характеристиками отримують однакові тайм-аути. Це також усуває проблему "розщеплення мозку" (split-brain) у кластері, забезпечуючи унікальність тайм-ауту для кожного вузла.

Тайм-аути вузлів обчислюються таким чином, щоб уникнути одночасного завершення. Розподіл тайм-аутів у просторі часу забезпечує уникнення ситуацій, коли кілька вузлів одночасно ініціюють переобрання лідера. Це досягається завдяки випадковій складовій $rand(a,b)$ та корекції через затримки $T_{dlbc}, T_{bccc}, T_{dlcc}$

Ймовірність одночасного завершення тайм-аутів $P_{collision}$ є незначною, оскільки:

$$P_{collision} \propto \int_a^b f_{rand}(x)dx \tag{4.22}$$

де $f_{rand}(x)$ — функція розподілу випадкової складової. Залежно від вибраного діапазону $[a,b]$, ця ймовірність може бути зменшена до прийнятного рівня.

Таким чином, алгоритм вирішує проблему непередбачуваності вибору лідера завдяки використанню метрик стану мережі, які впливають на значення тайм-аутів. Вузли з кращими характеристиками отримують менші тайм-аути, що

збільшує їхні шанси стати лідером. Випадкова компонента додатково усуває конфлікти, забезпечуючи стабільність системи.

## 4.2 Розроблення програмно-апаратного комплексу на основі автономного адаптивного підходу для управління функціонуванням розподіленої системи

Щоб емпірично перевірити ефективність адаптивного підходу, було створено програмно-апаратний комплекс з автономним адаптивним підходом до управління та розроблено стратегію порівняльного тестування [129]. Як скінченний автомат (state machine), стан якого синхронізується за допомогою Raft, було реалізовано сховище ключ-значення (KV Store). Цей компонент надає HTTP API (порт 8000), що дозволяє клієнтам виконувати операції запису (HTTP POST) та читання (HTTP GET). Для спрощення клієнтської логіки, вузли-послідовники використовують HTTP перенаправлення (код 307) для перенаправлення клієнтських запитів до поточного лідера.Тестове середовище було розгорнуте за допомогою Docker та docker-compose.yml. Воно включало три вузли Raft, кожен у своєму контейнері (на базі Alpine Linux), що працюють в ізольованій віртуальній мережі (Docker bridge) зі статично призначеними IP-адресами. Це забезпечувало контрольованість та відтворюваність експериментів. Ключовим елементом стратегії була симуляція деградації мережі. За допомогою стандартної утиліти Linux tc (Traffic Control) та її модуля netem (Network Emulator), що виконувались всередині контейнерів, вносилася штучна затримка на мережеві інтерфейси двох з трьох вузлів (Вузол 1 та Вузол 2). Важливо, що затримка змінювалася динамічно за визначеним патерном протягом 120-секундного циклу: вона плавно зростала від невеликого базового значення до піку (приблизно +670 мс) і потім так само плавно спадала. Це імітувало типові сценарії тимчасового погіршення якості мережевих з'єднань. Генерація навантаження здійснювалася окремим скриптом (perf-test.sh), який послідовно надсилав 200 запитів на запис до API одного з вузлів (Вузла 2) за

допомогою утиліти curl. Протягом цього процесу ретельно вимірювалися та логувалися ключові показники якості обслуговування (QoS): відсоток успішно виконаних запитів (доступність), середня кількість запитів за секунду (пропускна здатність), а також статистичні характеристики затримки відповіді (середня, P99, максимальна, стандартне відхилення). Стратегія порівняння полягала у проведенні ідентичних тестів (однакове навантаження, однаковий патерн деградації мережі) для двох конфігурацій кластера: "базової" (де вузли використовували статичні тайм-аути) та "адаптивної" (де працював новий автономний адаптивний метод). Наприкінці кожного тесту проводилася верифікація даних для підтвердження коректності роботи консенсусу. Аріхтектура програмно-апаратного комплексу наведена на рисунках 4.20 та 4.21 відповідно.

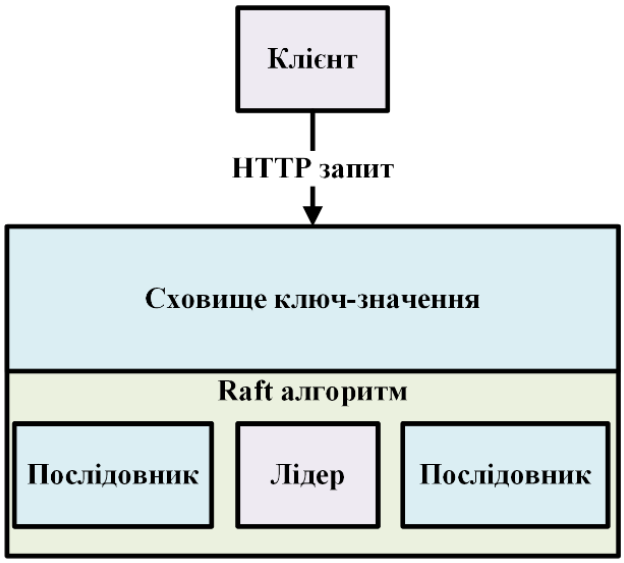



Рис. 4.20. Архітектура розподіленого сховища ключ-значення на базі алгоритму Raft

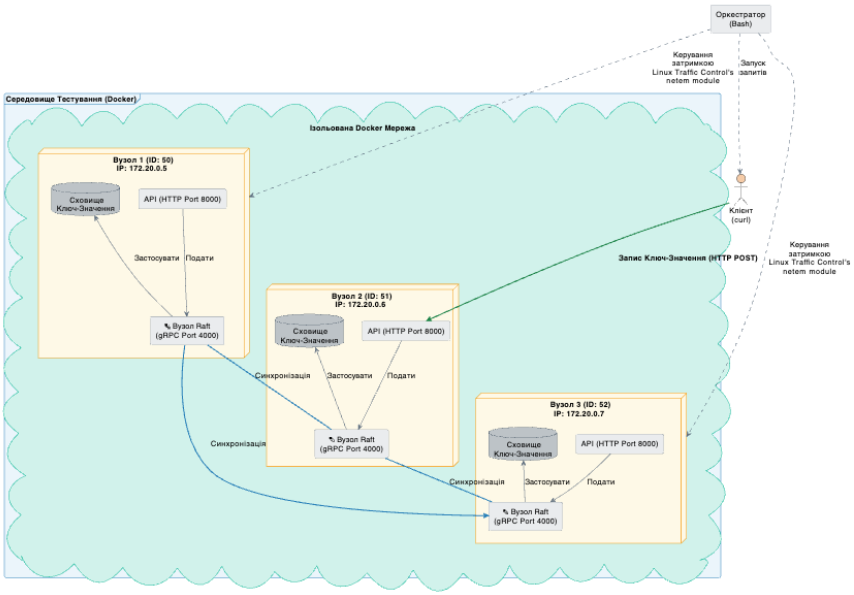
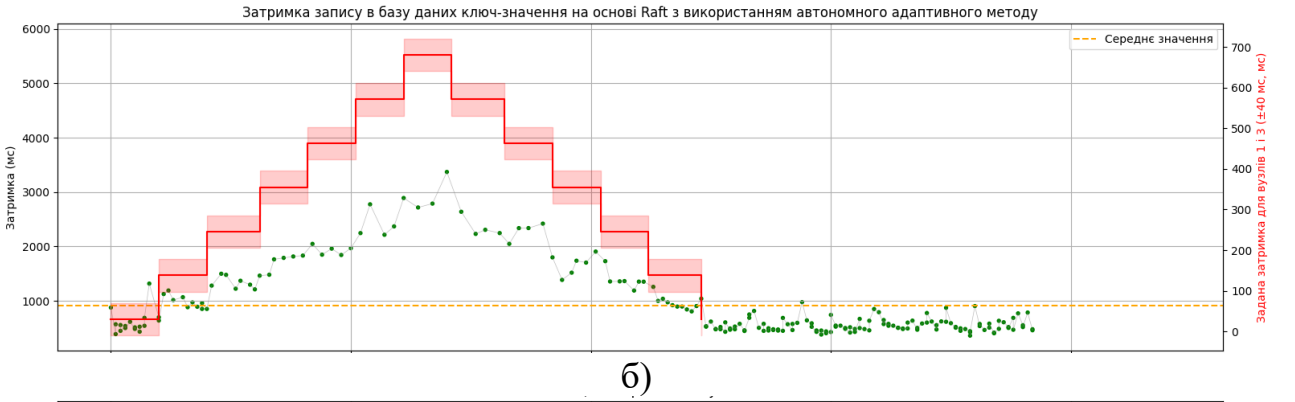



Рис. 4.21. Деталізована архітектура програмно-апаратного комплексу розподіленого сховища ключ-значення з реалізацією алгоритму Raft

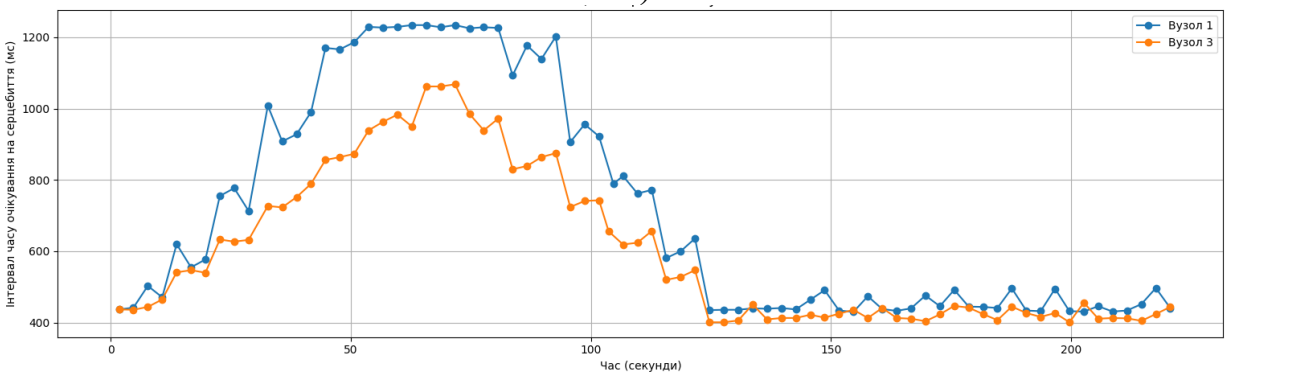
### 4.2.1. Дослідження ефективності використання запропонованих рішень та їх вплив на якість обслуговування

Порівняльний аналіз продуктивності та відмовостійкості базової та адаптивної реалізацій алгоритму Raft в умовах симульованої циклічної деградації мережі (змінна затримка до +~670 мс) виявив суттєві відмінності в їх поведінці, підтверджені як логами вимірювань (raft_benchmark_*.log), так і візуальним аналізом графіків. Ключовим результатом є значно вища доступність адаптивної системи. У той час як базова реалізація зі статичними тайм-аутами змогла успішно обробити лише 191 з 200 запитів (95.5% доступності), зазнавши 9 відмов (ідентифікованих як HTTP 307 помилки через втрату лідерства) під час пікових мережевих затримок, адаптивна версія продемонструвала 100% доступність, успішно завершивши всі 200 операцій запису за тих самих умов.


а)




б)




в)

Рис. 4.22. Вплив мережевих затримок на час запису в базу даних ключ–значення, що працює на основі протоколу Raft.

(а) Базова реалізація Raft: затримка запису за умов зовнішньо інжектованих змінних затримок на вузлах 1 і 3 (з варіацією ±40 мс);

(б) Модифікована реалізація Raft з автономним адаптивним методом: затримка запису за аналогічних мережевих умов;

(в) Еволюція інтервалів очікування (timeouts) на вузлах 1 і 3 у часі — як результат адаптації до змін мережевого середовища.

Графіки затримок (рис. 4.22 в) чітко візуалізують ці відмови у базовій версії (червоні точки/піки на верхньому графіку) та їх відсутність у адаптивній

(середній графік).Оцінка гіпотетичного часу, який знадобився б базовій версії для відновлення після кожної з 9 відмов та завершення запису (приблизно 3000 мс на відновлений запит під час піку), свідчить про додаткові ~27 секунд до загального часу виконання. Це знижує ефективну пропускну здатність базової версії при обробці всього набору даних до ~0.87 зап/сек, тоді як адаптивна версія показала фактичну пропускну здатність ~1.04 зап/сек.Аналіз затримки (latency) демонструє очікуваний компроміс адаптивного підходу. Хоча середня затримка для успішно завершених запитів була дещо нижчою у адаптивної версії (912 мс проти 954 мс у базової), її пікові показники були значно вищими (P99: 2892 мс проти 2043 мс; Max: 3371 мс проти 2066 мс). Це підтверджується графіками, де зелені точки (затримки запитів) для адаптивної версії сягають вищих значень під час пікової мережевої деградації. Адаптивна система свідомо збільшує час очікування для досягнення консенсусу, щоб уникнути перевиборів, що й дозволяє зберегти доступність.

Таблиця 4.2

Порівняння метрик якості обслуговування (QoS):

| Метрика | Звичайний | Адаптивний |
|---|---|---|
| Доступність | 95.50% | 100.00% |
| Пропускна здатність | 1.00 зап/сек | 1.04 зап/сек |
| Середня затримка | 954 мс | 912 мс |
| P99 Затримка | 2043 мс | 2892 мс |
| Максимальна затримка | 2066 мс | 3371 мс |
| Стандартне відхилення затримки | 1099.80 мс | 621.73 мс |
| Успішні/Невдалі записи | 191 / 9 | 200 / 0 |

Стабільність роботи адаптивної версії була значно вищою, що підтверджується суттєво нижчим стандартним відхиленням затримки (621.73 мс проти 1099.80 мс). Це вказує на більш передбачувану поведінку системи навіть

в умовах стресу. Графік еволюції внутрішніх інтервалів підтверджує роботу адаптивного механізму: внутрішні інтервали вузлів 1 та 3 динамічно зростали синхронно зі збільшенням симульованої мережевої затримки і спадали при її зменшенні, активно адаптуючись до умов мережі.

Застосування адаптивного методу призвело до значних покращень ключових показників якості обслуговування (QoS) порівняно з базовою реалізацією Raft. Найважливішим досягненням стало забезпечення 100% доступності адаптивною версією проти 95.5% у базовій, що означає повну обробку всіх 200 запитів без відмов навіть в умовах пікової мережевої деградації. Це безпосередньо вплинуло на ефективну пропускну здатність: адаптивна система продемонструвала ~1.04 зап/сек, тоді як базова, хоч і мала фактичну ~1.00 зап/сек по успішних запитах, її гіпотетична ефективна пропускна здатність для обробки всіх 200 запитів (з урахуванням часу на відновлення 9 відмов) оцінюється лише в ~0.87 зап/сек, що означає приблизно 19.5% перевагу адаптивного підходу в швидкості обробки всього навантаження. Щодо затримки, адаптивна версія показала дещо нижчу середню затримку (912 мс проти 954 мс), однак це супроводжувалося вищими піковими показниками P99 (2892 мс проти 2043 мс) та максимальної затримки (3371 мс проти 2066 мс), що є компромісом заради доступності. Проте, стабільність роботи значно покращилася: стандартне відхилення затримки для адаптивної версії склало 621.73 мс, що майже на 44% нижче, ніж у базової версії (1099.80 мс), вказуючи на значно вищу передбачуваність часу відповіді.

## 4.3 Висновки до четвертого розділу

1. Завдяки інтеграції автомномного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем у імітаційну модель кластеру вдалося досягти зменшення ймовірності передчасної реконфігурації системи у 3 рази у стандартному режимі функціонування. Водночас у періоди пікового навантаження впровадження даного підходу

забезпечує додаткове підвищення ймовірності реконфігурації на 5,6%, що свідчить про своєчасну реакцію системи на динамічну зміну структури з'єднань, підкреслює автономність та адаптивність методу, і як наслідок, можливість забезпечити узгодженість роботи кластеру незалежно від умов її функціонування.

2. Розроблено програмно-апаратний комплекс з автономним адаптивним підходом до управління для підвищення ефективності функціонування розподіленої системи, що дозволить на практиці підтвердити ефективність запропонованих методів та алгоритмів, залучаючи при цьому не лише програмну складову, а й комплекс реального мережевого обладнання. В результаті роботи розробленого програмно-апаратного комплексу підтверджується необхідність ефективного адаптивного підходу до коригування параметру затримки між вузлами кластеру, що впливає на вибір лідера в кластері, та зменшення ймовірності реконфігурації кластеру і, як, наслідок, підвищення якості надання послуг.

3. Завдяки впровадженню автономного адаптивного методу у роботу програмно-апартного комплексу вдалося зменшити стандартне відхилення затримки на 44%, що вказує на підвищену стабільність і передбачуваність роботи системи. Як наслідок, вдалося досягти зменшення ймовірності реконфігурації кластеру на 4.5%, що дозволить обробляти запити в моменти пікових навантажень без переобрання нових лідерів. Ефективна інтеграція та застосування методів виявлення відмов та адаптивного визначення порогу відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA дозволила підвищити швидкість обробки повідомлень системою на 19.5%, що свідчить про здатність адаптивної системи краще утилізовувати ресурси без втрати стабільності.

# ОСНОВНІ РЕЗУЛЬТАТИ ТА ВИСНОВКИ

В результаті виконання дисертаційних досліджень розв’язано науково-практичне завдання - розроблення нових методів та моделей підвищення ефективності функціонування розподілених сервісних систем з прогнозованим адаптивним виявленням відмов, які враховуватимуть поточні характеристики мережевого середовища, динаміку затримок і поведінку вузлів у кластері та визначати їх порогові значення.

Основні результати роботи полягають у наступному:

1. Проаналізовано основні методи та моделі, які забезпечують функціонування розподілених сервісних систем. Встановлено, що основою узгодженості функціонування розподілених систем є робота алгоритмів консенсусу, що забезпечують їх стабільну роботу та гарантують узгодженість даних між вузлами навіть у випадку збоїв. Основним критерієм підтримки стійкості розподіленої системи є своєчасне виявлення збоїв (fault detection), що забезпечується застосуванням як традиційних методів контролю стану (наприклад, перевірка часів відповіді та heartbeat-механізмів), так і сучасних статистичних та машинних підходів до виявлення відмов. Встановлено, що ефективним вирішенням цієї задачі є ймовірнісні підходи, які дозволяють оцінювати ступінь відмов залежно від поведінкових та мережевих метрик, що сприяє підвищенню точності реагування. Здатність системи до швидкої реконфігурації в умовах часткових відмов є критичною передумовою збереження її функціональної цілісності та надання безперервних сервісів кінцевим користувачам.

2. Для підвищення ефективності функціонування алгоритмів консенсусу в умовах нестабільності мережевих з’єднань у розподілених системах набув подальшого розвитку метод виявлення відмов, який, на відміну від традиційних підходів, враховує стан функціонування системи та мінімізує кількість хибнопозитивних рішень про відмови, спричинених короткочасними збоями в мережі. Особливістю даного методу є інтеграція ймовірнісного байєсівського

підходу для оцінювання стану вузлів на основі динамічного оновлення ймовірностей збоїв. Це дозволило враховувати такі мережеві аномалії, як затримки, втрата пакетів і короткі переривання зв’язку, та приймати більш обґрунтовані рішення щодо реальних відмов, що, своєю чергою, підвищує надійність алгоритмів консенсусу та забезпечує стабільність роботи розподілених сервісних систем.

Розвинений метод виявлення відмов з використанням ймовірнісного байєсівського підходу дав змогу зменшити кількість хибних рішень про відмову вузлів на 4.5%, залежно від обраного порогу реагування, зберігаючи при цьому чутливість системи до реальних збоїв, мінімізувати передчасні втручання у функціонування системи та забезпечити її надійність і стабільність, завдяки більш гнучкому, накопичувальному аналізу подій та адаптивному порогу, що враховує історію поведінки вузла і додаткові ознаки збою.

3. Удосконалено метод адаптивного визначення порогу відмов, який базується на прогнозуванні часу очікування повідомлень від потенційно несправного вузла. Особливістю даного методу є застосування моделі машинного навчання SARIMA для часової оцінки та прогнозування затримок у мережі на основі історичних даних, що збираються кожним вузлом у процесі роботи системи. Це дозволить враховувати реальні мережеві умови при визначенні часу очікування відповіді, динамічно адаптуючи поріг тайм-ауту для кожного з’єднання, що, своєю чергою, зменшує чутливість до короткочасних флуктуацій затримки та підвищує точність виявлення справжніх збоїв.

4. Проведено імітаційне моделювання роботи кластеру розподіленої сервісної системи із використанням методу адаптивного визначення порогу відмов та моделі машинного навчання SARIMA, яке дозволиво підвищити точність виявлення збоїв на 20% за рахунок прогнозування тривалостей часу відповіді, динамічної корекції порогу відмов і чіткого розмежування пікових моментів функціонування системи та відмов в обслуговуванні, забезпечуючи

при цьому адаптивну та ефективну реакцію системи на нестабільні умови функціонування мережі.

5. Запропоновано автономний адаптивний метод підвищення продуктивності РСС, який на основі знань про лідер-орієнтовану структуру алгоритму консенсусу, дозволяє динамічно коригувати параметри затримки між вузлами кластеру, обирати лідера в кластері, та забезпечити максимальну стабільність і продуктивність роботи системи вцілому. Завдяки інтеграції автомномного адаптивного методу підвищення ефективності функціонування розподілених сервісних систем у імітаційну модель кластеру вдалося досягти зменшення ймовірності передчасної реконфігурації системи у 3 рази у стандартному режимі функціонування. Водночас у періоди пікового навантаження впровадження даного підходу забезпечує додаткове підвищення ймовірності реконфігурації на 5,6%, що свідчить про своєчасну реакцію системи на динамічну зміну структури з'єднань, підкреслює автономність та адаптивність методу, і як наслідок, можливість забезпечити узгодженість роботи кластеру незалежно від умов її функціонування.

6. Розроблено програмно-апаратний комплекс на основі автономного адаптивного підходу для управління функціонуванням розподіленої системи. Завдяки впровадженню автономного адаптивного методу у роботу програмно-апартного комплексу вдалося зменшити стандартне відхилення затримки на 44%, що вказує на підвищену стабільність і передбачуваність роботи системи. Як наслідок, вдалося досягти зменшення ймовірності реконфігурації кластеру на 4.5%, що дозволить обробляти запити в моменти пікових навантажень без переобрання нових лідерів. Ефективна інтеграція та застосування методів виявлення відмов та адаптивного визначення порогу відмов шляхом прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA дозволила підвищити швидкість обробки повідомлень системою на 19.5%, що свідчить про здатність адаптивної системи краще утилізовувати ресурси без втрати стабільності.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ


1. M. Herlihy and J. Wing, "Linearizability: A correctness condition for concurrent objects," ACM Trans. Program. Lang. Syst., vol. 12, no. 3, pp. 463–492, Jul. 1990.
2. L. Lamport, "On interprocess communication," Distrib. Comput., vol. 1, pp. 77–101, 1986.
3. S. Chand and Y. A. Liu, "What's live? Understanding distributed consensus," arXiv preprint arXiv:2001.04787, 2020. [Online]. Available: https://arxiv.org/abs/2001.04787
4. P. Viotti and M. Vukolić, "Consistency in Non-Transactional Distributed Storage Systems," ACM Computing Surveys, vol. 49, no. 1, pp. 19:1–19:34, Jun. 2016. doi: 10.1145/2926965.
5. M. Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media Inc., Springfield, MO, USA, 2017, pp. 412–425.
6. S. Zhou, K. Li, L. Xiao, J. Cai, W. Liang, and A. Castiglione, "A systematic review of consensus mechanisms in blockchain," Mathematics, vol. 11, art. 2248, 2023. doi: 10.3390/math11102248
7. D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in Proc. 2014 USENIX Annu. Tech. Conf., USENIX, 2014, pp. 305–319.
8. C. Cachin, R. Guerraoui, and L. Rodrigues, Introduction to Reliable and Secure Distributed Programming, 2nd ed. Berlin, Germany: Springer, 2011. doi: 10.1007/978-3-642-15260-3
9. F. Muñoz Palacios, E. S. Espinoza Quesada, H. La, S. Salazar, S. Commuri, and L. R. Garcia Carrillo, "Adaptive consensus algorithms for real-time operation of multi-agent systems affected by switching network events," Int. J. Robust Nonlinear Control, Oct. 20, 2016.

10. Z. Hussein, M. A. Salama, and S. A. El-Rahman, "Evolution of blockchain consensus algorithms: A review on the latest milestones of blockchain consensus algorithms," Cybersecurity, vol. 6, p. 30, 2023. doi: 10.1186/s42400-023-00163-y

11. J. Liu, W. Feng, M. Huang, S. Feng, and Y. Zhang, “Grouped multilayer practical Byzantine fault tolerance algorithm: a consensus algorithm optimized for digital asset trading scenarios,” Sensors, vol. 23, no. 18, Art. 8903, 2023

12. R. Friedman, G. Kliot, and A. Kogan, "Hybrid Distributed Consensus," in Proceedings of the 17th International Conference on Principles of Distributed Systems (OPODIS 2013), Lecture Notes in Computer Science, vol. 8304, pp. 145–159, Springer, 2013. DOI: 10.1007/978-3-319-03850-6_11.

13. S. Liu, R. Zhang, C. Liu, et al., "An improved PBFT consensus algorithm based on grouping and credit grading," Sci. Rep., 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-28856-x

14. L. Chen, J. Liao, and N. Xiong, "Byzantine fault-tolerant consensus algorithms: A survey," Electronics, 2023. [Online]. Available: https://doi.org/10.3390/electronics12183801

15. L. Lamport, "Paxos made simple," ACM SIGACT News, vol. 32, no. 4, pp. 51–58, 2001.

16. D. Rystsov, "Dynamic plain Paxos," arXiv preprint arXiv:1506.04650, 2015.

17. H. Howard and R. Mortier, "Paxos vs Raft: Have we reached consensus on distributed consensus?" in Proc. 15th ACM SIGCOMM Conf., 2020, pp. 1–9. doi: 10.1145/3380787.3393681

18. W.-C. Shi and J.-P. Li, "Research on consistency of distributed systems based on Paxos algorithm," in Proc. 2012 Int. Conf. Wireless Mobile Technol. Ind. Pract., pp. 257–259, IEEE, 2012. doi: 10.1109/ICWAMTIP.2012.6413488

19. L. Lamport, "The part-time parliament," ACM Trans. Comput. Syst., vol. 16, no. 2, pp. 133–169, 1998, doi: 10.1145/279227.279229.

20. Y. Li, Y. Fan, L. Zhang, and J. Crowcroft, "RAFT consensus reliability in wireless networks: Probabilistic analysis," IEEE Internet Things J., vol. 10, no. 14, pp. 12839–12853, 2023. doi: 10.1109/JIOT.2023.3257402

21. W. Ren and R. W. Beard, “Consensus seeking in multiagent systems under dynamically changing interaction topologies,” IEEE Transactions on Automatic Control, vol. 50, no. 5, pp. 655–661, May 2005. doi: 10.1109/TAC.2005.846556.

22. E. Buchman, J. Kwon, and Z. Milosevic, “The latest gossip on BFT consensus,” arXiv preprint arXiv:1807.04938, 2018. [Online]. Available: https://arxiv.org/abs/1807.04938

23. P. Giura and W. Wang, "Using large scale distributed computing to unveil advanced persistent threats," Sci. J., vol. 1, pp. 93–105, 2012.

24. S. Zhuravel, O. Shpur, and Y. Pyrih, “Method of achieving consensus in distributed service systems,” Infocommunication Technologies and Electronic Engineering, vol. 2, no. 2, pp. 58–66, 2022. doi: 10.23939/ictee2022.02.058.

25. S. Li, H. Wu, Y. Li, and D. Guo, “Dynamic reconfiguration of distributed consensus protocols: A survey,” IEEE Access, vol. 8, pp. 168075–168089, 2020. doi: 10.1109/ACCESS.2020.3022858

26. A. Bessani, J. Sousa, and E. Alchieri, “State machine replication for the masses with BFT-SMART,” in Proc. 44th Annual IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN), Atlanta, GA, USA, 2014, pp. 355–362.

27. L. Lamport, M. Dahlia, and Z. Lidong, "Vertical Paxos and primary-backup replication," in Proc. ACM SIGACT-SIGOPS Symp. Principles Distrib. Comput., 2009. doi: 10.1145/1582716.1582783

28. Á. García-Pérez, A. Gotsman, Y. Meshman, and I. Sergey, "Pax-os consensus, deconstructed and abstracted," in Proc. 24th Int. Conf. Concurrency Theory (CONCUR), LNCS, vol. 9999, pp. 433–451, Springer, Heidelberg, 2018. doi: 10.1007/978-3-319-89884-1_32

29. M. Biely, Ž. Milosevic, N. Santos, and A. Schiper, “S-Paxos: Offloading the leader for high throughput state machine replication,” in Proc. 31st IEEE Symp. Reliable Distributed Systems (SRDS), Irvine, CA, USA, 2012, pp. 111–120.

30. S. Devineni, S. Kathiriya, and A. Shende, "Machine learning-powered anomaly detection: Enhancing data security and integrity," J. Artif. Intell. Cloud Comput., vol. 2, pp. 1–9, 2023. [Online]. Available: https://doi.org/10.47363/JAICC/2023(2)184

31. K. Venkatesan and S. B. Rahayu, "Blockchain security enhancement: An approach towards hybrid consensus algorithms and machine learning techniques," Sci. Rep., vol. 14, art. 1149, 2024. doi: 10.1038/s41598-024-51578-7

32. M. Križmančić and S. Bogdan, "Adaptive connectivity control in networked multi-agent systems: A distributed approach," PLOS ONE, vol. 19, no. 12, Dec. 2024.

33. Y. Tang, W. Zhang, and Y. Xia, “Leader–follower consensus of multi-agent systems under stochastic communication delays,” International Journal of Systems Science, vol. 44, no. 8, pp. 1496–1505, 2013.

34. N. Kryvinska and C. Strauss, "Conceptual model of business services availability vs. interoperability on collaborative IoT-enabled eBusiness platforms," in Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence, Berlin, Heidelberg: Springer, 2013, pp. 167–187.

35. T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," Journal of the ACM, vol. 43, no. 2, pp. 225–267, Mar. 1996, doi: 10.1145/226643.226647.

36. M. Balakrishnan, D. Malkhi, and T. Wobber, "Tango: Distributed data structures over a shared log," in Proc. 24th ACM Symp. Operating Syst. Principles (SOSP), 2013, pp. 325–340. doi: 10.1145/2517349.2522732

37. M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," J. Assoc. Comput. Mach. (JACM), vol. 32, no. 2, pp. 374–382, 1985. doi: 10.1145/3149.214121

38. H. Chen, H. Yu, S. Zhao, and Q. Shi, "Consensus-based distributed clustering for IoT," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), 2020.

39. J. C. Ji, C. T. Lam, and B. K. K. Ng, “A Survey on Consensus Algorithms for Distributed Wireless Networks,” in Proc. Int. Conf. Optoelectronic Information and Optical Engineering (OIOE), SPIE vol. 13513, 2024 (published 2025).

40. D. Gifford, Information Storage in a Decentralized Computer System, Xerox Palo Alto Research Center, Tech. Rep. CSL-81-8, 1981.

41. R. van Renesse and F. B. Schneider, "Chain replication for supporting high throughput and availability," in Proc. 6th USENIX Symp. Oper. Syst. Design Implement. (OSDI), 2004.

42. S. Zhuravel, M. Klymash, O. Shpur, and O. Lavriv, "Achieving consistency and consensus of distributed infocommunication systems," in Proc. 16th Int. Conf. Adv. Trends Radioelectron., Telecommun. Comput. Eng. (TCSET), Lviv, Ukraine, Feb. 2022, pp. 386–389.

43. H. Knudsen, J. S. Notland, P. H. Haro, T. B. Ræder, and J. Li, "Consensus in blockchain systems with low network throughput: A systematic mapping study," arXiv preprint, [Online]. Available: https://ar5iv.org/pdf/2103.02916

44. X. Li, H. Zhang, and H. Li, "Event-triggered leader-following consensus of multi-agent systems with communication delays," International Journal of Systems Science, vol. 47, no. 3, pp. 726–736, 2016.

45. B. Wang, S. Liu, H. Dong, X. Wang, W. Xu, J. Zhang, P. Zhong, and Y. Zhang, "Bandle: Asynchronous state machine replication made efficient," in Proc. 19th Eur. Conf. Comput. Syst. (EuroSys), 2024, pp. 265–280, doi: 10.1145/3627703.3650091

46. H. Howard, D. Malkhi, and A. Spiegelman, “Flexible Paxos: Quorum Intersection Revisited,” in Proc. 20th Int. Conf. on Principles of Distributed Systems (OPODIS), 2016, pp. 25:1–25:14.

47. H. Yu, H. Chen, S. Zhao, and Q. Shi, "Distributed soft clustering algorithm for IoT based on finite time average consensus," IEEE Internet Things J., vol. 8, pp. 16096–16107, 2021. doi: 10.1109/JIOT.2020.2981774

48. S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolić, “XFT: Practical Fault Tolerance Beyond Crashes,” in Proc. 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI), 2016, pp. 485–500.

49. M. Yin, D. Malkhi, M. K. Reiter, G. Golan-Gueta, and I. Abraham, “HotStuff: BFT Consensus with Linearity and Responsiveness,” in Proc. ACM Symp. on Principles of Distributed Computing (PODC), 2019, pp. 347–356.

50. J. Li, E. Michael, N. Sharma, A. Szekeres, and D. R. K. Ports, “Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering,” in Proc. 12th USENIX OSDI, 2016, pp. 467–483.

51. T. Crain, V. Gramoli, M. Larrea, and M. Raynal, “DBFT: Efficient Byzantine Consensus with Cryptography and Partial Synchrony,” in Proc. 17th IEEE Int. Symp. on Network Computing and Applications (NCA), 2018, pp. 1–8.

52. M. Santos, F. M. Quintao Pereira, and F. Pedone, "Dynamic reconfiguration in Byzantine fault-tolerant systems," ACM Trans. Comput. Syst. (TOCS), vol. 37, no. 3, pp. 1–37, 2019. doi: 10.1145/3361567

53. M. N. Cheraghlou, A. Khadem-Zadeh, and M. Haghparast, “A Survey of Fault Tolerance Architecture in Cloud Computing,” J. Netw. Comput. Appl., vol. 61, pp. 81–92, 2016.

54. F. Nawab and M. Sadoghi, "Consensus in data management: From distributed commit to blockchain," Found. Trends Databases, vol. 12, no. 4, pp. 221–364, 2023. doi: 10.1561/1900000075

55. M. Hasan and M. S. Goraya, “Fault tolerance in cloud computing environment: A systematic survey,” Computers in Industry, vol. 99, pp. 156–172, 2018.

56. P. Kumari and P. Kaur, “A survey of fault tolerance in cloud computing,” J. King Saud Univ. – Comput. Inf. Sci., vol. 33, no. 10, pp. 1159–1171, 2021.

57. S. Bakhshi Kiadehi, A. M. Rahmani, and A. S. Molahosseini, “Increasing fault tolerance of data plane on the Internet of Things using software-defined networks,” PeerJ Comput. Sci., vol. 7, article e562, 2021.

58. N. O. Pinciroli Vago, F. Forbicini, and P. Fraternali, “Predicting Machine Failures from Multivariate Time Series: An Industrial Case Study,” Machines, vol. 12, no. 6, art. 357, 2024.

59. N. Hayashibara, X. Défago, R. Yared, and T. Katayama, "The φ Accrual Failure Detector," in Proc. 23rd IEEE Int. Symp. Reliable Distributed Systems (SRDS), 2004, pp. 66–78.

60. D. Jauk, D. Yang, and M. Schulz, “Predicting Faults in High Performance Computing Systems: An In-Depth Survey of the State-of-the-Practice,” in Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC’19), 2019, pp. 1–12.

61. R. Maheshwari, P. Ram, and K. S. Trivedi, “Proactive Fault Tolerance in Cloud Ecosystems Using Failure Predictions,” in Proc. 10th IEEE Int. Conf. on Cloud Computing (CLOUD), 2017, pp. 88–95.

62. S. Xu, J. Li, X. Wang, and Z. Zhang, “FALL: Prior Failure Detection in Large-Scale Systems Based on Log Sequence Learning,” IEEE Trans. Dependable Secure Comput., vol. 19, no. 6, pp. 3578–3595, 2022.

63. O. Serradilla, E. Zugasti, J. Rodriguez, and U. Zurutuza, “Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects,” Appl. Intell., vol. 52, no. 9, pp. 10934–10964, 2022.

64. A. Das, I. Gupta, and A. Motivala, "SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol," in Proc. 2002 Int. Conf. Dependable Systems and Networks (DSN), 2002, pp. 303–312.

65. R. van Renesse, Y. Minsky, and M. Hayden, "A Gossip-Style Failure Detection Service," in Proc. IFIP Int. Conf. Distributed Systems Platforms and Open Distributed Processing (Middleware '98), 1998, pp. 55–70.

66. B. Chaurasia and A. Verma, “A Comprehensive Study on Failure Detectors of Distributed Systems,” J. Sci. Res. (BHU), vol. 64, no. 2, pp. 250–260, 2020.

67. J. Dunagan, N. J. A. Harvey, M. B. Jones, D. Kostić, M. Theimer, and A. Wolman, "FUSE: Lightweight guaranteed distributed failure notification," in Proc. 6th Symp. Operating Systems Design and Implementation (OSDI), San Francisco, CA, USA, Dec. 2004, pp. 245–260.

68. S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," IEEE Internet Things J., vol. 7, pp. 4641–4654, 2020. doi: 10.1109/JIOT.2020.2964162

69. H. Meling and L. Jehl, "Tutorial summary: Paxos explained from scratch," in Proc. Int. Conf. Principles of Distributed Systems (OPODIS), R. Baldoni, N. Nisse, and M. van Steen, Eds., Lecture Notes in Computer Science, vol. 8304, Springer, Cham, 2013. doi: 10.1007/978-3-319-03850-6_1

70. S. Benz and F. Pedone, "Elastic Paxos: A dynamic atomic multicast protocol," in Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS), 2017, pp. 2157–2164. doi: 10.1109/ICDCS.2017.84

71. H. Howard and J. Crowcroft, "Coracle: Evaluating consensus at the internet edge," ACM SIGCOMM Comput. Commun. Rev., vol. 45, no. 4, pp. 85–96, 2015. doi: 10.1145/2829988.2790010

72. J. Liu, Z. Wu, J. Lan, J. Yu, and D. Qian, “A Low-Overhead Cooperative Failure Detector,” in Proc. 5th Int. Conf. on Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2015, pp. 811–815.

73. J. Liu, Z. Wu, J. Dong, J. Wu, and D. Wen, “An energy-efficient failure detector for vehicular cloud computing,” PLoS ONE, vol. 13, no. 1, e0191577, 2018.

74. B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Managing Critical State: Distributed Consensus for Reliability, in Site Reliability Engineering: How Google Runs Production Systems, 1st ed., Sebastopol, CA, USA: O’Reilly Media, 2016.

75. G. Stein, L. A. Rodrigues, E. P. Duarte Jr., and L. Arantes, “Diamond-P-vCube: An Eventually Perfect Hierarchical Failure Detector for Asynchronous Distributed Systems,” J. Braz. Comput. Soc., vol. 23, no. 1, art. 15, 2017.

76. Y. Chen, B. Li, and X. Zhou, “LA-FD: A Low-Overhead Adaptive Accrual Failure Detector for Internet Applications,” in Proc. IEEE Int. Conf. on Smart Cloud (SmartCloud), 2017, pp. 50–57.

77. Haiwen Du, Dongjie Zhu, Yundong Sun, and Zhaoshuo Tian, "Leader Confirmation Replication for Millisecond Consensus in Private Chains," IEEE Internet of Things Journal, vol. 9, no. 11, pp. 7944–7958, June 2022, doi: 10.1109/JIOT.2021.3113835.

78. J. Misic, V. B. Misic, Z. Milosevic, and H. Xiong, “Performance evaluation of a scalable Byzantine fault-tolerant consensus protocol for permissioned blockchains,” Comput. Netw., vol. 178, p. 107344, 2020.

79. R. Nedelkoski, E. J. Hammer, S. Mühlbauer, M. Ettl, and C. Reich, “Systematic Evaluation of Deep Learning Methods for Log-Based Failure Prediction,” J. Syst. Softw., vol. 172, p. 110778, 2021.

80. A. R. Varshney and M. N. S. Swamy, “Fault-tolerant solutions for distributed data processing systems: a systematic survey,” Cluster Comput., vol. 23, no. 4, pp. 2915–2932, 2020.

81. K. Kanawaday and A. Sane, “Machine Learning for Predictive Maintenance: A Multiple Classifier Approach,” in Proc. IEEE Int. Conf. on Machine Learning and Applications (ICMLA), 2017, pp. 112–117.

82. S. Zhuravel, “Adaptive probabilistic methods for boosting the reliability of consensus algorithms in distributed systems,” in Proc. 2024 IEEE 17th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 2024, pp. 145–149. doi: 10.1109/TCSET64720.2024.10755585.

83. S. Zhuravel, O. Shpur, and M. Klymash, “Adaptive consensus algorithms: Designing for durability against unstable network connections,” International Journal of Computing, vol. 23, no. 4, pp. 574–582, 2024. doi: 10.47839/ijc.23.4.3756.

84. Zhuravel S., Klymash M., Shpur O., Mrak V. “Reducing the impact of unstable connections among nodes of wireless IIoT clusters using machine learning methods”, Lecture Notes in Electrical Engineering. Vol. 1198: Digital ecosystems: interconnecting advanced networks with AI applications. P. 144–159, 2024. doi: 10.1007/978-3-031-61221-3_8

85. S. Zhuravel, "Development of network simulation model for evaluating the efficiency of distributed consensus taking into account the instability of network connections," Inf. Commun. Technol. Electron. Eng., vol. 4, no. 1, pp. 10–19, 2024.

86. R. Castro and B. Liskov, “Practical Byzantine Fault Tolerance and Proactive Recovery,” ACM Transactions on Computer Systems, vol. 20, no. 4, pp. 398–461, Nov. 2002. doi: 10.1145/571637.571640.

87. Chandra, T., Griesemer, R., & Redstone, J. (2007). "Paxos Made Live: An Engineering Perspective." Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC), 398–407.

88. M. Rahmani and H. A. Talebi, “Robust consensus in multi-agent systems with unreliable communication links,” International Journal of Control, vol. 87, no. 5, pp. 1019–1029, 2014.

89. S. Gupta, J. Hellings, and M. Sadoghi, "Fault-Tolerant Distributed Transactions on Blockchain," Synthesis Lectures on Data Management, vol. 13, no. 3, pp. 1–169, 2021

90. M. Botezatu, J. Bogojeska, S. Tresp, E. Patareshcu, and G. Schintke, “Predicting Disk Replacement Towards Reliable Data Centers,” in Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), 2016, pp. 39–48.

91. N. Y. Lim, E. Lee, T. Zhu, T. Kelly, and S. Y. Ko, “Making Disk Failure Predictions SMARTer,” in Proc. 14th USENIX Conf. on File and Storage Technologies (FAST), 2016, pp. 157–168.

92. J. Kang, C. Koh, T. Kim, and S.-W. Han, “Temporal-Contextual Attention Network for Solid-State Drive Failure Prediction in Data Centers,” IEEE Access, vol. 12, pp. 36705–36715, 2024.

93. P. Viotti and M. Vukolić, "Consensus in Distributed Systems: A Comprehensive Survey," ACM Computing Surveys, vol. 51, no. 2, pp. 49:1–49:41, Mar. 2018.

94. G. Bracha and S. Toueg, “Asynchronous Consensus and Broadcast Protocols,” Journal of the ACM, vol. 32, no. 4, pp. 824–840, Oct. 1985, doi: 10.1145/4221.214134.

95. L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," arXiv preprint arXiv:2009.13012, 2020. [Online]. Available: https://arxiv.org/abs/2009.13012

96. A. Guru, H. Mohapatra, B. Mohanta, C. Altrjman, and A. Yadav, "A survey on consensus protocols and attacks on blockchain technology," Appl. Sci., vol. 13, p. 2604, 2023, doi: 10.3390/app13042604.

97. M. Nathan and W. James, Big Data: Principles and Best Practices of Scalable Real-Time Data Systems, New York, NY, USA, 2014, pp. 312–226.

98. R. Anand and D. Jeffrey, Mining of Massive Datasets, Stanford Univ., CA, USA, 2014, pp. 422–427.

99. B. Marr, "Big Data: Using SMART big data, analytics and metrics to make better decisions and improve performance," in Proc. QUIS5 Quality in Services Conf., Wiley, Karlstad, Sweden, 2015, pp. 222–226.

100. H. Srivatsa and M. Jagadeesh, Big Data Imperatives: Enterprise “Big Data” Warehouse, “BI” Implementations and Analytics, Apress, New York, NY, USA, Jun. 24, 2013, pp. 311–333.

101. A. Alaa, H. Hamza, and A. Kotb, "Performance evaluation of open source IoT platforms," [Online]. Available: https://www.researchgate.net/publication/330582133_Performance_Evaluation_of_Open_Source_IoT_Platforms

102. D. Toran, "IoT platforms comparisons," [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/114211/tfg-report-daniel-toran.pdf

103. F. P. Junqueira and B. Reed, ZooKeeper: Distributed Process Coordination, O'Reilly Media, 2013.

104. A. Raj, T. Truong-Huu, P. Mohan, and M. Gurusamy, "Crossfire attack detection using deep learning in software-defined ITS networks," in Proc. IEEE Int. Conf. Commun., Kuala Lumpur, Malaysia, 2019.

105. A. Saied, R. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," Neurocomputing, vol. 172, pp. 385–393, 2016.

106. X. Shu, J. Smiy, D. Yao, and H. Lin, "Massive distributed and parallel log analysis for organizational security," in Proc. 2013 IEEE Globecom Workshops (GC Wkshps), 2013, pp. 194–199.

107. R. Smith, A. Zincir-Heywood, M. Heywood, and J. Jacobs, "Initiating a moving target network defense with a real-time neuro-evolutionary detector," in Proc. ACM Symp. Appl. Comput. (SAC), New York, NY, USA, 2016, pp. 1095–1102.

108. C. Ten, G. Manimaran, and C. Liu, "Cybersecurity for critical infrastructures: Attack and defense modeling," IEEE Trans. Syst., Man, Cybern. A, Syst. Humans, vol. 40, no. 4, pp. 853–865, Jul. 2010.

109. S. Zhuravel, "Investigating the impact of unstable network connections on the cluster running a consensus algorithm," Inf. Process. Syst., vol. 2024, no. 1, pp. 29–38, 2024. [Online]. Available: https://doi.org/10.30748/soi.2024.176.04

110. Q. Hu, B. Tang, and D. Lin, "Anomalous user activity detection in enterprise multi-source logs," in Proc. IEEE Int. Conf. Big Data, New Orleans, LA, USA, 2017, pp. 797–804.

111. M. Klymash, N. Peleh, O. Shpur, and S. Hladun, "Monitoring of web service availability in distributed infocommunication systems," in Proc. Int. Conf. Adv. Trends Radioelectron., Telecommun. Comput. Eng. (TCSET), Lviv-Slavske, Ukraine, 2020, pp. 723–728.

112. M. Landauer, et al., "Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection," Comput. Secur., vol. 79, pp. 94–116, 2018.

113. H. Lin and P. Wang, "Implementation of an SDN-based security defense mechanism against DDoS attacks," in Proc. IEEE Secur. Def. Conf., Pennsylvania, 2016.

114. Y. Sang, H. Shen, Y. Tan, and N. Xiong, "Efficient protocols for privacy preserving matching against distributed datasets," in Information and Communications Security (ICICS 2006), P. Ning, S. Qing, and N. Li, Eds. Berlin, Heidelberg: Springer, 2006, vol. 4307, Lecture Notes in Computer Science, pp. 210–227, doi: 10.1007/11935308_15.

115. C. Li, M. Xu, J. Zhang, H. Guo, and X. Cheng, "SoK: Decentralized storage network," Cryptology ePrint Arch., Paper no. 2024/258, 2024. [Online]. Available: https://eprint.iacr.org/2024/258

116. N. El Rharbi, H. Atteriuas, A. Younes, A. Harchaoui, and O. Izem, "A comparative study of the recent blockchain consensus algorithms," in Proc. E-Learning and Smart Engineering Systems (ELSES 2023), Atlantis Press, 2024, pp. 316–327. doi: 10.2991/978-94-6463-360-3_32.

117. D. Gonçalves, J. Bota, and M. Correia, "Big data analytics for detecting host misbehavior in large logs," in Proc. IEEE Trustcom/BigDataSE/ISPA, 2015, pp. 238–245.

118. V. Vimal, K. U. Singh, A. Kumar, S. K. Gupta, M. Rashid, R. K. Saket, and S. Padmanaban, "Clustering isolated nodes to enhance network's lifetime of WSNs for IoT applications," IEEE Syst. J., vol. 15, pp. 5654–5663, 2021. doi: 10.1109/JSYST.2021.3103696

119. H. Beshley, M. Beshley, M. Medvetskyi, and J. Pyrih, "QoS-aware optimal radio resource allocation method for machine-type communications in 5G LTE and

beyond cellular networks," Wirel. Commun. Mob. Comput., vol. 2021, pp. 1–18, 2021. doi: 10.1155/2021/9966366

120. M. Klymash, H. Beshley, O. Panchenko, and M. Beshley, "Method for optimal use of 4G/5G heterogeneous network resources under M2M/IoT traffic growth conditions," in Proc. 2017 Int. Conf. Inf. Telecommun. Technol. Radio Electron. (UkrMiCo), IEEE, 2017.

121. L. Liu, Q. Yang, and Z. Zhu, "Industrial Internet data collection and processing technology research," Acad. J. Sci. Technol., vol. 5, pp. 269–272, 2023. [Online]. Available: https://doi.org/10.54097/ajst.v5i1.5663

122. M. Yu and X. Zhang, "An efficient way to parse logs automatically for multiline events," Comput. Syst. Sci. Eng., vol. 46, pp. 2975–2994, 2023. [Online]. Available: https://doi.org/10.32604/csse.2023.037505

123. N. Peleh, S. Zhuravel, O. Shpur, and O. Rybytska, "Structured and unstructured log analysis as a methods to detect DDoS attacks in SDN networks," Internet Things Eng. Appl., vol. 6, no. 1, 2021. doi: 10.23977/iotea.2021.060101

124. N. Bawany, J. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," Arab. J. Sci. Eng., vol. 42, pp. 425–441, 2017.

125. R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in Proc. IEEE GLOBECOM Workshops, Denver, CO, USA, 2010, pp. 408–415.

126. A. Dawoud, S. Shahristani, and C. Raun, "Deep learning and software-defined networks: Towards secure IoT architecture," Internet Things, vol. 3, pp. 82–89, 2018.

127. Y. Yang, W. Li, H. Gao, and Z. Hu, “Prediction and analysis of aircraft failure rate based on SARIMA model,” in Proc. 2nd IEEE Int. Conf. on Computational Intelligence and Applications (ICCIA), 2017, pp. 530–534.

128. S. Zhuravel, Network Instability Consensus Simulator (NICS): A Tool for Assessing Distributed Systems' Resilience [Software]. GitHub. [Online]. Available: https://github.com/ZLStas/simulation

129. S. Zhuravel, Software Complex with an Autonomous Adaptive Management Approach for Improving the Efficiency of a Distributed System [Software]. GitHub. [Online]. Available: https://github.com/ZLStas/raft-kotlin/tree/modification

130. Журавель С., Думич С., Шпур О. "Дослідження методів збору та обробки даних в розподілених інформаційних системах" Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. vol. 1, № 1, с. 20–38, 2021. doi: 10.23939/ictee2021.01.020

131. M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," Journal of the ACM, vol. 27, no. 2, pp. 228–234, 1980, doi: 10.1145/322186.322188.

132. D. Dolev and H. R. Strong, "Authenticated Algorithms for Byzantine Agreement," SIAM Journal on Computing, vol. 12, no. 4, pp. 656–666, 1983, doi: 10.1137/0212045.

133. M. Ben-Or, "Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols," in Proc. 2nd ACM Symposium on Principles of Distributed Computing (PODC), 1983, pp. 27–30, doi: 10.1145/800221.806707.

134. C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," Journal of the ACM, vol. 35, no. 2, pp. 288–323, 1988, doi: 10.1145/42282.42283.

135. B. M. Oki and B. H. Liskov, "Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems," in Proc. 7th ACM Symposium on Principles of Distributed Computing (PODC), 1988, pp. 8–17, doi: 10.1145/62546.62549.

136. J. Turek and D. Shasha, "The many faces of consensus in distributed systems," IEEE Computer, vol. 25, no. 6, pp. 8–17, 1992, doi: 10.1109/2.153253.

137. 7. E. Gafni and L. Lamport, “Disk Paxos,” Distributed Computing, vol. 16, pp. 1–20, 2003, doi: 10.1007/s00446-002-0070-8.

138. L. Lamport and M. Massa, “Cheap Paxos,” in Proc. Int. Conf. Dependable Systems and Networks (DSN), 2004, pp. 307–314, doi: 10.1109/DSN.2004.1311900.

139. L. Lamport, “Fast Paxos,” Distributed Computing, vol. 19, no. 2, pp. 79–103, 2006, doi: 10.1007/s00446-006-0005-x.

140. R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, “Zyzzyva: Speculative Byzantine Fault Tolerance,” in Proc. 21st ACM Symp. Operating Systems Principles (SOSP), 2007, pp. 45–58, doi: 10.1145/1294261.1294267.

141. F. Junqueira, B. Reed, and M. Serafini, “Zab: High-performance broadcast for primary-backup systems,” in Proc. IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN), 2011, pp. 245–256, doi: 10.1109/DSN.2011.5958223.

142. P. J. Marandi, M. Primi, and F. Pedone, “High throughput state-machine replication for parallel applications (S-Paxos),” in Proc. 2012 IEEE 31st Symposium on Reliable Distributed Systems (SRDS), 2012, pp. 362–373, doi: 10.1109/SRDS.2012.66.

143. I. Moraru, D. G. Andersen, and M. Kaminsky, “There is more consensus in egalitarian parliaments (EPaxos),” in Proc. 24th ACM Symposium on Operating Systems Principles (SOSP), 2013, pp. 358–372, doi: 10.1145/2517349.2517350.

144. A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, “The Honey Badger of BFT protocols,” in Proc. 2016 ACM SIGSAC Conf. Computer and Communications Security (CCS), 2016, pp. 31–42, doi: 10.1145/2976749.2978389.

145. A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, “On the security and performance of proof of work blockchains,” in Proc. 2016 ACM SIGSAC Conf. Computer and Communications Security (CCS), 2016, pp. 3–16, doi: 10.1145/2976749.2978341.

146. A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” in Proc. 37th Annual International

Cryptology Conference (CRYPTO), 2017, pp. 357–388, doi: 10.1007/978-3-319-63688-7_12.

147. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling Byzantine agreements for cryptocurrencies,” in Proc. 26th ACM Symposium on Operating Systems Principles (SOSP), 2017, pp. 51–68, doi: 10.1145/3132747.3132757.

148. M. J. Fischer, “The consensus problem in unreliable distributed systems (a brief survey),” in Proc. International Conference on Fundamentals of Computation Theory (FCT), 1983, pp. 127–140, doi: 10.1007/3-540-12689-9_99.

# ДОДАТОК А. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНИХ ДОСЛІДЖЕНЬ


"ЗАТВЕРДЖУЮ"
Директор
ТзОВ «Вітертек»
Назарук А.М.
"13" 05 2025 р.

**АКТ**

про використання результатів дисертаційної роботи
Журавля Станіслава Сергійовича на тему:
**"Методи та моделі підвищення ефективності функціонування розподілених сервісних систем"**

Даний акт складено про те, що у ТзОВ «Вітертек», яке спеціалізується на розробці, тестуванні програмного забезпечення, розгортанні інфраструктур та платформ в якості сервісів, з метою підвищення ефективності роботи власної серверної інфраструктури, були впроваджені результати дисертаційної роботи Журавля Станіслава Сергійовича "Методи та моделі підвищення ефективності функціонування розподілених сервісних систем", представленої на здобуття наукового ступеня доктора філософії, зокрема:

- Впроваджено модель машинного навчання SARIMA для прогнозування мережевих затримок і адаптивного визначення порогу виявлення збоїв, що дозволило системі своєчасно реагувати на пікові значення навантаження у мережі. Такий підхід суттєво знизив кількість хибнопозитивних спрацьовувань — з 5% до <0.5%, що забезпечило зменшення кількості помилкових рішень щодо відмови вузлів, зберігаючи чутливість до реальних збоїв.
- Використано автономний адаптивний метод підвищення ефективності функціонування розподілених сервісних систем, який дозволив досягти покращення часу доступності системи на 4.5%, замість втрати частини запитів під час пікових навантажень мережі. Як результат, система забезпечила стійку обробку запитів.

Внаслідок впровадження розроблених методів на серверній інфраструктурі ТзОВ «Вітертек» встановлено, що результати знаходяться в межах п'ятивідсоткового середньоквадратичного відхилення від поданих у дисертаційній роботі.


Директор
ТзОВ «Вітертек» Назарук А.М.

"ЗАТВЕРДЖУЮ"
Проректор з науково-педагогічної роботи
Національного університету
"Львівська політехніка"

_____ доц. Олег ДАВИДЧАК
"04" 03 2025 р.

**АКТ**

про впровадження в навчальний процес результатів
дисертаційної роботи на здобуття наукового ступеня доктора філософії
Журавля Станіслава Сергійовича

Цей акт складений комісією у складі завідувача кафедри ІКТ, д.т.н., проф. Михайла КЛИМАША, д.т.н., доц., доцент кафедри ІКТ Тараса МАКСИМЮКА, д.т.н., проф., професора кафедри ІКТ Мар'яна КИРИКА, д.т.н., доц., професора кафедри ІКТ Миколи БЕШЛЕЯ, про те, що результати дисертаційної роботи Журавля Станіслава Сергійовича на тему «Методи та моделі підвищення ефективності функціонування розподілених сервісних систем», представленої на здобуття наукового ступеня доктора філософії, використовуються у навчальному процесі кафедри «Інформаційно-комунікаційних технологій» Національного університету «Львівська політехніка».

Матеріали дисертаційного дослідження використовуються під час викладання дисципліни «Розподілені сервісні системи та Cloud-технології» для здобувачів другого (магістерського) рівня вищої освіти спеціальності 172 Електронні комунікації та радіотехніка. Зокрема, у навчальному процесі використовуються такі методи, запропоновані Станіславом Журавлем:

– модернізовано курс лекцій для здобувачів другого (магістерського) рівня вищої освіти спеціальності 172 Електронні комунікації та радіотехніка ОНП «Телекомунікації та радіотехніка» з дисципліни «Розподілені сервісні системи та Cloud-технології» - у частині теоретичних основ проектування розподілених сервісно-орієнтованих телекомунікаційних систем та методики побудови і забезпечення узгодженості функціонування мереж на основі Cloud-технологій;

– модернізовано лабораторні роботи з дисциплін «Розподілені інфомаційно-комунікаційні мережі» для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Інформаційно-комунікаційні системи», у яких використано запропоновані у роботі методи та моделі прогнозування часу очікування повідомлень від потенційно непрацездатного вузла за допомогою моделі машинного навчання SARIMA, що дозволяє зменшити вплив нестабільності мережевих з'єднань на функціонування алгоритму консистентності, і як наслідок, на працездатність розподіленої сервісної системи

Голова комісії:
завідувач кафедри ІКТ, д.т.н., проф. _____ Михайло КЛИМАШ

Члени комісії:

доцент кафедри ІКТ, д.т.н., доц. _____ Тарас МАКСИМЮК

професор кафедри ІКТ, д.т.н., проф. _____ Мар'ян КИРИК

професор кафедри ІКТ, д.т.н., доц. _____ Микола БЕШЛЕЙ

«ЗАТВЕРДЖУЮ»
Директор
ТзОВ ВТФ «Контех»
Смольницький Є.С.
04" березня 2025 р.

**АКТ**

про використання результатів дисертаційної роботи
Журавля Станіслава Сергійовича
**«Методи та моделі підвищення ефективності функціонування розподілених сервісних систем»**

Даний акт складений про те, що у ТзОВ ВТФ «Контех» для майбутнього розвитку телекомунікаційної мережі використані результати дисертаційної роботи Журавля Станіслава «Методи та моделі підвищення ефективності функціонування розподілених сервісних систем», представленої на здобуття наукового ступеня доктора філософії.

Зокрема, представники компанії ТзОВ ВТФ «Контех» за згодою автора Журавля С.С. використали імітаційну модель функціонування кластеру розподіленої сервісно-орієнтованої системи, яка, на відміну від відомих, враховує спрогнозоване значення тайм-ауту та визначає вузол, який має найсприятливіші умови для виконання ролі лідера, з урахуванням фактичних характеристик функціонування системи, та автоматизує автономний адаптивний метод, який на основі знань про лідер-орієнтовану структуру дозволяє динамічно коригувати параметри затримки між вузлами кластеру, обирати лідера в кластері, та забезпечити максимальну стабільність і продуктивність роботи системи вцілому.

Заступник директора Смольницький О.Є.

# ДОДАТОК Б. СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ ТА ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ

*Наукові праці, в яких опубліковані основні наукові результати дисертації:*


1. Zhuravel S., Klymash M., Shpur O., Mrak V. “Reducing the impact of unstable connections among nodes of wireless IIoT clusters using machine learning methods”, Lecture Notes in Electrical Engineering. Vol. 1198: Digital ecosystems: interconnecting advanced networks with AI applications. P. 144–159, 2024. doi: 10.1007/978-3-031-61221-3_8
2. S. Zhuravel, O. Shpur, and M. Klymash, “Adaptive consensus algorithms: Designing for durability against unstable network connections,” *International Journal of Computing*, vol. 23, no. 4, pp. 574–582, 2024. doi: 10.47839/ijc.23.4.3756.
3. N. Peleh, S. Zhuravel, O. Shpur, and O. Rybytska, “Structured and unstructured log analysis as a method to detect DDoS attacks in SDN networks,” *Internet of Things (IoT) and Engineering Applications*, vol. 6, no. 1, pp. 1–9, 2021. doi: 10.23977/iotea.2021.060101.
4. S. Zhuravel, “Development of network simulation model for evaluating the efficiency of distributed consensus taking into account the instability of network connections,” *Infocommunication Technologies and Electronic Engineering*, vol. 4, no. 1, pp. 10–19, 2024. doi: 10.23939/ictee2024.01.010.
5. S. Zhuravel, O. Shpur, and Y. Pyrih, “Method of achieving consensus in distributed service systems,” *Infocommunication Technologies and Electronic Engineering*, vol. 2, no. 2, pp. 58–66, 2022. doi: 10.23939/ictee2022.02.058.
6. Журавель С., Думич С., Шпур О. “Дослідження методів збору та обробки даних в розподілених інформаційних системах” *Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія*. vol. 1, № 1, с. 20–38, 2021. doi: 10.23939/ictee2021.01.020

*Наукові праці, які засвідчують апробацію матеріалів дисертації:*

7. S. Zhuravel, “Adaptive probabilistic methods for boosting the reliability of consensus algorithms in distributed systems,” in *Proc. 2024 IEEE 17th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2024, pp. 145–149. doi: 10.1109/TCSET64720.2024.10755585.

8. M. Klymash, O. Shpur, O. Lavriv, and S. Zhuravel, “Achieving consistency and consensus of distributed infocommunication systems,” in *Proc. 2022 IEEE 16th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2022, pp. 386–389. doi: 10.1109/TCSET55632.2022.9767019.

9. M. Klymash, S. Zhuravel, O. Shpur “Machine learning approaches to stabilize wireless IIoT clusters facing network instabilities” *Наукоємні технології в інфокомунікаціях: збірник наукових праць Міжнародної науково-практичної конференції HICT'2023*, Харків - Кам'янець-Подільський, Україна, 2023, с. 90–92.