Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Метод синтезу розподілених комп'ютерних систем на основі протоколів популяції
Назва теми

Рівень вищої освіти *другий (магістерський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 240231.24.02.09. ПЗ

Виконав здобувач II курсу, група КІ2м-24-2 — Підпис — Юлія ІЛЬЧИШИНА (Ініціали, прізвище)

Керівник — д. техн. наук, професор (Науковий ступінь, учене звання) — Підпис — Олег САВЕНКО (Ініціали, прізвище)

Нормоконтролер — д. техн. наук, професор (Науковий ступінь, учене звання) — Підпис — Сергій ЛИСЕНКО (Ініціали, прізвище)

До захисту допускаю:
завідувач кафедри КІІС — Підпис — Ольга ПАВЛОВА (Ініціали, прізвище)
«___» _травня_ 2026 р.
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ДРУГИЙ (МАГІСТЕРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КПС
_____ Ольга ПАВЛОВА

" 12 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ільчишиній Юлії Віталіївні
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод синтезу розподілених комп'ютерних систем на основі протоколів популяції

Керівник проекту (роботи) Савенко Олег Станіславович, д. тех. наук, професор
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 12.01.2026 р. № 6

2. Термін подання здобувачем роботи на кафедру 01.05.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз відомих методів синтезу розподілених комп'ютерних систем

Модель синтезу розподілених комп'ютерних систем на основі протоколів популяцій

Метод синтезу розподілених комп'ютерних систем на основі протоколів популяцій

Експериментальне дослідження методу синтезу розподілених комп'ютерних систем

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « 12 » 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	12.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	12.01.2026	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	20.01.2026	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.02.2026	виконано
5	Робота над науковою статтею	01.03.2026	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.03.2026	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	01.04.206	виконано
8	Оформлення пояснювальної записки згідно вимог	18.04.2026	виконано
9	Попередній захист ДРМ	29.04.2026	виконано
10	Захист ДРМ на засіданні ЕК	До 20.05.2026	

Здобувач _____ Юлія ІЛЬЧИШИНА
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Олег САВЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Метод синтезу розподілених комп’ютерних систем на основі протоколів популяції.

Автор роботи: Ільчишина Юлія Віталіївна

Керівник роботи: Савенко Олег Станіславович

Пояснювальна записка: 78 с., 3 рис., 12 табл., 3 дод., 81 джерел.

РОЗПОДІЛЕНІ СИСТЕМИ, ПРОТОКОЛИ ПОПУЛЯЦІЙ, ІНТЕРНЕТ РЕЧЕЙ, МЕРЕЖА, МЕТОД СИТЕЗУ, АНОНІСНІ АГЕНТИ, ПРОСТОРОВА СКЛАДНІСТЬ, ВІДМОВОСТІЙКІСТЬ

Об’єктом дослідження є процес функціонування та автоматизованого синтезу розподілених комп’ютерних систем із масовою взаємодією анонімних ресурсно-обмежених вузлів.

Предметом дослідження є математичні моделі, методи та алгоритми синтезу протоколів популяцій для забезпечення децентралізованого консенсусу та мінімізації просторової складності обчислювальних агентів.

Метою кваліфікаційної роботи магістра є підвищення ефективності проєктування ресурсно-обмежених розподілених комп’ютерних систем шляхом розроблення методу автоматизованого синтезу протоколів популяцій, які забезпечують мінімальну просторову складність та оптимальний час стабілізації обчислень.

Для розв’язання поставлених задач використовувалися методи теорії скінченних автоматів, математичний апарат мереж Петрі, методи математичного логічного виводу, елімінація кванторів Пресбургера, а також методи стохастичного моделювання та аналізу марківських ланцюгів за допомогою інваріантів і потенційних функцій.

Наукова новизна отриманих результатів:

− набув подальшого розвитку метод автоматизованого синтезу протоколів популяцій для обчислення предикатів Пресбургера, який базується на модульній декомпозиції глобальної логічної специфікації на атомарні підзадачі, що дозволило

забезпечити логарифмічну залежність кількості внутрішніх станів агентів від параметрів задачі та оптимізувати використання пам'яті у ресурсно-обмежених пристроях.

Практична значимість отриманих результатів полягає у розробленні програмного забезпечення для автоматизованого синтезу, симуляції та верифікації правил взаємодії агентів. Запропонований метод може бути використаний при проєктуванні децентралізованих мереж мобільних датчиків, систем Інтернету речей та самоорганізовних обчислювальних середовищ.

У першому розділі проведено аналіз сучасного стану проблеми синтезу розподілених комп'ютерних систем та обґрунтовано вибір моделі популяційних протоколів для ресурсно-обмежених середовищ. Визначено недоліки існуючих автоматних підходів, пов'язані з високою просторовою складністю при реалізації логічних виразів, та обґрунтовано доцільність розроблення нового методу модульного синтезу для оптимізації простору станів без втрати швидкості збіжності.

У другому розділі досліджено формальну математичну модель обчислень у популяційних системах та описано структуру простору станів агентів. Виконано аналіз існуючих базових конструкцій протоколів, їхніх часово-просторових компромісів та здатності переходити у тихий режим роботи. Сформовано математичний апарат верифікації коректності обчислень на основі системних інваріантів та монотонних потенційних функцій.

У третьому розділі розроблено структуру методу автоматизованого синтезу розподілених систем на основі протоколів популяцій. Реалізовано підхід до декомпозиції формул арифметики Пресбургера на порогові та періодичні підзадачі з використанням ієрархічних рівневих конструкцій та субпротоколів. Математично обґрунтовано критерій локальної робастності для забезпечення відмовостійкості обчислень в умовах динамічної зміни популяції.

У четвертому розділі виконано програмну реалізацію модуля синтезу та верифікації протоколів популяцій. Проведено експериментальні дослідження розробленого методу за допомогою моделювання процесів стабілізації на тестових

наборах предикатів. Отримані результати підтвердили асимптотичну оптимальність паралельного часу обчислень та ефективність логарифмічного стиснення простору станів, що дозволяє значно зменшити апаратні витрати мікросистемної техніки.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

РКЗ - Розподілена комп’ютерна система

CCS - Теорія комунікаційних систем

BFT - Візантійська відмовостійка система

val - Інваріант системи

pot - Потенційна функціяS

IoT – Інтернет речей

ВСТУП

Сучасний етап розвитку глобального інформаційного простору та мікросистемної техніки характеризується фундаментальним революційним переходом від жорстко детермінованих централізованих архітектур до надмасштабних розподілених середовищ.

Сьогодні можна помітити експоненціальне зростання кількості автономних обчислювальних одиниць у таких критично важливих галузях, як мобільні сенсорні мережі, наноробототехніка та синтетична біологія. Водночас цей тренд стає наскрізним для кіберфізичного виробництва, розумної енергетики, безпілотного транспорту та космічних систем зв'язку, де децентралізація обчислень є основою технологічної стійкості. У цих специфічних доменах виникає гостра потреба в координації колосальної кількості компонентів, кожен з яких володіє критично обмеженими ресурсами обробки даних та енергоспоживання.

Традиційні мережеві протоколи, що базуються на використанні унікальних ідентифікаторів, таких як IP-адреси, підтримці глобальних таблиць маршрутизації чи постійному зв'язку з центральним сервером, стають абсолютно непридатними. Це зумовлено не лише апаратними лімітами, а й динамічною природою середовищ, де інфраструктура може бути відсутня, а топологія мережі змінюється щосекунди, роблячи будь-яку спробу централізованого керування точкою критичного відмови.

Розподілена частина таких систем вимагає принципово нового рівня автономності, де кожен вузол діє не як підлеглий елемент ієрархії, а як рівноправний учасник глобального процесу. У масових мережах класична маршрутизація повідомлень замінюється механізмами дифузії інформації, де дані поширюються подібно до біологічних процесів або хімічних реакцій. У таких умовах обчислення стають «невидимими» для окремого агента, адже жоден вузол не володіє повним знанням про стан системи чи кількість учасників.

Кінцевий підсумок стає властивістю всієї мережі разом – це спільний наслідок тисяч дрібних взаємодій, який неможливо вгадати, якщо дивитися лише на якусь одну частину. Це допомагає створювати системи, що діють як «спільний

розум». Оскільки в них немає єдиного головного керівника, мережа не боїться ні навмисних нападів, ні випадкових поломок техніки. Завдяки такому принципу роботи системи, що функціонують за принципом «мережевого інтелекту».

Одним із найбільш перспективних і концептуально елегантних шляхів вирішення цієї проблеми є застосування моделі протоколів популяцій. Ця концепція пропонує радикально інший підхід до розподілених обчислень: замість складної координації згори, глобальна інтелектуальна поведінка системи формується через послідовність дуже простих локальних взаємодій між анонімними вузлами. У цій моделі агенти розглядаються як обмежені скінченні автомати, що обчислюють результат під час випадкових парних зустрічей агентів.

Такий стохастичний підхід дозволяє створювати системи, що володіють природною масштабованістю: додавання мільйона нових агентів не вимагає перепрограмування існуючих, а лише прискорює час досягнення консенсусу. Система продовжує виконувати своє завдання навіть за умови втрати значної частини вузлів, оскільки знання про стан обчислення розподілене по всій популяції, а ймовірнісний характер зустрічей гарантує, що інформація зрештою досягне кожного учасника з певним проміжком часу.

Проте шлях до практичного впровадження популяційних обчислень у реальні мікросистеми супроводжується серйозними технологічними викликами, пов'язаними із межею між теоретичною потужністю та фізичною реалізацією. Головна проблема більшості сучасних теоретичних рішень полягає у їхній високій просторовій складності, що стає вузьким місцем для розподілених систем. Часто для реалізації навіть базових логічних операцій, наприклад як, перевірки умови консенсусу, алгоритми вимагають такої кількості внутрішніх станів агента, що перевищує фізичні можливості пам’яті нано- чи мікропристроїв.

На сьогодні виникла чітка суперечність: з одного боку, існує потреба у створенні високоточних децентралізованих систем для складних розподілених обчислень у ворожих середовищах, а з іншого – наявні методи синтезу не дозволяють ефективно оптимізувати простір станів без втрати швидкості збіжності.

Саме тому доцільність даного дослідження полягає у необхідності розроблення нових методів синтезу так званих лаконічних протоколів, які б гарантували мінімальне використання пам'яті, зменшували логарифмічну залежність від параметрів та оптимальний час стабілізації для складних логічних предикатів у розподілених мережах. Лаконічні протокли також часто зустічаються у літературі як так звані стислі протоколи.

Актуальність роботи полягає у розробленні системного методу синтезу розподілених систем на основі лаконічних протоколів популяцій, що дають змогу мінімізувати апаратні вимоги до вузлів, кількісті станів, при збереженні повної функціональності та надійності в анонімних мережах.

Метою кваліфікаційної роботи магістра є підвищення ефективності проектування розподілених комп’ютерних систем шляхом розроблення методу синтезу протоколів популяцій, які забезпечують мінімальну просторову складність та гарантовану відмовостійкість обчислень.

Поставлена мета досягається розв’язанням таких основних завдань:

- проаналізувати сучасний стан теорії розподілених обчислень та існуючих моделей стохастичної взаємодії в анонімних мережах;
- розробити цільову функцію та алгоритм синтезу для побудови лаконічних протоколів, що здатні обчислювати складні логічні предикати Пресбургера;
- дослідити механізми забезпечення робастності та автономної відмовостійкості синтезованих систем до динамічних змін у чисельному складі популяції, зокрема до раптового видалення або появи нових обчислювальних агентів;
- здійснити дослідження ефективності розробленого методу синтезу на основі програмної симуляції та моделювання процесів досягнення стабільного консенсусу в розподілених середовищах.

Об’єктом дослідження є процес функціонування та синтезу розподілених комп’ютерних систем із масовою взаємодією анонімних вузлів.

Предметом дослідження є методи та алгоритми синтезу протоколів популяцій для забезпечення стабільного консенсусу та мінімізації апаратних витрат вузлів.

Наукова новизна отриманих результатів:

– розроблено новий метод синтезу протоколів популяцій, який забезпечує логарифмічну залежність кількості станів від параметрів задачі, що дає можливість суттєво оптимізувати використання пам’яті в ресурсно-обмежених системах;

– удосконалено критерії оцінки відмовостійкості розподілених систем через впровадження поняття локальної робастності для сценаріїв нестабільної кількості учасників.

На основі проведених досліджень розроблено метод автоматизованої генерації правил взаємодії агентів, що гарантує досягнення коректного результату обчислень за оптимальний паралельний час.

Практична значимість отриманих результатів полягає у розробленому програмному забезпеченні для симуляції та верифікації розподілених протоколів, яке дозволяє автоматизувати процес проектування децентралізованих мереж мобільних датчиків та наноробототехніки.

Для розв’язання поставлених задач використовувалися методи теорії автоматів, апарат мереж Петрі, методи математичного логічного виводу та стохастичного моделювання обчислювальних процесів.

За темою кваліфікаційної роботи опубліковано публікація з тезами у 17-й Міжнародній студентській науковотехнічній конференції «Перспективні мережеві та комп’ютерні технології» – ПЕРСИК 2026 (м. Харків, 23 квіт. 2026). Харків, 2026 та опубліковано наукову статтю «Generali- zed method for managing the lifecycle of Terraform infrastructure accross multiple environments» [81].

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ СИНТЕЗУ РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМ

1.1 Загальна характеристика розподілених комп'ютерних систем

Сучасний етап розвитку інформаційних технологій характеризується стрімким переходом від централізованих обчислювальних архітектур до складних розподілених систем. Розподілена комп’ютерна система, , визначається як сукупність автономних обчислювальних вузлів, які взаємодіють між собою через мережу передачі даних для досягнення спільної мети. Ключовою особливістю таких систем є те, що для користувача вони виглядають як єдиний цілісний ресурс, попри фізичну та логічну розрізненість їхніх компонентів.

Розподілені комп'ютерні системи є одним із ключових об'єктів сучасної інформатики та інженерії програмного забезпечення. Розподілені комп’ютерні системи виникли в результаті розвитку обчислювальної техніки, мереж і постійного зростання потреб у швидкій та ефективній обробці даних. Отже, це можна вважати логічним результатом становлення технологій. Поява розподілених систем стала відповіддю на обмеження попередніх підходів і прагнення створити системи, які працюють швидше, надійніше та можуть легко масштабуватися.

Концептуальні основи розподілених систем формувались протягом кількох десятиліть. На початкових етапах, а саме в 1950–1960-ті роки домінували централізовані обчислювальні системи [2, 19]. Усі ресурси зосереджувалися в одному центрі, а користувачі фактично не взаємодіяли з процесом виконання задач напряму [2, 19]. Обробка відбувалася у пакетному режимі, завдання запускалися послідовно, одне за одним. Такий підхід мав очевидні недоліки - низьку гнучкість, обмежену продуктивність і повну залежність від одного обчислювального вузла [12, 15].

Історія становлення сучасних обчислювальних архітектур бере свій початок у 1960–1970-х роках із виникненням концептуально нового підходу – систем розподілу часу. Цей період став переломним моментом у розвитку IT-індустрії, оскільки впровадження таких систем вперше дозволило кільком користувачам

взаємодіяти з одним потужним комп’ютером одночасно через віддалені термінали. Подібна інновація була спрямована на подолання неефективності тогочасних методів експлуатації електронно-обчислювальних машин, забезпечивши значно раціональніше використання дорогих апаратних ресурсів.

Проте, попри очевидний прогрес у зручності доступу, архітектура цих систем залишалася фундаментально централізованою. Усі обчислювальні операції зосереджувалися на одному пристрої, що не лише потребувало колосальних системних ресурсів для підтримки черг запитів, а й створювало критичну залежність від стабільності центрального вузла.

Як наслідок, обмеження централізованих систем та постійне зростання потреб у потужностях змусили наукову спільноту вже у 1970-х роках розпочати активні дослідження у сфері мережевих середовищ. Це дало поштовх до вивчення фундаментальних проблем, які раніше не мали такого критичного значення: синхронізації паралельних процесів, забезпечення узгодженості даних між різними вузлами та розробки алгоритмів відмовостійкості. Дослідники прагнули створити середовища, де обчислення могли б виконуватися автономними елементами, що взаємодіють між собою, мінімізуючи ризики, характерні для єдиної точки відмови. Саме цей перехід від локального розподілу часу до глобальної мережевої взаємодії заклав теоретичний фундамент для створення складних ієрархічних структур, які здатні масштабуватися відповідно до запитів користувачів.

Протягом 1970–1980-х років парадигма обчислювальних процесів зазнала докорінних змін, що було зумовлено стрімким прогресом у сфері розробки та впровадження комп’ютерних мереж [6, 21]. Фундаментальну роль у цьому процесі відіграло створення ARPANET0 [6] – інноваційної мережі, яка сьогодні одностайно визнається науковою спільнотою як прототип сучасного глобального Інтернету [21]. Проєкт було реалізовано за ініціативи Міністерства оборони США у тісній співпраці з провідними академічними та науковими установами [21]. Першочерговою метою було об’єднання військових і дослідницьких організацій у цілісну систему для оперативної передачі даних, проте, з огляду на геополітичну нестабільність періоду Холодної війни, стратегічним завданням стало

проектування інфраструктури, здатної зберігати працездатність навіть за умов ядерної атаки [21, 22].

Завдяки успішній реалізації цих мережевих протоколів з'явилася технічна можливість інтегрувати декілька територіально віддалених комп'ютерів у єдину спільну систему [3, 8]. У цей період відбулося формування та масове розповсюдження класичної архітектурної моделі «клієнт–сервер» [14, 18]. В межах даної концепції функціональні обов'язки розподілялися між кінцевими користувачами та потужними серверами, що допомогло значно зменшити навантаження на центральний комп'ютер. Проте, попри переваги у масштабованості, залежність від серверної частини все ще залишалася суттєвою вразливістю [15]. Початок 2000-х років ознаменувався остаточним формуванням сучасного наукового розуміння розподілених систем як сукупності автономних вузлів, які взаємодіють через мережу для спільного виконання обчислювальних задач [5, 20].

Визначальним етапом цієї епохи стало впровадження розподілених баз даних, розвиток хмарних обчислень та систем обробки великих даних [7, 9, 10]. Це дозволило подолати попередні бар'єри продуктивності, забезпечивши майже безмежну масштабованість систем і можливість оперувати гігантськими обсягами інформації [8].

Сучасний етап розвитку, що охоплює період від 2010-х років і дотепер, характеризується тотальною інтеграцією розподілених систем у всі ключові сфери людської діяльності: від хмарних сервісів та соціальних мереж до складних фінансових систем та штучного інтелекту [5, 23]. Сьогодні ми спостерігаємо, як ідеї децентралізації та віддаленої взаємодії реалізуються у хмарних обчисленнях, де тисячі серверів працюють як єдиний організм, та у великих розподілених базах даних, що обслуговують мільйони запитів щосекунди. Подібні принципи лежать в основі Інтернету речей та мобільних сенсорних мереж, де дрібні автономні пристрої постійно обмінюються інформацією без необхідності централізованого керування [22].

Крім того, блокчейн-мережі забезпечують безпрецедентний рівень безпеки та прозорості даних завдяки повній децентралізації. Таким чином, сучасні системи реального часу та глобальні цифрові платформи є прямим продовженням еволюційного шляху, що розпочався з перших спроб розділити машинний час між терміналами, перетворившись на складну інфраструктуру, без якої неможливо уявити функціонування сучасного суспільства.

Ключовими властивостями розподілених комп'ютерних систем є: паралелізм, тобто одночасне виконання задач різними вузлами; відсутність спільного глобального стану – кожен вузол має доступ лише до локального стану; асинхронність взаємодії – вузли не мають спільного годинника та можуть взаємодіяти у довільний момент часу; відмовостійкість – система повинна продовжувати функціонувати навіть у разі відмови окремих вузлів або каналів зв'язку.

З погляду архітектури розподілені системи можна класифікувати за декількома ознаками. За моделлю взаємодії розрізняють системи з повідомленнями та системи зі спільною пам'яттю. За ступенем однорідності вузлів – однорідні та неоднорідні системи. За характером синхронізації – синхронні, асинхронні та частково синхронні.

Для практичного застосування особливого значення набувають так звані слабкі моделі розподілених обчислень, в яких обчислювальні можливості окремих вузлів суттєво обмежені. Прикладами таких систем є пасивні RFID-мітки, хімічні реакційні мережі та мережі мобільних датчиків. У таких умовах традиційні підходи до розподілених обчислень стають непридатними, і виникає потреба у спеціалізованих моделях і методах синтезу.

Загальна задача синтезу розподіленої комп'ютерної системи полягає у визначенні структури, поведінки та протоколів взаємодії вузлів таким чином, щоб система в цілому реалізовувала задану функцію або вирішувала поставлену задачу. Синтез принципово відрізняється від аналізу: якщо аналіз починається з готової системи і досліджує її властивості, то синтез починається зі специфікації бажаних властивостей і будує систему, що їм задовольняє.

1.2 Класифікація архітектур розподілених комп'ютерних систем

Архітектура розподіленої комп'ютерної системи визначає фундаментальний спосіб організації її компонентів, характер їхньої взаємодії та стратегію розподілу функціональних обов'язків між ними. Слід зауважити, що різноманіття архітектурних підходів обумовлене широким спектром вимог – від пропускної здатності та мінімальної затримки до критичної потреби у відмовостійкості та масштабованості. Власне, вибір певної моделі диктується специфікою середовища, у якому функціонуватиме система, та ресурсами, доступними кожному окремому вузлу. Клієнт-серверна архітектура представлена на рисунку 1.1, розглядається як класичний ієрархічний підхід, де вузли чітко диференціюються на два типи: сервери, що надають послуги, та клієнти, що їх споживають.

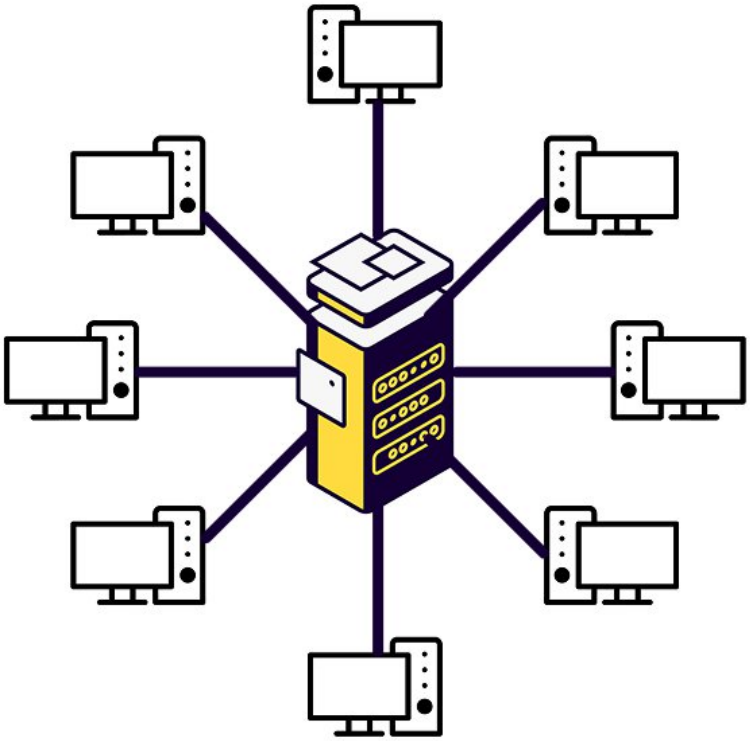


Рисунок 1.1 – Клієнт-серверна архітектура

У цій моделі сервери зазвичай виступають потужними сховищами ресурсів та виконують основні обчислення, тоді як клієнти ініціюють запити та

відображають результати. Перевагою такої структури є гранична простота управління та прозорий розподіл відповідальності між учасниками мережі.

Однак суттєвим недоліком залишається наявність єдиної точки відмови – сервера. Це означає, що вихід із ладу центрального вузла призводить до повної паралізації всієї системи, що робить її вразливою у динамічних або нестабільних умовах.

Альтернативою виступає децентралізована архітектура, де всі вузли мають рівні права та можуть одночасно виконувати функції як споживачів, так і постачальників ресурсів. Іншими словами, у такій системі відсутня жорстка ієрархія, що дозволяє мережі бути значно стійкішою до відмов окремих компонентів. Схематичну структуру децентралізованої архітектури представлено на рисунку 1.2. Фактично, вихід із ладу одного або декількох учасників не зупиняє загальний процес, оскільки їхні функції автоматично перебирають на себе інші вузли. Така модель демонструє високу масштабованість, проте вона вимагає складніших алгоритмів для пошуку ресурсів та координації дій без центрального нагляду.

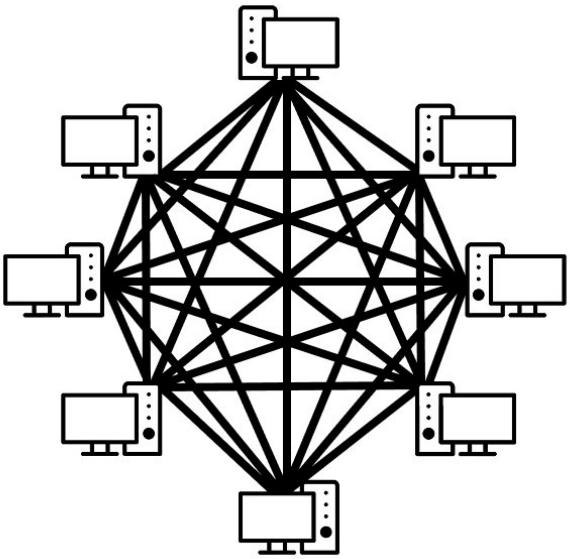


Рисунок 1.2 – Модель децентралізованої архітектури

Фактично, вихід із ладу одного або декількох учасників не зупиняє загальний процес, оскільки їхні функції автоматично перебирають на себе інші вузли. Така модель демонструє високу масштабованість, проте вона вимагає складніших алгоритмів для пошуку ресурсів та координації дій без центрального нагляду.

Хмарна архітектура базується на фундаментальній концепції делегування обчислювальних потужностей та систем зберігання даних як сервісу, що надається через глобальну мережу. Ключовою характеристикою таких систем є еластичність, під якою розуміють здатність інфраструктури автоматично підлаштовувати обсяг доступних ресурсів під поточні коливання навантаження. Фактично, це дозволяє уникати простою обладнання або, навпаки, дефіциту потужностей у моменти пікової активності. Рисунок 1.3 демонструє наочну модель хмарної архітектури.

Рисунок 1.3 – Зображення моделі хмарної архітектури

Залежно від моделі розгортання та прав доступу, хмарні середовища класифікують на публічні, приватні та гібридні. Останні набувають особливої популярності, оскільки дозволяють поєднувати високий рівень безпеки закритих корпоративних сегментів із безмежною масштабованістю зовнішніх хмарних платформ.

Сучасним стандартом проектування складних розподілених програмних продуктів є мікросервісна архітектура. В її основі лежить принцип декомпозиції монолітної системи на набір функціонально завершених, автономних сервісів, кожен з яких може бути розгорнутий незалежно від інших. Такий підхід передбачає, що кожен компонент відповідає за вузьку бізнес-задачу та взаємодіє з іншими частинами системи через чітко визначені програмні інтерфейси (API). Головна перевага цієї моделі полягає у можливості вибіркового масштабування найбільш навантажених вузлів, а також у високій ізоляції відмов. Як результат, збій в одному мікросервісі не призводить до краху всієї системи, що критично важливо для великих екосистем із мільйонами користувачів.

Реакторна архітектура, або архітектура, керована подіями, фокусується на асинхронній обробці сигналів, що генеруються різними компонентами або зовнішніми факторами. У такій структурі учасники взаємодії поділяються на виробників та споживачів подій, а зв'язок між ними забезпечується через спеціалізовані шини подій або черги повідомлень.

Таке рішення сприяє досягнути слабке зв'язування між елементами мережі, що робить систему надзвичайно гнучкою до змін та здатною ефективно обробляти нерівномірні потоки вхідних даних. Як наслідок, реакторні системи демонструють високу продуктивність у середовищах, де швидкість реакції на зміну контексту має вирішальне значення.

Окрему і найбільш релевантну для даного дослідження категорію складають архітектури для ресурсно-обмежених систем. Мова йде про мережі низькоспоживаючих сенсорів та великі масиви автономних агентів, обчислювальні можливості яких обмежені рамками скінченних автоматів. У таких умовах кожен вузол володіє мінімальним обсягом пам'яті та позбавлений можливості зберігати складні таблиці маршрутизації чи глобальні ідентифікатори. Власне, саме для таких специфічних середовищ протоколи популяцій виступають як найбільш природна модель обчислень. Вони допомагають перетворити хаотичні локальні взаємодії між обмеженими агентами на злагоджений механізм вирішення

глобальних логічних завдань, забезпечуючи високу надійність без необхідності у дорогому та енергоємному обладнанні.

Особливе місце в сучасній класифікації посідають анонімні розподілені системи, що базуються на моделі популяційних протоколів. На відміну від традиційних мереж, тут агенти не володіють унікальними ідентифікаторами і мають вкрай обмежену пам’ять.

Обчислення в таких архітектурах відбувається через випадкові або ж іншими словами стохастичні зустрічі, що нагадує поширення інформації у біологічних популяціях. Це означає, що глобальний результат є наслідком взаємодії всієї мережі через прості локальні правила взаємодії вся система приходить до спільного рішення. Власне, такий підхід є ідеальним для нанотехнологій та сенсорних мереж, де неможливо забезпечити стабільну топологію або великі обчислювальні потужності кожного вузла.

Комплексний аналіз наведених архітектур дозволяє стверджувати, що перехід до децентралізованих та анонімних моделей є логічним кроком у розвитку сучасних обчислювальних систем. Як наслідок, синтез протоколів, здатних ефективно працювати у таких середовищах, стає пріоритетним завданням для розробки надійного та відмовостійкого програмного забезпечення нового покоління.

1.3 Огляд існуючих підходів до синтезу розподілених комп'ютерних систем

Задача синтезу розподілених систем є однією з фундаментальних у теорії розподілених обчислень. Вона полягає у побудові коректної системи, що задовольняє заданій специфікації, з урахуванням обмежень середовища виконання. За роки досліджень у цій галузі сформувалось кілька ключових підходів.

Автоматний підхід базується на моделюванні поведінки системи за допомогою скінченних автоматів або мереж автоматів. Синтез зводиться до побудови автомата, що реалізує задану специфікацію. Цей підхід добре

розроблений теоретично та тісно пов'язаний з верифікацією моделей. Проте при збільшенні кількості компонентів виникає проблема комбінаторного вибуху простору станів.

Підхід на основі теорії процесів використовує формальні мови опису паралельних систем, такі як теорія комунікаційних або π-числення. Ці формалізми дозволяють точно специфікувати поведінку компонентів та їхню взаємодію. Синтез у рамках цього підходу часто зводиться до пошуку реалізації процесу, що відповідає заданій специфікації.

Методи на основі темпоральних логік дозволяють специфікувати бажані властивості системи у вигляді формул темпоральної логіки (LTL, CTL). Процедури синтезу будують систему, що задовольняє цим формулам. Перевагою є висока виразність специфікацій. Однак задача синтезу в загальному випадку є обчислювально складною, а для деяких логік – нерозв'язною.

Підхід на основі протоколів узгодження зосереджений на розробці алгоритмів досягнення консенсусу між вузлами системи. Класичними прикладами є протокол Паксос, алгоритм Рафт та Візантійська відмовостійка система протоколи. Ці методи орієнтовані на системи з обмеженою кількістю відмов та відносно потужними вузлами.

Підхід популяційних протоколів орієнтований на системи з великою кількістю ресурсно-обмежених агентів. Агенти мають скінченну кількість станів і взаємодіють попарно та стохастично. Цей підхід був введений Англуїном, Аспнесом, Діамаді, Фішером та Пералтою у 2004 році і став одним з найактивніших напрямів досліджень у розподілених обчисленнях.

Порівняно з іншими підходами, популяційні протоколи мають суттєві переваги для певного класу задач: вони природно моделюють системи без ідентифікаторів, з анонімними агентами та стохастичною взаємодією. Крім того, для них розроблено потужний математичний апарат аналізу та синтезу, що дозволяє точно характеризувати обчислювальну силу та складність протоколів.

1.4 Аналіз протоколів популяцій як моделі розподілених обчислень

Модель протоколів популяцій є спеціалізованою моделлю розподілених обчислень, розробленою для систем із великою кількістю простих агентів. Формально протокол популяцій визначається кортежем (Q, δ, I, O), де Q – скінченна множина станів, $\delta \subseteq Q^2 \times Q^2$ – множина переходів, $I \subseteq Q$ – множина вхідних станів, $O : Q \to \{0, 1\}$ – функція виведення.

Обчислення у моделі протоколів популяцій відбувається наступним чином. Система складається з n агентів, кожен із яких у будь-який момент часу перебуває в одному зі станів із Q. Конфігурацією системи називається мультимножина станів усіх агентів. Обчислення починається з початкової конфігурації, у якій усі агенти перебувають у вхідних станах із I, а їх кількість у кожному стані визначає вхідні дані.

На кожному кроці два агенти обираються випадковим чином і взаємодіють відповідно до функції переходів δ. Якщо перший агент перебуває у стані p, а другий – у стані q, і в δ існує перехід $(p, q) \to (p', q')$, то після взаємодії перший агент переходить у стан p', а другий – у стан q'. Загальна кількість агентів при цьому не змінюється.

Протокол вважається коректним, якщо з будь-якої вхідної конфігурації він досягає стабільного консенсусу. Стабільний консенсус – це конфігурація, у якій усі агенти мають однакове значення виходу (0 або 1), і жоден агент не може змінити своє значення. Протокол обчислює предикат φ, якщо для кожної вхідної конфігурації C він досягає консенсусу зі значенням $\varphi(C)$.

Ключовим теоретичним результатом у теорії протоколів популяцій є теорема про виразну силу: клас предикатів, які можна обчислити протоколами популяцій, точно збігається з класом предикатів Пресбургера. Предикати Пресбургера – це предикати, що виражаються в арифметиці першого порядку з додаванням. Використовують арифметику Пресбургера. Цей результат, отриманий Англуїном та співавторами, є основоположним для теорії протоколів популяцій.

Стохастична природа взаємодій у протоколах популяцій має принципове значення для аналізу їх поведінки. Замість детермінованого вибору пари агентів, взаємодія відбувається відповідно до пуассонівського процесу, де для кожної пари агентів є незалежний процес із однаковою інтенсивністю. Це дозволяє аналізувати часову складність протоколів у термінах паралельного часу – кількості взаємодій, поділеної на n.

З практичної точки зору протоколи популяцій знаходять застосування у різних галузях. У хімічних реакційних мережах молекули відіграють роль агентів, а хімічні реакції – роль переходів. У мережах мобільних датчиків окремі датчики є агентами, що взаємодіють при фізичному зближенні. У біологічних системах клітини або організми можуть взаємодіяти за схожою схемою.

1.5 Порівняльний аналіз моделей розподілених обчислень

Вибір відповідної моделі розподілених обчислень є критично важливим для ефективного розв'язання задач синтезу розподілених систем. Різні моделі мають різні характеристики виразної сили, обчислювальної складності та практичної застосовності.

Модель передачі повідомлень є одним із найпоширеніших підходів до розподілених обчислень. У цій моделі вузли взаємодіють, надсилаючи та отримуючи повідомлення. Вузли можуть мати унікальні ідентифікатори та зберігати довільний обсяг стану. Перевагою є висока гнучкість та виразна сила. Недоліком є необхідність складних протоколів координації та висока чутливість до відмов каналів зв'язку.

Модель спільної пам'яті передбачає, що вузли взаємодіють через спільну область пам'яті. Операції читання та запису можуть бути атомарними або виконуватись у критичних секціях. Ця модель добре вивчена теоретично, проте реалізація справжньої спільної пам'яті у розподілених системах вимагає складних механізмів когерентності кешу та синхронізації. Модель розподілених автоматів розглядає систему як мережу скінченних автоматів, що синхронно або асинхронно

виконують переходи. Ця модель добре підходить для верифікації та має компактне математичне представлення. Основним обмеженням є складність аналізу при великій кількості компонентів.

Порівняння основних моделей у контексті розподілених систем з великою кількістю ресурсно-обмежених агентів свідчить про переваги протоколів популяцій. Зокрема, на відміну від класичних моделей передачі повідомлень, протоколи популяцій не вимагають ідентифікаторів агентів, що природно відповідає ситуаціям, коли агенти є анонімними або взаємозамінними. Крім того, стохастична модель взаємодії добре відповідає фізичним системам, де контакти між агентами відбуваються випадково.

Суттєвою перевагою протоколів популяцій перед іншими моделями є наявність точних характеристик виразної сили та складності. Для класичних розподілених моделей такі точні характеристики часто або невідомі, або є нерозв'язними задачами. Натомість, як буде показано у наступних розділах, для протоколів популяцій відомі точні нижні та верхні межі просторової складності для широкого класу предикатів.

Важливим аспектом порівняння є також відмовостійкість. Класичні розподілені протоколи розроблялись для систем з відносно невеликою кількістю відмов. Протоколи популяцій, навпаки, природно орієнтовані на масштабні системи, де окремі відмови є нормою, а не виключенням. Дослідження відмовостійкості протоколів популяцій є одним із активних напрямів сучасних досліджень.

1.6 Постановка задачі дослідження

На основі проведеного аналізу існуючих підходів до синтезу розподілених комп'ютерних систем та аналізу можливостей і обмежень моделі протоколів популяцій сформулюємо задачу дослідження даної магістерської роботи.

Об'єктом дослідження є процес синтезу розподілених комп'ютерних систем, що функціонують в умовах обмежених обчислювальних ресурсів та стохастичної взаємодії між компонентами.

Предметом дослідження є методи та алгоритми синтезу протоколів популяцій для реалізації заданих обчислювальних функцій у розподілених комп'ютерних системах, а також характеристики просторової та часової складності таких протоколів.

Метою роботи є розробка методу синтезу розподілених комп'ютерних систем на основі протоколів популяцій, що забезпечує ефективне використання обчислювальних ресурсів при збереженні коректності та відмовостійкості системи.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– провести аналіз існуючих методів та підходів до синтезу протоколів популяцій для задач різного типу;

– розробити формальну модель задачі синтезу розподіленої системи на основі протоколів популяцій;

– розробити метод синтезу протоколів популяцій для предикатів порогового типу та предикатів Пресбургера;

– дослідити просторову та часову складність синтезованих протоколів;

– розробити та реалізувати програмне забезпечення для синтезу та симуляції протоколів популяцій;

– провести експериментальні дослідження розробленого методу та порівняти результати з відомими підходами.

1.7 Висновки до першого розділу

У першому розділі магістерської роботи проведено аналіз стану проблеми синтезу розподілених комп'ютерних систем та обґрунтовано вибір протоколів популяцій як основного інструменту для вирішення поставленої задачі.

Встановлено, що розподілені комп'ютерні системи є невід'ємною складовою сучасної інформаційної інфраструктури та характеризуються паралелізмом,

відсутністю спільного глобального стану, асинхронністю взаємодії та необхідністю відмовостійкості. Розглянуто основні архітектурні підходи до побудови таких систем, включаючи клієнт-серверну, однорангову, хмарну та мікросервісну архітектури.

Проаналізовано існуючі методи синтезу розподілених систем: автоматний підхід, методи на основі теорії процесів, темпоральних логік та протоколів узгодження. Показано, що для систем з великою кількістю ресурсно-обмежених агентів найбільш природним та ефективним є підхід на основі протоколів популяцій.

Детально розглянуто модель протоколів популяцій: формальне визначення, механізм обчислення, поняття стабільного консенсусу та виразну силу. Встановлено, що клас предикатів, обчислюваних протоколами популяцій, точно збігається з класом предикатів Пресбургера, що є фундаментальним теоретичним результатом.

Проведено порівняльний аналіз моделей розподілених обчислень та показано переваги протоколів популяцій для певного класу задач: відсутність вимог до ідентифікаторів агентів, природна відповідність стохастичним фізичним системам та наявність точних теоретичних характеристик складності.

На основі проведеного аналізу сформульовано задачу дослідження: розробити метод синтезу розподілених комп'ютерних систем на основі протоколів популяцій, що забезпечує мінімальну просторову складність при збереженні коректності та відмовостійкості, та реалізувати його у вигляді програмного забезпечення з подальшою експериментальною перевіркою

2 МОДЕЛЬ СИНТЕЗУ РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМ НА ОСНОВІ ПРОТОКОЛІВ ПОПУЛЯЦІЙ

2.1 Формальна модель протоколу популяцій

Формальна модель протоколу популяцій виступає фундаментальним математичним апаратом для проектування та аналізу поведінки розподілених комп'ютерних систем, що базуються на принципах самоорганізації. Модель протоколу популяцій є математичною основою для синтезу розподілених комп'ютерних систем. Вона допомагає точно специфікувати поведінку системи, доводити коректність протоколів та аналізувати їхню складність. У цьому підрозділі наведено повне формальне визначення моделі та базові поняття, необхідні для подальшого дослідження.

У межах даного дослідження протокол популяцій визначається як впорядкований кортеж P, де кожен елемент відповідає за конкретний аспект життєвого циклу обчислення.

$$P = (q, \delta, i, o), \tag{1.1}$$

де q – скінченна множина станів,

$\delta \subseteq q^2 \times q^2$ – множина переходів взаємодії,

$i \subseteq q$ – множина вхідних станів,

$o: q \to \{0, 1\}$ – функція виведення результатів.

Тобто, протокол популяцій визначається як кортеж $P = (q, \delta, i, o)$. Множина q представляє скінченний та незмінний набір внутрішніх станів агентів, що виконує роль пам'яті вузла; оскільки розмір q є константним, це гарантує, що складність окремого агента не зростає при збільшенні загальної кількості учасників мережі.

Множина переходів $\delta \subseteq q^2 \times q^2$ описує правила локальної взаємодії між парою агентів, де вхідна пара станів (p, q) трансформується у вихідну пару (p', q'), що дозволяє системі змінювати свій глобальний стан через серію випадкових контактів.

Множина вхідних станів $i \subseteq q$ визначає початкову ініціалізацію системи, де кожен агент переходить у певний стан залежно від отриманого зовнішнього сигналу або вхідного значення предиката.

Функція виведення $o{:}q \to \{0, 1\}$ забезпечує інтерпретацію поточного обчислювального стану агента, ставлячи йому у відповідність конкретне булеве значення, що дозволяє зчитувати результат роботи протоколу в будь-який момент часу.

Кожна елементарна операція в межах моделі популяційних протоколів реалізується через перехід із множини δ, який має суворий формальний вигляд $(p, q) \to (p', q')$. У цій структурі впорядкована пара (p, q) репрезентує стани двох агентів безпосередньо до моменту їхнього фізичного або логічного контакту, тоді як пара (p', q') визначає їхні модифіковані стани після завершення акту взаємодії. Фундаментальною особливістю таких переходів є їхня асиметричність, яка базується на розрізненні ролей учасників зустрічі: один агент завжди виступає в ролі ініціатора, перший операнд p, а інший – у ролі відповідача, другий операнд q. Такий підхід дає змогу системі диференціювати напрямок інформаційного потоку навіть у ситуаціях, коли обидва вузли перебувають в ідентичних початкових станах.

Математично це означає, що правила $(p, q) \to (p', q')$ та $(q, p) \to (q', p')$ розглядаються як дві абсолютно різні операції, якщо не виконується специфічна умова симетрії $\{p, q\} = \{p', q'\}$. Наявність подібної асиметрії надає розробнику протоколу гнучкий інструментарій для проектування складних алгоритмів розподіленого керування, зокрема для задач обрання лідера, генерації унікальних токенів або впорядкування агентів у безіменній популяції.

Окрім того, детермінована логіка кожного окремого переходу на локальному рівні, у поєднанні з імовірнісним характером зустрічей на глобальному рівні, формує механізм еволюції системи від початкового хаотичного розподілу до впорядкованого стабільного консенсусу. Важливо також враховувати, що функція переходу повинна бути скінченною та повністю визначеною для будь-якої комбінації станів із множини q, що гарантує відсутність «мертвих зон» або

тупикових конфігурацій, у яких обчислення могло б зупинитися до моменту досягнення правильного результату. Таким чином, саме через детальну специфікацію цих парних взаємодій досягається ефект виникнення складної інтелектуальної поведінки всієї системи при використанні вкрай простих локальних правил, коли система функціонує як єдиний мережевний інтелект.

Динаміка функціонування популяційного протоколу визначається через можливість активації конкретних правил взаємодії в межах поточної конфігурації системи. Перехід $t = (p, q \to p', q')$ вважається застосовним та дійсним у конфігурації C лише за умови фізичної наявності агентів у відповідних станах: кількість агентів у стані p та q повинна бути не меншою за одиницю, а у специфічному випадку взаємодії агентів в однаковому стані ($p = q$) конфігурація має містити щонайменше два таких агенти.

Результатом реалізації переходу t є трансформація системи в нову конфігурацію C', яка розраховується шляхом покомпонентної арифметичної зміни мультимножини: віднімання вихідних станів p, q та додавання результуючих станів p', q'. Такий крок позначається як $C \to_t C'$, а можливість переходу між віддаленими конфігураціями через довільну послідовність кроків фіксується відношенням досяжності $C \to^* C'$, що є критично важливим для аналізу траєкторії обчислень.

Процес функціонування системи в часі описується поняттям виконання, що являє собою впорядковану послідовність конфігурацій $\sigma = C_0, C_1, C_2, \ldots$. Кожен елемент цієї послідовності випливає з попереднього через акт взаємодії або залишається незмінним, що моделює дискретний час у розподіленому середовищі.

Особливе місце в аналізі займає прогін – це формалізація нескінченного виконання, яка враховує властивість справедливості планувальника. У прогоні множина конфігурацій, що зустрічаються нескінченно часто, повинна бути замкненою відносно досяжності. Це означає, що система не може назавжди ігнорувати певний перехід, якщо він постійно залишається доступним, що гарантує відсутність штучних зациклень у проміжних станах.

Кінцевою метою будь-якого популяційного протоколу є формування єдиної відповіді всією групою агентів, що описується через поняття b-консенсусу.

Конфігурація C визнається консенсусною, якщо для кожного стану q, який має хоча б одного представника в популяції, тобто належить до носія мультимножини $q \in \text{supp}(C)$, функція виходу O(q) повертає однакове булеве значення b. Проте для коректності обчислень недостатньо простого збігу думок; система повинна досягти стабільного консенсусу. Це такий стан, при якому не лише поточний вихід усіх агентів однаковий, а й будь-яка подальша досяжна конфігурація гарантовано зберігатиме цей же результат. Таким чином, стабільність забезпечує незворотність прийнятого рішення в умовах постійних випадкових взаємодій.

Загальна обчислювальна спроможність протоколу P визначається його здатністю обчислювати предикат φ. Предикат вважається обчислюваним, якщо для будь-якої валідної початкової конфігурації C_0, сформованої на основі вхідних станів I, кожен можливий справедливий прогін системи з часом стабілізується до значення $\varphi(C_0)$. Це визначення підкреслює стохастичну природу моделі: незалежно від того, які саме пари агентів зустрічатимуться і в якому порядку, архітектура правил переходів повинна бути спроектована таким чином, щоб спрямовувати популяцію до єдино правильного логічного висновку. Саме на основі цього формалізму здійснюється подальший синтез правил переходів, що забезпечують коректну обробку вхідних даних.

2.2 Структура агентів та простір станів у розподіленій системі

Структура агентів у протоколі популяцій є максимально спрощеною, що дозволяє системі бути надзвичайно стійкою та масштабованою. Кожен агент у такій мережі діє як крихітний обчислювальний вузол, чия поведінка повністю визначається множиною станів Q та функцією виведення O. Ключовим принципом тут є повна анонімність: агент не має ідентифікатора , як наприклад, IP-адреса, не володіє інформацією про свою позицію в просторі та не здатний розрізнити своїх «співрозмовників», якщо вони перебувають у однакових станах. Це означає, що стан агента – це не просто мітка, а єдиний доступний йому обсяг пам’яті, який містить усю історію його попередніх взаємодій у стислому вигляді.

З погляду синтезу розподіленої системи, простір станів Q диференціюється на кілька функціональних сегментів, де підмножина I кодує початкову конфігурацію вхідних параметрів, фактично перетворюючи агента на сенсор, що фіксує первинні дані. Вхідні стани $I \subseteq Q$ кодують початкові дані: кількість агентів у кожному вхідному стані визначає значення відповідного вхідного параметра предиката. Проміжні стани $Q \setminus (I \cup O^{(-1)}(0) \cup O^{(-1)}(1))$ які не належать до вхідного або вихідного спектра, слугують для транзитякного накопичення та обробки даних, що дозволяє агентам виконувати роль «носіїв» незавершених обчислень, які поступово трансформуються під час стохастичних контактів. Вихідні стани поділяються на стани з виходом 1 (стани прийняття) та стани з виходом 0 (стани відхилення). Вихідні стани, своєю чергою, визначають поточне логічне значення, яке транслює агент, тобто вони є індикаторами того, до якого висновку прийшов конкретний вузол на даному етапі прогону.

Ключовим та критичним параметром протоколу є потужність $|Q|$ множини станів, що безпосередньо відповідає просторовій складності системи. Отже, цей показник визначає «об'єм пам'яті» кожного окремого агента, вказуючи на те, наскільки складну інформацію він здатний зберігати в межах одного кроку обчислення. Мінімізація $|Q|$ є центральною задачею при синтезі ефективних протоколів популяцій, оскільки компактніша множина станів дозволяє будувати систему на базі найпростіших та найдешевших пристроїв, що робить всю мережу більш економічною та швидкою. Для простих предикатів, таких як предикати порогового типу $\varphi_k(x) \Leftrightarrow x \geq k$, навіть невелике $|Q|$ може забезпечити коректне обчислення, тобто системі достатньо «вміти рахувати» до певної межі, щоб прийти до правильної відповіді. Проте для складних предикатів Пресбургера може знадобитись значно більша кількість станів, адже такі завдання вимагають від агентів здатності кодувати заплутані арифметичні операції, порівняння та залишки від ділення.

З практичної точки зору, кожен стан агента відповідає певній комбінації локальних змінних, що описують поточну «роль» агента у виконанні протоколу. Тобто стан можна уявити як поточний статус або «професію» агента в конкретний

момент часу – він може бути або активним розповсюджувачем знань, або пасивним слухачем. Наприклад, у протоколах для обчислення порогових предикатів агент у стані $q \in \{0, 1, 2, ..., k\}$ може інтерпретуватися як агент, що «зберігає» значення q токенів. У протоколах для тверджень на основі переважної більшості учасники дійства поділяються на діяльних, які відстоюють власну думку, та споглядальних, котрі схильні погоджуватися. Тобто хід розв'язання задачі нагадує суспільне обговорення, де впевненіші члени громади поступово переконують тих, хто сумнівається, аж доки все товариство не дійде спільного висновку.

Важливою властивістю простору станів є його замкненість відносно функції переходів: застосування будь-якого переходу не може вивести систему за межі Q. Це означає, що межі роботи системи чітко окреслені заздалегідь, і жоден агент ніколи не опиниться в «неіснуючому» стані, який би призвів до збою алгоритму. Данна властивість гарантується самим визначенням протоколу і є принциповою відмінністю від розподілених систем із необмеженою пам'яттю, де простір станів може непередбачувано рости з часом. Отже, ми маємо справу з надзвичайно надійною архітектурою, яка не може гальмувати через брак пам'яті, оскільки всі можливі варіанти поведінки закладені в систему ще на етапі її створення.

Для аналізу просторової складності протоколів вводиться поняття просторової складності предиката φ – найменшого числа $k \in N$ такого, що будь-який протокол популяцій, який обчислює φ, має не менше k станів. Це формальне визначення дозволяє отримувати як нижні, так і верхні межі на мінімальний розмір протоколу для заданого предиката

2.3 Стохастична модель взаємодії агентів

Стохастична природа взаємодій є однією з визначальних рис моделі протоколів популяцій і суттєво відрізняє її від детермінованих моделей розподілених обчислень. Розуміння стохастичної моделі є необхідним як для коректного аналізу поведінки протоколів, так і для оцінки їхньої часової складності.

У базовій стохастичній моделі n агентів взаємодіють попарно. Для кожної пари агентів (i, j) існує незалежний пуассонівський процес з однаковою інтенсивністю λ. Коли пуассонівський процес для пари (i, j) спрацьовує, відбувається взаємодія: стани агентів i та j оновлюються відповідно до функції переходів δ. Таким чином, за одиницю часу відбувається в середньому $O(n^2)$ взаємодій, або O(n) взаємодій на одного агента.

Для аналізу часової складності вводиться поняття паралельного часу. Якщо протокол потребує T взаємодій для стабілізації, то його паралельний час становить O(T/n), оскільки за одиницю паралельного часу відбувається Θ(n) взаємодій. Таким чином, якщо T = O(n), то паралельний час є O(1), а якщо $T = O(n^2)$, то паралельний час є O(n).

Для цілей аналізу коректності протоколу (а не його часової складності) стохастичну модель замінюють детерміністичною з умовою справедливості. Умова справедливості вимагає, щоб у будь-якому прогоні σ множина inf(σ) була замкнена відносно відношення досяжності $\rightarrow^*$: якщо з конфігурації $C \in \inf(\sigma)$ досяжна конфігурація C', то $C' \in \inf(\sigma)$. Ця умова гарантує, що жоден застосовний перехід не буде нескінченно довго ігноруватися.

Стохастична модель взаємодії має важливі наслідки для синтезу протоколів. По-перше, протокол повинен бути коректним для будь-якого порядку взаємодій, що задовольняє умові справедливості, а не лише для деяких конкретних послідовностей. По-друге, швидкість збіжності протоколу до стабільного консенсусу залежить від стохастичних властивостей системи: зокрема, від того, наскільки ймовірним є виконання «прогресивних» переходів, що наближають систему до консенсусу.

Аналіз стохастичних протоколів популяцій часто використовує методи теорії марківських ланцюгів. Конфігурація системи утворює стан марківського ланцюга, а ймовірності переходів між конфігураціями визначаються стохастичним розкладом агентів.

Досягнення стабільного консенсусу відповідає поглинаючому стану ланцюга. Час до поглинання є ключовою характеристикою ефективності протоколу.

2.4 Модель стабільного консенсусу в протоколах популяцій

Стабільний консенсус є центральним поняттям моделі протоколів популяцій і визначає фундаментальну умову завершення обчислювального процесу. На відміну від класичних розподілених протоколів, де завершення обчислення є явним і агенти виконують певну дію «зупинки» (halt), у популяційних моделях цей процес є неявним і базується на досягненні динамічної стійкості всієї системи.

Отже, вузли продовжують взаємодіяти і змінювати свої внутрішні стани, проте ці зміни більше не впливають на глобальний результат обчислення. Отже, ми маємо справу не з фізичною зупинкою пристроїв, а з логічною фіксацією відповіді, коли подальші випадкові зустрічі лише підтверджують уже прийняте рішення, не маючи сили його змінити.

Формально конфігурація C визначається як b-консенсус, якщо функція виведення O(q) повертає однакове булеве значення b для всіх станів q, що входять до носія конфігурації supp(C). Інакше кажучи, якщо ми проведемо «миттєвий зріз» системи і опитаємо кожного агента, то кожен з них повинен видати ідентичну відповідь «так» або «ні».

Стабільним b-консенсусом, у свою чергу, називається такий стан C, з якого досяжні виключно інші b-консенсуси: для будь-якої конфігурації C', отриманої в результаті послідовності переходів $C \to^* C'$, значення виходу залишається незмінним. Інакше кажучи, система потрапляє у так звану «пастку стабільності», де результат обчислення стає інваріантним відносно будь-яких стохастичних збурень. Це означає, що навіть за умови продовження хаотичних зіткнень, популяція вже ніколи не зможе «передумати» або повернутися до невизначеного стану.

Ключова лема встановлює фундаментальну математичну еквівалентність між двома підходами до визначення коректності обчислень: через аналіз нескінченних послідовностей взаємодій та через аналіз досяжності станів у графі конфігурацій. Згідно з цією лемою, протокол P обчислює предикат φ тоді і тільки тоді, коли для кожної початкової конфігурації $C_0 \in N^I$ існує шлях до стабільного консенсусу C зі значенням $\varphi(C_0)$, причому стабільний консенсус із протилежним значенням є принципово недосяжним. Тобто, ця лема дозволяє нам значно спростити перевірку всієї системи: замість того, щоб аналізувати нескінченну кількість випадкових варіантів того, як агенти можуть зустрітися, нам достатньо довести, що правильна відповідь завжди «відкрита» для системи, а хибної відповіді в ній просто не існує як фінального стабільного пункту.

Отже, складне імовірнісне твердження про успіх обчислення перетворюється на чітку геометричну задачу про пошук шляху в лабіринті станів, що є фундаментальним для доведення коректності та безпеки синтезованих протоколів.

Техніка доведення коректності протоколів популяцій базується на двох основних математичних інструментах, які дозволяють «приборкати» стохастичну природу системи та гарантувати її збіжність. Перший інструмент – це інваріанти, тобто спеціальні функції $val : N^Q \to S$, значення яких залишається незмінним незалежно від того, які переходи виконуються між агентами. Інакше кажучи, інваріант – це певна «цифрова печатка» або відбиток пальця вхідних даних, який зберігається протягом усього обчислення, дозволяючи нам у будь-який момент часу пов'язати поточний хаотичний стан популяції з початковим і точно передбачити фінальний результат. Другий інструмент – це потенційні функції $pot : N^Q \to N$, які мають властивість монотонного незростання при виконанні кожного переходу.

Тобто, кожна корисна зустріч між агентами ніби «витрачає» частку віртуальної енергії системи, і оскільки цей показник не може падати нижче нуля, таке спадання гарантує, що система не може «зациклитися» у нескінченних марних повтореннях.

Таким чином, поєднання незмінного інваріанта, який вказує напрямок, та спадної потенційної функції, яка штовхає систему вперед, дозволяє математично довести, що протокол неминуче досягне стабільного консенсусу за скінченну кількість кроків.

Практична реалізація описаних аналітичних інструментів демонструється на прикладі протоколу P_{maj}, функціональне призначення якого полягає в обчисленні предиката більшості $\varphi(A, B) \Leftrightarrow A \leq B$. Фактично, робота цієї системи спрямована на з'ясування того, чи переважає обсяг однієї групи агентів над іншою, що є фундаментальною задачею для децентралізованих алгоритмів голосування.

Ключовим інваріантом у цьому випадку виступає функція $val(C) = C(B) + C(b) - C(A) - C(a)$. Її специфіка полягає у збереженні незмінного значення протягом усього обчислювального процесу, незалежно від послідовності зустрічей агентів. Власне, цей показник виконує роль математичної константи, яка фіксує чистий «баланс сил» у популяції: навіть коли окремі вузли змінюють свої стани після контакту, початкова різниця між даними нікуди не зникає, а лише трансформується у нові форми.

Паралельно з аналізом інваріантів доцільно використовувати потенційну функцію $pot(C) = C(A) + C(B)$, яка за своєю природою є монотонно незростаючою протягом усього прогону. Таку величину можна сприймати як міру «незавершеності» обчислення або віртуальну енергію системи: кожна корисна взаємодія, що наближає популяцію до відповіді, поступово вичерпує цей потенціал, доки будь-які зміни не припиняться зовсім. Поєднання незмінного орієнтира-інваріанта та спадної потенційної функції дозволяє математично довести неминучість досягнення фіналу. Це означає, що результат обчислення гарантовано зафіксується і стане незворотним, а фінальна відповідь буде цілком залежати від того інформаційного «заряду», який було внесено в систему на самому початку.

Логічним розвитком даного підходу є перехід до концепції майже самостабілізуючихся протоколів. Протокол класифікується як такий, якщо він здатний самостійно відновлювати коректний хід обчислень навіть за умови критичних збоїв або довільної ініціалізації.

Мова йде про сценарії, коли частина агентів починає роботу не з чистих вхідних даних, а перебуваючи у випадкових або помилкових станах. По суті, така архітектура не потребує ідеальних стартових умов, оскільки володіє внутрішнім механізмом виходу з хаосу та нівелювання помилок, спричинених зовнішніми завадами чи технічними дефектами вузлів.

Наявність подібної властивості суттєво підвищує рівень відмовостійкості розподіленої системи, дозволяючи їй зберігати працездатність навіть при частковому спотворенні даних або раптовому додаванні до групи нових, непідготовлених учасників. Як наслідок, здатність до автономного «самолікування» через прості локальні правила взаємодії стає вирішальним аргументом при синтезі надійних обчислювальних структур, призначених для роботи в агресивних або динамічних середовищах реального світу

2.5 Критерії ефективності синтезованих протоколів

Ефективність функціонування популяційного протоколу оцінюється через багатогранну систему взаємозалежних метрик, які в сукупності визначають життєздатність та якість синтезованої розподіленої системи. До зазначених критеріїв належать просторова складність, часова складність, відмовостійкість, а також виразна сила протоколу. Оптимальний синтез передбачає знаходження тонкого балансу між цими показниками. Наприклад, намагання максимально зменшити кількість внутрішніх станів, яка визначає просторову складність, може призвести до значного уповільнення швидкості досягнення консенсусу, що безпосередньо збільшує часову складність системи.

Тобто, розробник завжди стоїть перед вибором: зробити систему або надзвичайно «легкою» для пам’яті, або максимально швидкою у прийнятті рішень. Окрім цього, важливим аспектом є відмовостійкість, що визначає здатність алгоритму видавати правильний результат навіть за умови раптового зникнення агентів або появи шкідливих вузлів, а також виразна сила, яка вказує на складність логічних задач (предикатів), які в принципі може розв'язати дана модель.

Просторова складність протоколу, яка формально визначається потужністю множини станів |Q|, залишається ключовим індикатором для систем із жорстким лімітом обчислювальних ресурсів. Оскільки стан агента – це фактично вся його доступна пам'ять, мінімізація |Q| дозволяє впроваджувати такі протоколи на базі найпростіших мікросхем або наносенсорів. Для фундаментальних класів завдань, таких як предикати порогового типу $\varphi_k(x) \Leftrightarrow x \geq k$, математично доведено існування точних асимптотичних меж. Зокрема, встановлено, що просторова складність таких задач становить $\Theta(\log \log k)$, отже, кількість станів зростає надзвичайно повільно відносно зростання самого порогу k. Це означає, що навіть якщо нам потрібно підрахувати дуже велику кількість об'єктів, кожному окремому роботу достатньо мати лише мізерний обсяг пам'яті для збереження проміжних результатів. Важливо розуміти, що межа $o(\log \log k)$ є недосяжною для коректних протоколів, а отже, будь-яка спроба ще більше спростити систему призведе до її нездатності стабільно обчислювати порогове значення. Таким чином, знання цих меж дозволяє нам під час синтезу чітко розуміти, чи є наш алгоритм ідеальним з точки зору використання пам'яті, чи він потребує подальшої оптимізації.

Часова складність у моделі популяційних протоколів вимірюється через поняття паралельного часу, що визначається як відношення загальної кількості взаємодій, необхідних для досягнення стабільного консенсусу, до розміру популяції n. Тобто, цей показник відображає середню кількість «зустрічей», яку має провести кожен окремий агент, перш ніж вся система прийде до остаточної відповіді.

Відомим фундаментальним обмеженням у теорії децентралізованих систем є нижня асимптотична межа $\Omega(n)$ паралельного часу для більшості критично важливих завдань, серед яких особливе значення має предикат більшості. Вказане обмеження має глибокий логіко-математичний зміст і означає, що через виключно стохастичний характер контактів між анонімними вузлами інформація в популяції не може поширюватися миттєво.

Отже, існує об'єктивний теоретичний поріг швидкості дифузії даних, швидше за який жоден розподілений алгоритм без наявності глобального координатора

функціонувати не спроможний. Проте для сучасних стислих протоколів, архітектурні особливості яких дозволяють обчислювати довільні предикати Пресбургера, у нещодавніх дослідженнях було успішно досягнуто верхньої межі O(n) паралельного часу стабілізації. Подібний показник швидкодії є математично оптимальним, адже він фактично нівелює розрив між реальною швидкістю функціонування обчислювальної структури та її теоретичним лімітом, дозволяючи досягати стабільного консенсусу за мінімально можливий час у повністю децентралізованому та асинхронному середовищі.

Важливою метрикою оцінювання ефективності таких систем є стисліcть, яка виступає як специфічний критерій якості архітектурного синтезу, що встановлює чіткий функціональний зв'язок між просторовою складністю, тобто загальною кількістю внутрішніх станів вузла, та внутрішньою складністю самого обчислюваного предиката. Аналітично цей параметр демонструє, наскільки раціонально розроблений протокол використовує обмежений обсяг оперативної пам'яті агентів у міру ускладнення або розширення умов поставленого логіко-арифметичного завдання.

Розподілений протокол популяцій класифікується як стислий, якщо він спроможний обчислювати заданий предикат φ, використовуючи лише поліноміальну кількість локальних станів O(poly(|φ|)) відносно довжини компактного представлення або формульного запису цього предиката. Це дозволяє ефективно запобігати комбінаторному вибуху простору станів під час проектування логічних правил взаємодії, що є критично важливим для практичної реалізації обчислень на базі ресурсно-обмежених мікропристроїв та наносенсорів.

Тобто, навіть якщо логічна формула завдання стає довшою, кількість необхідних станів для кожного робота зростає помірно, а не вибухоподібно. Це особливо важливо при роботі з безквантворними формулами арифметики Пресбургера, де коефіцієнти представлені в двійковому кодуванні, що дозволяє компактно записувати операції з великими числами. Таким чином, стислі протоколи мають вирішальне значення для практичного впровадження, оскільки

вони дають змогу реалізовувати високорівневу інтелектуальну логіку навіть на пристроях із вкрай обмеженим обсягом пам'яті.

Відмовостійкість оцінюється здатністю протоколу зберігати функціональну коректність та гарантувати збіжність до істинного результату навіть за умови непередбачуваних змін у складі популяції або деструктивної поведінки окремих вузлів. Тобто, надійна система повинна вміти самовідновлюватися та коригувати свої обчислення, якщо частина даних була втрачена або навмисно спотворена.

У науковій літературі розрізняють кілька ключових моделей відмов, кожна з яких кидає свій виклик алгоритму. При динамічному додаванні агентів у систему можуть потрапляти нові учасники в довільних станах, що вимагає від протоколу здатності інтегрувати нову інформацію без перезапуску всього процесу та без руйнування вже досягнутого прогресу.

При видаленні агентів, що часто класифікують як модель снайпера, частина популяції може раптово зникнути під час обчислення, отже, протокол має бути спроектований так, щоб втрата навіть критичної кількості вузлів не призводила до зупинки системи або стабілізації на хибному значенні.

Найбільш складною для нейтралізації є модель візантійських відмов або модель ніндзя іншими словами, де деякі агенти діють як зловмисники, поводячись абсолютно довільно або навмисно транслюючи суперечливі стани для дезорієнтації решти популяції.

Інакше кажучи, це ситуація, коли в мережі з'являються агенти хаосу, і головне завдання алгоритму – це нівелювати їхній вплив через колективні правила взаємодії, щоб рішення чесної більшості завжди переважає. Для кожної з цих моделей формулюються специфічні математичні критерії коректності, які визначають межі виживання системи в агресивному середовищі.

Комплексним критерієм є оптимальність за часово-просторовими компромісами, що відображає фундаментальну залежність між обсягом доступної пам'яті агентів та загальною швидкістю прийняття рішення всією популяцією. Зменшення числа станів, як правило, призводить до збільшення часу обчислення, і навпаки. Точне встановлення цих залежностей є важливою теоретичною задачею.

Розробник завжди шукає баланс, щоб система не характеризувалася низькою швидкодією, але при цьому могла функціонувати на найпростішому та економічно доступному обладнанні. Важливою теоретичною задачею в цьому контексті є аналіз тихих протоколів, у яких агенти повністю припиняють зміну станів після знаходження загальної згоди.

Відомо, що для порогових предикатів, зокрема предикатів типу зграя птахів, існує суттєва розбіжність: надто стислі протоколи з мінімальною кількістю станів порядку подвійного логарифма від параметра k не здатні забезпечити такий режим роботи. Отже, навіть після формування правильної відповіді агенти продовжують взаємодіяти та змінювати стани. Це створює надлишковий обчислювальний шум і може збільшувати загальний час остаточної стабілізації.

Інакше кажучи, енергетична ефективність децентралізованої системи та її здатність припиняти активність після виконання завдання безпосередньо залежить від можливості виділити агентам додаткову пам'ять для фіксації фінального результату. Точне встановлення цих залежностей дозволяє проектувати розподілені системи, які ідеально збалансовані під конкретні технічні обмеження реального світу.

2.6 Порівняльний аналіз існуючих конструкцій протоколів популяцій

Еволюційний розвиток теорії популяційних протоколів призвів до виникнення цілої низки спеціалізованих обчислювальних конструкцій, кожна з яких пропонує власне бачення того, як знайти оптимальний баланс між обсягом пам'яті агентів, часом стабілізації та здатністю системи витримувати зовнішні збої.

Проведення глибокого порівняльного аналізу цих архітектур є обов'язковим етапом дослідження, оскільки це дозволяє науково обґрунтувати вибір конкретних механізмів для розроблюваного методу синтезу. Фактично, у контексті даної роботи синтез не розглядається як проста генерація випадкових правил; це складний процес підбору такої структури станів, яка б максимально відповідала жорстким обмеженням реального середовища.

Однією з найбільш фундаментальних моделей у цій галузі є протокол токенів P_{tok}. Він зазвичай використовується як база для вирішення порогових задач, відомих у науковій літературі як предикати «зграї птахів» $\varphi_k(x) \Leftrightarrow x \geq k$.

Основна концепція тут базується на тому, що кожен окремий агент виступає в ролі «носія знань», зберігаючи певну кількість токенів – від повної відсутності до досягнення цільового порогу k. Під час випадкових зустрічей вузли просто обмінюються цими одиницями інформації, що дозволяє поступово накопичувати сумарне значення в межах обмеженої групи агентів. Це означає, що простір станів системи $Q = \{0, 1, ..., k\}$ прямо залежить від величини шуканого числа, що стає серйозним бар'єром для практичного застосування у великих системах. Фактично, якщо параметр k досягає астрономічних значень (наприклад, $k \approx 10^{23}$), кількість необхідних станів перевищує фізичні можливості пам’яті мікросенсорів або нанороботів.

Інакше кажуть, це означає, що крихітний пристрій просто не зможе «запам'ятати», на якому саме кроці підрахунку він перебуває, якщо кількість варіантів буде настільки великою. Тим не менш, P_{tok} залишається еталоном для теоретичних досліджень завдяки прозорості своїх математичних інваріантів. Саме простота структури дозволяє легко будувати доведення коректності, використовуючи закон збереження загальної кількості токенів у системі, що стає надійним фундаментом для верифікації складніших та більш стислих алгоритмів.

Суттєвим кроком в оптимізації обчислювальних ресурсів став протокол степенів двійки P_{tok2}, який розглядається як вдосконалена модифікація базової токенної моделі через впровадження суворих обмежень на допустимі стани агентів.

Власне, у цій архітектурі пам'ять кожного окремого вузла налаштована на збереження інформації виключно у формі степенів числа два. Таке рішення дозволяє кардинально змінити характер використання простору станів: якщо в початковому протоколі P_{tok} кількість станів зростала лінійно разом із порогом k, то в даному випадку ми отримуємо простір $Q = \{0, 2^0, 2^1, ..., 2^p\}$, де при $k = 2^p$ загальний обсяг пам'яті визначається як $p+2 = O(\log k)$.

Інакше кажучи, замість того, щоб змушувати крихітний сенсор запам'ятовувати кожну одиницю підрахунку до мільярда, система використовує принцип двійкового кодування, що дозволяє оперувати величезними масивами даних за допомогою всього лише кількох біт пам’яті. Це досягнення стало першим нетривіальним верхнім обмеженням на просторову складність предикатів типу зграя птахів, продемонструвавши можливість логарифмічного стиснення інформації в анонімних розподілених середовищах.

Окрім компактності, важливою технічною характеристикою P_{tok2} є його приналежність до класу тихих протоколів. Це означає, що після досягнення глобального стабільного консенсусу всі агенти в популяції повністю припиняють зміну своїх внутрішніх станів, незалежно від подальших випадкових зустрічей. По суті, система володіє вбудованим механізмом заспокоєння, як тільки правильна відповідь знайдена і розповсюджена мережею, вузли переходять у стан енергоефективного спокою.

Для практичної реалізації на базі автономних пристроїв така властивість є критичною, адже вона дозволяє уникнути безглуздих обчислювальних циклів та марних витрат заряду акумулятора після завершення основного завдання. Таким чином, протокол степенів двійки поєднує в собі математичну витонченість логарифмічної складності та високу експлуатаційну надійність, що робить його ключовою ланкою в еволюції методів синтезу розподілених систем.

Найбільш прогресивними з погляду просторової ефективності вважаються дуже стислі протоколи P_{tiny}, розроблені у межах досліджень 2024 року. Ці алгоритми стали першими в історії галузі, що змогли досягти теоретичного мінімуму $\Omega(\log \log k)$ для просторової складності.

Фактично, це означає неймовірну економію ресурсів: якщо для обробки величезних чисел раніше потрібні були сотні станів, то тепер достатньо лише декількох, що дозволяє запускати такі системи на мікросхемах із мінімальним обсягом пам'яті.

На відміну від попередніх тихих конструкцій, ці протоколи характеризуються динамічною поведінкою. Це означає, що агенти продовжують

активний обмін даними навіть після того, як відповідь знайдена, оскільки система постійно проводить внутрішню перевірку власної коректності.

В основі архітектури P_{tiny} лежить імітація роботи лічильникових машин, доповнена механізмом вибору лідера та спеціальними фільтрами для відсіювання помилок. Іншими словами, група агентів працює як єдиний віртуальний комп'ютер, де один вузол координує дії інших, постійно звіряючи отримані дані із заданим алгоритмом. Такий підхід забезпечує властивість майже повної самостабілізації, що надає системі унікальної відмовостійкості. Власне, протокол здатний самостійно виправляти випадкові збої, повертаючись до правильного результату без втручання ззовні, що робить його ідеальним рішенням для роботи в нестабільних або ворожих середовищах.

Окрему категорію в ієрархії обчислювальних моделей складають стислі протоколи для роботи з довільними предикатами Пресбургера. Важливість цього класу задач зумовлена тим, що згідно з фундаментальною теоремою Англуїна, популяційні протоколи здатні обчислювати саме предикати Пресбургера, що еквівалентно класу напівлінійних множин. Будь-яка логічна операція, яку в принципі може виконати анонімна розподілена система, підпадає під цю категорію.

Якщо перша конструкція 2020 року забезпечувала компактність станів ціною дуже повільної роботи, що вимірювалася експоненціальним часом, то вдосконалена модель 2024 року здійснила справжній прорив, досягнувши лінійного паралельного часу O(n). Фактично, це означає, що швидкість отримання результату тепер прямо залежить від кількості агентів, що дозволяє системі майже миттєво реагувати на зміни навіть у мільйонних популяціях.

Ключова стратегія побудови таких систем базується на декомпозиції – процесі розділення одного складного логічного завдання на низку простіших підпредикатів. Замість того, щоб намагатися створити один громіздкий алгоритм, завдання розбивається на елементарні лінійні нерівності та операції порівняння за модулем. Для кожної такої частки створюється окремий стислий субпротокол, який функціонує максимально ефективно у своїй вузькій зоні відповідальності. Це означає, що загальна логіка системи формується через модульну композицію

результатів окремих підпротоколів, що значно спрощує верифікацію всієї структури.

Особлива роль у моделі 2024 року відведена спеціальному механізму широкомовного розсилання та агрегування даних. У цій архітектурі використовується внутрішній механізм вибору лідера, який бере на себе роль координатора: він збирає вхідні сигнали від популяції, спрямовує їх до відповідних субпротоколів, а потім акумулює їхні висновки для формування остаточної відповіді. Такий підхід дозволяє уникнути хаотичного поширення інформації, яке було причиною повільної роботи ранніх моделей. Як наслідок, координація через лідера дозволяє системі працювати злагоджено, як єдиний обчислювальний механізм, де кожен вузол точно знає свою роль у поточному кроці алгоритму.

Такий модульний підхід не лише прискорює обчислення, а й кардинально спрощує процес синтезу складних розподілених структур. Можливість комбінувати готові стислі блоки для вирішення довільних задач робить систему більш зрозумілою для розробника та гнучкою до змін у специфікації. Отже, сучасні методи синтезу дозволяють створювати протоколи, які одночасно є і пам'яте-ефективними та стислими, і швидкими, що раніше вважалося технічно недосяжним компромісом у теорії розподілених обчислень.

Наведений аналіз свідчить про суттєвий прогрес у теорії протоколів популяцій: від простих конструкцій з лінійною просторовою складністю до стислих протоколів з оптимальними часово-просторовими характеристиками. Проте залишаються відкриті питання: зокрема, чи можна поєднати оптимальну просторову складність з лінійним паралельним часом для довільних предикатів Пресбургера, та яким чином підвищити відмовостійкість стислих протоколів.

Для комплексного оцінювання еволюції наукової думки у зазначеному напрямі та визначення місця кожної архітектури в ієрархії розподілених систем доцільно провести їх системне зіставлення. Кожна з існуючих модифікацій розроблялася для вирішення конкретного технологічного протиріччя між обсягом пам'яті вузла та швидкістю дифузії інформації в мережі.

Базова архітектура P_{tok} заклала фундаментальні принципи взаємодії, забезпечуючи високу швидкість збіжності за рахунок значних витрат пам'яті кожного агента, що робить її непридатною для ультрамалих сенсорних мереж. Подальший розвиток у формі моделі P_{tok2} дозволив логарифмічно зменшити просторову складність, проте це призвело до пропорційного уповільнення обчислювального процесу. Радикальним кроком у мінімізації пам'яті стала концепція Ptiny, яка досягла теоретичного мінімуму станів. Цей протокол продемонстрував унікальні властивості стійкості до зовнішніх збурень і здатність до самовідновлення в агресивному середовищі, проте платою за таку компактність стало суттєве квадратичне зростання часу стабілізації та неспроможність системи переходити в абсолютно тихий режим роботи після досягнення консенсусу.

Новий етап розвитку пов'язаний із моделюванням складних логічних виразів Пресбургера. Перша успішна спроба реалізації такого протоколу у 2020 році забезпечила математичну коректність і компактність станів, але характеризувалася практично неприйнятним експоненціальним часом виконання. Модифікація 2024 року вирішила цю проблему завдяки інтеграції згаданого лінійно-часового координатора, хоча й змусила розробників відмовитися від тихої поведінки системи на користь безперервної динамічної самоперевірки.

Детальний порівняльний аналіз архітектурних рішень, їхніх математичних меж та обчислювальних можливостей наведено нижче.

Варто наголосити, що вибір оптимальної конструкції протоколу завжди диктується специфікою цільового середовища розгортання та апаратними обмеженнями обчислювальних вузлів. Представлені в таблиці 2.1 дані унаочнюють еволюційний перехід від монолітних архітектур до гнучких модульних рішень, де кожен крок оптимізації просторової складності вимагає переосмислення часових характеристик та моделей поведінки системи. Систематизація цих параметрів дозволяє не лише оцінити поточний стан розвитку теорії популяційних обчислень, а й сформувати чітку методологічну базу для розробників. Це дає змогу прогнозувати поведінку децентралізованої мережі ще на етапі проектування, спираючись на строгі математичні межі обчислювальної складності та враховуючи

критичний баланс між енергоефективністю, швидкістю стабілізації й автономною стійкістю системи до внутрішніх збоїв.

Таблиця 2.1 – Порівняння основних конструкцій протоколів популяцій за ключовими критеріями ефективності

	P_{tok}	P_{tok2}	P_{tiny}	Ppresburger (2020)	Ppresburger (2024)
Просторова складність	O(k)	O(log k)	O(log log k)	O(poly|φ|)	O(poly|φ|)
Часова складність	O(n)	O(n log k)	$O(n^2)$	exp(n)	O(n)
Тихий	Так	Так	Ні	Так	Ні
Стислий	Ні	Так	Так	Так	Так
Відмовостійкість	Базова	Базова	Висока	Базова	Базова

2.7 Висновки до другого розділу

У другому розділі розроблено формальну модель синтезу розподілених комп'ютерних систем на основі протоколів популяцій та проведено детальний аналіз її компонентів.

Встановлено, що протокол популяцій визначається кортежем P = (Q, δ, I, O) і є повною математичною специфікацією поведінки розподіленої системи з анонімними агентами та скінченним простором станів. Формально визначено поняття конфігурації, переходу, прогону та стабільного консенсусу як основних елементів моделі.

Досліджено структуру агентів та простору станів. Показано, що стан агента є єдиним носієм інформації, а розмір |Q| безпосередньо визначає просторову складність системи. Мінімізація |Q| є центральною задачею синтезу, оскільки вона визначає ресурсомісткість протоколу.

Описано стохастичну модель взаємодії агентів на основі пуассонівських процесів та введено поняття паралельного часу як основної міри часової складності протоколів популяцій. Показано, що для цілей аналізу коректності стохастична модель може замінюватися детерміністичною з умовою справедливості.

Формалізовано модель стабільного консенсусу та основні техніки доведення коректності протоколів: інваріанти та потенційні функції. Встановлено еквівалентність між різними визначеннями коректності, що спрощує процес верифікації протоколів.

Визначено та обґрунтовано критерії ефективності протоколів популяцій: просторова складність, часова складність, стислість та відмовостійкість. Показано, що між цими критеріями існують суттєві компроміси, урахування яких є необхідним при синтезі протоколів.

Проведено порівняльний аналіз основних відомих конструкцій протоколів популяцій. Встановлено, що найкращі відомі протоколи досягають просторової складності $O(\log \log k)$ для порогових предикатів та $O(\text{poly}|\varphi|)$ станів при лінійному паралельному часі для довільних предикатів Пресбургера. Ці результати є теоретичною основою для розробки методу синтезу, описаного в наступному розділі.

Окрему увагу в розділі приділено аналізу поведінки протоколів у нестабільних розподілених середовищах, зокрема в умовах динамічної зміни кількості агентів або виникнення апаратних збоїв. Проведене дослідження показало, що забезпечення відмовостійкості класичних моделей найчастіше вимагає експоненційного розширення простору станів $|Q|$, що суперечить базовому критерію мінімізації ресурсомісткості агента. Виявлення цих критичних обмежень та компромісів дозволило чітко окреслити математичні межі застосовності відомих конструкцій і обґрунтувати необхідність розробки нових, гнучкіших підходів до проектування правил взаємодії.

Таким чином, сформований у другому розділі теоретичний апарат та аналітичні висновки стали основою для переходу від емпіричного проектування до системного інженерного синтезу. Систематизація критеріїв ефективності, а також

формалізація умов справедливості стохастичних процесів дали змогу сформулювати чіткі оптимізаційні обмеження для простору переходів δ. Це закладає необхідне методологічне підґрунтя для побудови та практичної реалізації цілісного методу автоматизованого синтезу самоорганізовних обчислювальних систем.

3 МЕТОД СИНТЕЗУ РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМ НА ОСНОВІ ПРОТОКОЛІВ ПОПУЛЯЦІЙ

3.1 Загальна схема методу синтезу протоколів популяцій

Запропонований метод синтезу розподілених комп'ютерних систем на основі протоколів популяцій являє собою систематичну процедуру автоматизованої побудови коректних та ефективних правил взаємодії для заданих обчислювальних задач. Фактично, цей підхід виступає як міст між високорівневою логічною специфікацією, тобто описом того, що система має обчислити та низькорівневою реалізацією у формі скінченних автоматів.

Метод охоплює повний цикл проектування: від формальної специфікації предиката до отримання готового протоколу з математично доведеними властивостями коректності, просторової та часової складності, а також відмовостійкості. Використання такої процедури дозволяє уникнути суб'єктивних помилок при ручному проектуванні правил переходів, що особливо важливо для анонімних систем, де локальні помилки в коді окремих агентів можуть призводити до непередбачуваної емерджентної поведінки всієї популяції.

Загальна схема методу складається з чотирьох основних етапів, що послідовно виконуються у процесі синтезу протоколу, забезпечуючи перехід від математичної абстракції до функціонуючого алгоритму.

Першим етапом аналіз та формалізація предиката. На цьому кроці задача обчислення формулюється як предикат φ: $N^I \rightarrow \{0, 1\}$ над мультимножинами вхідних даних. Предикат представляється у вигляді безквантворної формули арифметики Пресбургера (QFPR), що є канонічним представленням для класу напівлінійних множин – єдиного класу задач, обчислюваних у моделі популяційних протоколів. Формула піддається декомпозиції на атомарні підпредикати двох типів: порогові умови вигляду $\Sigma c_i x_i \geq k$ та предикати порівняння за модуле вигляду $\Sigma c_i x_i$ equiv (mod m). Це означає, що будь-яке складне обчислювальне завдання розбивається на елементарні блоки, для кожного з яких згодом буде синтезовано окремий стислий субпротокол. Власне, на цьому етапі

визначається довжина представлення $|\varphi|$, яка безпосередньо впливає на підсумкову просторову складність: ефективна декомпозиція дозволяє мінімізувати кількість станів агентів, забезпечуючи стислість підсумкового рішення.

На даному етапі для кожної логічної одиниці φ_i, отриманої в результаті декомпозиції, розробляється автономний обчислювальний модуль – субпротокол P_i. Власне, вибір архітектури субпротоколу прямо залежить від математичної природи конкретного підпредиката. Для порогових предикатів застосовується ієрархічна конструкція на основі рівнів. В основі цієї моделі лежить організація простору станів у вигляді впорядкованих шарів, що дозволяє системі ефективно накопичувати та порівнювати значення зважених сум $\Sigma c_i x_i$ із заданим порогом k. Іншими словами, агенти поступово просуваються за ієрархією рівнів, що забезпечує стабільне зростання точності підрахунку без надмірного розширення пам'яті.

Третій етап це композиція субпротоколів та розподіл вхідних даних. Субпротоколи об'єднуються паралельно шляхом злиття їхніх просторів станів та множин переходів. Вводяться додаткові стани та переходи для розподілу вхідних агентів між субпротоколами відповідно до коефіцієнтів у представленні підпредикатів. Реалізується механізм стартового поштовху, що ініціює обчислення у разі малої кількості вхідних агентів.

Четвертий етап займає побудова механізму консенсусу. До складеного протоколу додається компонент, що визначає значення виходу кожного субпротоколу на основі носія термінальної конфігурації та обчислює булеву комбінацію цих значень відповідно до структури предиката φ. Результат транслюється у стабільний консенсус усієї популяції.

Запропонований метод гарантує комплексні властивості синтезованого протоколу, забезпечуючи його коректність через точне обчислення заданої функції для будь-якої початкової конфігації, а також стислість, завдяки якій кількість станів залишається поліноміальною відносно розміру вхідної формули φ. Крім того, досягається висока оптимальність за часом, оскільки паралельний час

стабілізації становить порядок O(n), що повністю відповідає відомим нижнім обмеженням для більшості нетривіальних предикатів.

3.2 Алгоритмічне забезпечення методу синтезу

Алгоритмічне забезпечення методу синтезу реалізує формальні процедури побудови протоколів популяцій для кожного з чотирьох етапів. Нижче наведено деталізований опис ключових алгоритмів та їхніх властивостей.

Першим кроком є декомпозиція предиката: процедура приймає на вхід формулу φ у вигляді безквантворної формули арифметики Пресбургера та повертає набір атомарних підпредикатів $\varphi_1, ..., \varphi_s$ разом зі структурою їхньої булевої комбінації. У межах цієї процедури виконується синтаксичний розбір формули, нормалізація коефіцієнтів шляхом зведення до стандартної форми зі знаком та перевірка розмірності. Обчислювальна складність є лінійною відносно $|\varphi|$ завдяки використанню процедури елімінації кванторів Пресбургера.

Наступним етапом є синтез субпротоколу для порогового предиката. Побудова субпротоколу P_i здійснюється для предиката вигляду $\psi(x_1, ..., x_t) \Leftrightarrow a_1 \cdot x_1 + ... + a_t \cdot x_t \geq k$. Визначається параметр $h := \min\{i : 2^i \geq 2m\}$, де $m = \max\{|a_1|, ..., |a_t|, |k|\}$. Простір станів будується як $Q_2 = \{-2^h, ..., -2^0, 0, 2^0, ..., 2^h\}$. Переходи реалізують правило накопичення: якщо два агенти зустрічаються і сума їхніх значень може бути представлена в Q_2, один агент збирає обидва значення, тоді як інший переходить у стан 0. Розмір субпротоколу становить $O(\log m) = O(\log |\varphi|)$ станів.

Синтез субпротоколу для предиката за модулем передбачає побудову субпротоколу P_i для предиката вигляду $\psi(x_1, ..., x_t) \Leftrightarrow a_1 \cdot x_1 + ... + a_t \cdot x_t \equiv k \pmod m$. Визначається $h := \max\{i : 2^i < m\}$ як найбільший степінь двійки, менший за m. Простір станів $Q_1 = \{0, 2^0, 2^1, ..., 2^h\}$ є множиною степенів двійки. Ключовою операцією є взяття залишку: якщо два агенти у стані 2^h зустрічаються, вони переходять у стани 2^a та 2^b, де $2^{(h+1)} - m = 2^a + 2^b$. Розмір субпротоколу становить $O(\log m)$ станів.

Окремою процедурою є розподіл вхідних даних між субпротоколами. Для кожного вхідного стану x_i вводиться відповідний стан у спільному просторі. Оскільки пряме розщеплення одного агента на багато є неможливим у моделі попарних взаємодій, реалізується каскадний механізм: спочатку два агенти об'єднуються в одного проміжного, після чого проміжний агент розщеплюється за рахунок агентів-резервуарів відповідно до коефіцієнтів підпредикатів. Кількість додаткових станів, що вводяться цією процедурою, є лінійною відносно кількості субпротоколів.

Завершальною процедурою є побудова механізму консенсусу. До складеного протоколу додається компонент визначення виходу: для кожного стану термінальної конфігурації кожного субпротоколу визначається, чи свідчить присутність агентів у цьому стані про значення 0 або 1 відповідного підпредиката. На основі цього будується булева схема із логічних вентилів AND, OR, NOT, що обчислює булеву комбінацію виходів субпротоколів, де кожному вентилю відповідає один агент у протоколі. Вихідний вентиль транслює отриманий результат у стабільний консенсус усієї популяції.

3.3 Синтез протоколів для предикатів порогового типу

Порогові предикати є одним із двох фундаментальних класів предикатів Пресбургера. Вони мають вигляд $\varphi(x_1, ..., x_t) \Leftrightarrow a_1 \cdot x_1 + ... + a_t \cdot x_t \geq k$, де $a_1, ..., a_t, k \in Z$. Синтез протоколів для порогових предикатів є центральною задачею методу, оскільки саме ці предикати найчастіше зустрічаються у практичних застосуваннях розподілених систем.

Особливий підклас порогових предикатів – висхідно-замкнені порогові предикати,у яких усі коефіцієнти a_i є додатними. Ці предикати мають властивість монотонності: якщо $\varphi(C) = 1$, то $\varphi(C') = 1$ для будь-якої $C' \geq C$. Предикати зграя птахів $\varphi_k(x) \Leftrightarrow x \geq k$ є найпростішим прикладом висхідно-замкнених порогових предикатів.

Для висхідно-замкнених порогових предикатів запропонований метод використовує конструкцію на основі рівнів, узагальнюючи базовий протокол P_{lvl}. Уявімо вежу висотою k. Кожен агент у початковому стані x_i займає інтервал рівнів висоти a_i, починаючи з першого рівня. Коли два агенти зустрічаються та їхні інтервали перетинаються, той агент, чий інтервал починається вище, піднімається на один рівень вгору. Щойно один агент досягає рівня k, протокол переходить у режим прийняття.

Формально протокол P_{lvl2} визначається так. Простір станів $Q = \{\{i, ..., j\} : 1 \leq i \leq j \leq k\}$ є множиною всіх інтервалів від 1 до k. Вхідні стани $I = \{\{1, ..., a_i\} : i \in \{1, ..., t\}\}$: агент у початковому стані x_i займає рівні від 1 до a_i. Перехід підйому: $(q, p) \to (q+1, p)$ для $q, p \in Q$ таких, що $k \notin p \cup q$, $q \cap p \neq \varnothing$ та $\min(q) \geq \min(p)$, де $q+1 = \{i+1 : i \in q\}$. Перехід прийняття: $(q, p) \to (q, p \cup \{\max(p)+1, ..., k\})$ для $q \in Q$, $k \in q$. Функція виходу: $O(q) = 1$ тоді і тільки тоді, коли $k \in q$.

Коректність протоколу P_{lvl2} доводиться за допомогою інваріанту: у будь-якій конфігурації загальна довжина всіх інтервалів, зважена на кількість агентів, дорівнює загальній довжині інтервалів у початковій конфігурації. Це означає, що якщо сума $a_1 \cdot C_0(x_1) + ... + a_t \cdot C_0(x_t) \geq k$, то сума довжин інтервалів достатня для досягнення рівня k, і протокол прийме. У протилежному випадку протокол відхилить.

Для загальних порогових предикатів, у яких коефіцієнти можуть бути від'ємними, використовується гомогенна конструкція. У цьому випадку $k = 0$ і коефіцієнти можуть бути будь-якими цілими числами. Протокол P_{hom} будується за аналогією з протоколом більшості P_{maj}: агенти з різними знаками «скасовують» один одного, а пасивні агенти наслідують думку активних. Простір станів Q включає стани від a_1 до a_t (для від'ємних коефіцієнтів) та стан $-\varepsilon$ для пасивних агентів.

Загальний пороговий предикат зводиться до комбінації висхідно-замкненого та гомогенного: $\varphi(x) \Leftrightarrow a \cdot x \geq k \iff (a \cdot x \geq 0) \wedge (a \cdot x \geq k)$. Перший субпредикат є

гомогенним (k=0), другий – висхідно-замкненим (якщо k > 0). Синтезовані субпротоколи для цих двох підзадач поєднуються за схемою.

Ключовою властивістю всіх протоколів для порогових предикатів є їхня відмовостійкість: при правильно обраних параметрах вони робастно обчислюють предикат навіть за умов видалення агентів під час обчислення. Зокрема, протокол P_{lvl2} є робастним у тому сенсі, що після видалення щонайбільше s агентів він стабілізується до $\varphi(C_0 - C)$, де C – мультимножина видалених агентів, $|C| \leq s$.

3.4 Синтез протоколів для предикатів Пресбургера

Довільний предикат Пресбургера є булевою комбінацією порогових предикатів та предикатів за модулем. Синтез протоколу для такого предиката включає побудову субпротоколів для кожного атомарного підпредиката та їх наступну композицію. Цей підхід, розроблений у роботах Блондіна, Еспарзи та співавторів і вдосконалений Чернером, Гуттенбергом, Гельфріхом та Еспарзою, дозволяє будувати стислі протоколи з оптимальними часово-просторовими характеристиками.

Базовим будівельним блоком є субпротокол для предиката за модулем $\psi(x_1, ..., x_t) \Leftrightarrow a_1 \cdot x_1 + ... + a_t \cdot x_t \equiv k \pmod m$. Ідея конструкції полягає в тому, що кожен агент зберігає певне «значення» – степінь двійки від 2^0 до 2^h, де $h = \max\{i : 2^i < m\}$. При зустрічі двох агентів з однаковими значеннями 2^i один з них забирає обидва значення, переходячи у стан $2^{(i+1)}$, якщо $2^{(i+1)} < m$. Якщо обидва агенти перебувають у стані 2^h, виконується операція взяття залишку: вони переходять у стани 2^a та 2^b відповідно до розкладу $2^{(h+1)} - m = 2^a + 2^b$.

Для прикладу розглянемо предикат $\varphi_1(x, y) \Leftrightarrow x + 2y \equiv 1 \pmod 5$. Тут $m = 5$, $h = 2$ (оскільки $2^2 = 4 < 5 \leq 2^3 = 8$). Простір станів $Q_1 = \{0, 1, 2, 4\}$. Початкові стани: агент у x стартує у стані 1 (коефіцієнт $a_1 = 1$), агент у y – у стані 2 (коефіцієнт $a_2 = 2$). Операція взяття залишку: якщо два агенти у стані 4 зустрічаються, вони переходять у стани 1 та 2, оскільки $2^3 - 5 = 3 = 2^1 + 2^0$. Таким чином, загальне

значення (сума станів) завжди залишається рівним сумі вхідних значень за модулем 5.

Субпротокол для порогового предиката $\varphi_2(x, y) \Leftrightarrow 2x - y \geq 3$ будується на основі простору станів $Q_2 = \{-4, -2, -1, 0, 1, 2, 4\}$. Агент у стані x стартує у стані 2 з коефіцієнтом 2, агент у y – у стані -1 з коефіцієнтом -1. При зустрічі двох агентів один збирає обидва значення, якщо їхня сума може бути представлена в Q_2, інший переходить у 0.

Для побудови протоколу для $\varphi = \varphi_1 \wedge \varphi_2$ субпротоколи P_1 та P_2 об'єднуються паралельно: $Q = Q_1 \cup Q_2 \cup \{R, x, y, X, Y\}$, де R – резервуарний стан, спільний для обох субпротоколів, x та y – вхідні стани, X та Y – проміжні стани для розподілу вхідних даних. Переходи розподілу: $(x, x) \rightarrow (X, R)$, $(X, R) \rightarrow (2\_1, 4\_2)$ реалізують розщеплення агентів x між субпротоколами з відповідними коефіцієнтами.

Стисліть синтезованого протоколу забезпечується тим, що розмір кожного субпротоколу є $O(\log |\varphi_i|)$, а загальна кількість субпротоколів та вентилів булевої схеми є $O(|\varphi|)$. Таким чином, загальний розмір протоколу є $O(|\varphi| \cdot \log |\varphi|) = O(\text{poly}|\varphi|)$, що відповідає визначенню стислості.

Часова складність протоколу аналізується за допомогою потенційних функцій. Для кожного субпротоколу будується строга потенційна функція Φ_i: $N^{Q_i} \rightarrow N$, що є лінійною та строго спадною при виконанні будь-якого переходу. Наявність строгої потенційної функції гарантує, що субпротокол досягає термінальної конфігурації за $O(n^2)$ паралельного часу. Для досягнення лінійного $O(n)$ паралельного часу використовується поняття швидко спадних потенційних функцій – це додаткова властивість, що гарантує наявність достатньої кількості пар агентів, здатних зменшити потенціал на кожному кроці.

3.5 Методи побудови відмовостійких протоколів популяцій

Відмовостійкість є критичною властивістю для розподілених систем, що функціонують в умовах реального середовища. У контексті протоколів популяцій розрізняють дві основні моделі відмов: додавання агентів та видалення агентів.

Метод синтезу, що розробляється, орієнтований насамперед на модель видалення агентів як більш реалістичну для практичних систем.

Ключовою перешкодою для побудови відмовостійких протоколів є централізація інформації. Більшість класичних конструкцій протоколів популяцій ґрунтуються на накопиченні критично важливих даних в одному або кількох специфічних вузлах, які виконують функції одноосібних лідерів або спеціалізованих агентів-накопичувачів. Раптове видалення або апаратний збій такого детермінованого агента призводить до безповоротної втрати інформаційного масиву та, як наслідок, до повної некоректності подальшої роботи всього протоколу через руйнування координаційного зв'язку. Саме тому сучасні відмовостійкі протоколи повинні базуватися на принципах розподіленого консенсусу та підтримувати децентралізоване зберігання інформації, за якого втрата будь-якого окремого елемента мережі не здатна порушити цілісність обчислювального процесу.

Формально, протокол P стійко та автономно обчислює предикат φ, якщо для будь-яких $s \in N$, $C_0 \in N^I$ та $C' \in N^Q$ таких, що $C_0 \rightarrow^{(-s)} C'$, тобто C_0 досягає C' з s операціями видалення агентів. Це означає, що після s видалень протокол може видати лише ті відповіді, які були б правильними для початкових конфігурацій, що відрізняються від C_0 щонайбільше на s агентів.

Зазначений вираз виступає фундаментальним математичним критерієм, який безпосередньо визначає здатність популяційного протоколу зберігати свою функціональну коректність та цілісність в умовах динамічної й агресивної зміни чисельного складу мережі. Цей режим деструктивного впливу описується в літературі як модель відмов типу видалення агентів або модель снайпера, а відповідні кількісні залежності обчислюються за формулою 3.1.

$$\mathrm{out}(C\prime) \subseteq \{\varphi(C_0 - C) : C \leq C_0, |C| \leq s\}, \qquad (3.1)$$

де C_0 – початкова конфігурація системи, сформована на основі чистих вхідних даних;

C' – термінальна конфігурація, якої система досягає після того, як над нею було виконано s операцій раптового видалення (зникнення) довільних вузлів ($C_0 \to^{(-s)} C'$);

C – мультимножина, що репрезентує конкретних агентів, які безповоротно зникли з системи під час її прогону ($C \leq C_0$), при цьому їхня загальна кількість обмежена параметром відмовостійкості ($|C| \leq s$).

$\varphi(C_0 - C)$ – істинне значення логічного предиката, розраховане для «урізаної» популяції, тобто без урахування видалених компонентів;

out (C') – множина булевих значень виходу, які транслюють агенти у фінальному стані.

Для перевірки робастності протоколу розроблено ефективний достатній критерій – локальну робастність. Протокол P є локально робастним, якщо для будь-якої конфігурації C', будь-якого переходу $t = (q_1, q_2 \to q'_1, q'_2) \in \delta$ та будь-якого $b \in out(C')$ існує $q \in \{q_1, q_2, q_1+q_2\}$, таке що $b \in out(C' - q' + q)$, де $q' = q'_1$ або $q' = q'_2$ – той з вихідних станів переходу, що присутній у C'. Ця властивість перевіряється локально для кожного переходу окремо, що суттєво спрощує процес верифікації.

Теорема про достатність,ьяка в роботі представлена як лема під номером 52, стверджує, що за умови, коли протокол P є локально робастним і обчислює предикат φ, то P робастно обчислює φ. Доведення проводиться індукцією за довжиною виконання $C_0 \to^* C' + D$, де D – мультимножина видалених агентів. На кожному кроці локальна робастність гарантує можливість «скасувати» останній перехід, зберігаючи правильність виходу.

Відмовостійкий протокол для висхідно-замкнених порогових предикатів будується на основі рівневої конструкції P_{lvl2}. Ключова властивість, що забезпечує робастність: наявність агента на рівні i свідчить про те, що сумарна висота всіх інтервалів перевищує k - i. Після видалення одного агента в стані q сумарна висота зменшується на |q|, де |q| – довжина інтервалу. Це відповідає видаленню |q| одиниць із суми $a_1 \cdot x_1 + ... + a_t \cdot x_t$, що є коректною поведінкою за визначенням робастності.

Відмовостійкий протокол для гомогенних порогових предикатів є узагальненням протоколу більшості. Ключовою властивістю є те, що функція

val(C) = Σ q·C(q) є інваріантом. Після видалення агента у стані q значення val зменшується на q. Оскільки вихід визначається знаком val(C), а вилучення агента змінює val(C) на $|q| \leq \max\{|a_i|\}$, протокол може видати лише відповіді, що відповідають конфігураціям із val, зміненим не більше ніж на |q|. Це відповідає визначенню робастності.

Для моделі додавання агентів запропонований метод базується на властивості майже самостабілізації. Протокол є майже самостабілізуючим, якщо він досягає правильного консенсусу навіть за умови, що деякі агенти стартують у довільних станах, за умови, що кількість агентів у правильних початкових станах не менша за |Q|. Ця властивість виникає природно для дуже стислих протоколів з O(log log k) станами завдяки тому, що вони не є 1-aware.

3.6 Просторова складність синтезованих протоколів

Просторова складність є одним із ключових критеріїв якості синтезованого протоколу і визначається кількістю станів |Q|. Аналіз просторової складності включає встановлення верхніх меж на основі конструкцій та нижніх меж на основі теоретичних аргументів. Точне встановлення обох меж для різних класів предикатів є одним із фундаментальних результатів теорії протоколів популяцій.

Для предикатів «зграя птахів» $\varphi_k(x) \Leftrightarrow x \geq k$ встановлено такі межі. Верхня межа: протокол P_{tok2} використовує O(log k) станів, а протокол P_{tiny} – O(log log k) станів. Нижня межа: будь-який протокол для φ_k потребує Ω(log log k) станів (Теорема 17). Таким чином, просторова складність φ_k є Θ(log log k), і протокол P_{tiny} є асимптотично оптимальним. Це є одним з найглибших теоретичних результатів у цій галузі, оскільки нижня межа доводиться нетривіальним методом з використанням теорії мереж Петрі та теореми Раккоффа.

Доведення нижньої межі Ω(log log k) базується на наступній схемі. Нехай P – протокол для φ_k з $m = |Q|$ станами, $k > 2^{(2^m)}$. Розглядається конфігурація $C_0 = (k-1)\cdot q_{init}$ – найбільший відхиляючий вхід. Показується, що C_0 досягає деякої конфігурації C_{many}, у якій кожен стан має принаймні 2^m агентів. З C_{many} досягається

стабільний 0-консенсус C_{cons}. Потім за допомогою теореми Раккоффа встановлюється існування мультимножини D у носії C_{cons}, таке що C_{cons} + D залишається стабільним 0-консенсусом. Однак $C_0 + |D| \cdot q_{init}$ досягає C_{cons} + D, що суперечить умові $\varphi_k(C_0 + |D| \cdot q_{init}) = 1$.

Для довільних предикатів Пресбургера синтезований метод забезпечує стисліcть у такому сенсі: протокол для предиката φ має O(poly|φ|) станів. Зокрема, якщо φ є кон'юнкцією s атомарних підпредикатів $\varphi_1, ..., \varphi_s$, кожен з яких має розмір $|\varphi_i|$ та коефіцієнти розміром не більше M, то загальна кількість станів є O(s · log M · log s). При s = O(|φ|) та log M = O(log |φ|) отримуємо O(|φ| · log |φ|) = O(poly|φ|).

Існують також більш точні верхні обмеження для специфічних класів предикатів. Для лінійних предикатів вигляду $a \cdot x + b \cdot y \geq k$ кількість станів є O(log k + log |a| + log |b|). Для модульних предикатів $x \equiv r \pmod m$ – O(log m). Ці оцінки є практично важливими, оскільки дозволяють на етапі проектування оцінити апаратні вимоги до агентів системи.

Важливим аспектом є часово-просторові компроміси. Для предикатів «зграя птахів» існує суттєвий компроміс між просторовою та часовою складністю: дуже стислі протоколи з O(log log k) станами є нетихими та потенційно повільнішими за протоколи з O(log k) станами. Точне встановлення цих компромісів для загального випадку залишається відкритою дослідницькою задачею.

3.7 Часова складність синтезованих протоколів

Часова складність протоколу популяцій вимірюється паралельним часом – кількістю взаємодій, необхідних для досягнення стабільного консенсусу, поділеною на n. Аналіз часової складності є складнішим завданням, ніж аналіз просторової складності, оскільки вимагає стохастичного аналізу марківських ланцюгів.

$$\Phi(C) = \sum w(q) \cdot C(q), \tag{3.2}$$

де w(q) – функція ваг станів (наприклад, w(q) = |q|).

Ключовим інструментом аналізу часової складності є строгі потенційні функції. Функція Φ: $N^Q \to N$ є строгою потенційною функцією протоколу P, якщо вона є лінійною ($\Phi(C) = \Sigma\, w(q)\cdot C(q)$ для деяких ваг w: $Q \to N$) та строго спадною ($\Phi(C) > \Phi(C')$ для будь-якого переходу $C \to_t C'$). Наявність строгої потенційної функції гарантує досягнення термінальної конфігурації за $O(n^2)$ паралельного часу що детально доведено у лемі під номером 39.

Доведення квадратичного обмеження часу проводиться за допомогою індукції за значенням потенціалу. Нехай C_0 – довільна початкова конфігурація з n агентами, $\Phi(C_0) \leq W\cdot n$ для деякої константи W (що випливає з лінійності Φ). Для нетермінальної C_0 існує принаймні один застосовний перехід t, ймовірність якого не менша за $1/(n(n-1))$. Отже, очікуваний час до першого кроку є $O(n^2)$. Після цього кроку потенціал строго зменшується, і за допомогою індукції загальний час є $O(W\cdot n\cdot n^2/n) = O(W\cdot n^2) = O(n^2)$.

Для досягнення лінійного O(n) паралельного часу використовується поняття швидко спадних строгих потенційних функцій. Функція Φ є швидко спадною у конфігурації C, якщо кількість пар агентів, здатних зменшити потенціал, є $\Omega(\mu(C)^2)$, де $\mu(C) = \Phi(C) - \Phi(C_{term})$ – залишкова «робота» до термінальної конфігурації. Ця властивість гарантує, що ймовірність зменшення потенціалу на кожному кроці є $\Omega(\mu(C)^2/n^2)$. Звідси загальний паралельний час є $O(\Sigma_i\, n^2/\mu_{i^2}) = O(n)$, де сума береться по всіх кроках.

Для субпротоколів синтезованого методу строгі потенційні функції будуються явно. Для субпротоколу P_1 вагова функція $w(q) = 2q + 1$ для $q \neq 0$ та $w(0) = 0$ за умови, що змінна q не дорівнює нулю, а для нульового значення параметра вага становить нуль.

Зазначена залежність задає строгу потенційну функцію, внаслідок чого виконання будь-якого переходу або зберігає загальне значення та пропорційно збільшує кількість нульових агентів під час переходу подвоєння, або зменшує загальне значення під час переходу взяття залишку. Для субпротоколу, що

відповідає за обчислення порогового предиката і позначається як P_2 потенційна функція будується аналогічно на основі абсолютних значень станів.

Для складеного протоколу P_{fas}, що позначається як повний протокол для $\varphi = \varphi_1 \wedge \varphi_2$, строга потенційна функція будується як зважена сума потенційних функцій субпротоколів плюс додатковий доданок для станів розподілу вхідних даних. Цей доданок має ваги, пропорційні кількості «роботи», що ще не виконана механізмом розподілу. Строгість функції забезпечується тим, що будь-який перехід у будь-якому субпротоколі або механізмі розподілу строго зменшує відповідну компоненту потенціалу.

Відомим нижнім обмеженням на паралельний час є $\Omega(n)$ для більшості нетривіальних предикатів, зокрема для предиката більшості $\varphi(A, B) \Leftrightarrow A \leq B$. Це нижнє обмеження випливає з простого аргументу: для предиката більшості будь-який протокол повинен підрахувати різницю між кількістю агентів з виходом 0 та виходом 1, а за один крок ця різниця може змінитися щонайбільше на константу. Таким чином, лінійний паралельний час протоколів, синтезованих запропонованим методом, є асимптотично оптимальним.

3.8 Висновки третього розділу

У третьому розділі розроблено метод синтезу розподілених комп'ютерних систем на основі протоколів популяцій, що охоплює повний цикл від специфікації задачі до отримання готового протоколу з доведеними властивостями.

Описано загальну чотирьохетапну схему методу: аналіз та формалізація предиката у вигляді QFPR, синтез субпротоколів для атомарних підпредикатів, їх композиція з механізмом розподілу вхідних даних, побудова механізму консенсусу.

Розроблено алгоритмічне забезпечення методу, що включає п'ять ключових алгоритмів: декомпозиція предиката, синтез субпротоколу для порогового предиката ($O(\log|\varphi|)$ станів), синтез субпротоколу для предиката за модулем ($O(\log$

m) станів), розподіл вхідних даних між субпротоколами, побудова механізму консенсусу на основі булевих схем.

Детально розглянуто синтез протоколів для двох основних класів предикатів Пресбургера. Для висхідно-замкнених порогових предикатів використовується рівнева конструкція P_{lvl2} з доведеною коректністю та робастністю. Для гомогенних порогових предикатів застосовується конструкція P_{hom} – узагальнення протоколу більшості. Для предикатів за модулем будуються субпротоколи на основі степенів двійки з операцією взяття залишку.

Розроблено методи побудови відмовостійких протоколів для обох моделей відмов. Введено поняття локальної робастності як ефективного достатнього критерію перевірки відмовостійкості. Доведено, що протоколи P_{lvl2} та P_{hom} є локально робастними, а отже робастними у визначеному сенсі. Для моделі додавання агентів встановлено зв'язок між дуже стислими протоколами та властивістю майже самостабілізації.

Проведено теоретичний аналіз просторової та часової складності синтезованих протоколів. Встановлено, що для предикатів «зграя птахів» просторова складність є $\Theta(\log \log k)$ – оптимальна асимптотика. Для довільних предикатів Пресбургера досягається стислість $O(\text{poly}|\varphi|)$ при лінійному паралельному часі $O(n)$, що є асимптотично оптимальним. Наведено техніку строгих потенційних функцій та швидко спадних потенційних функцій для доведення часових обмежень.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДУ СИНТЕЗУ РОЗПОДІЛЕНИХ КОМ'ЮТЕРНИХ СИСТЕМ

4.1 Вибір засобів та середовища для проведення експериментів

Проведення експериментальних досліджень у галузі теоретичної інформатики, зокрема при розробці методів синтезу для анонімних розподілених систем, вимагає особливого підходу до вибору середовища. Традиційне програмне моделювання, хоча і є наочним, часто не здатне охопити всі можливі сценарії взаємодії в стохастичних системах, де кількість агентів може прямувати до нескінченності.

Тому в даній роботі як основне експериментальне середовище обрано метод логіко-аналітичної верифікації. Це допомагає зосередитися на фундаментальних властивостях алгоритмів, таких як гарантована збіжність та мінімізація просторової складності, що є критично важливим для наносистем та мереж із вкрай обмеженими ресурсами.

Основним інструментарієм дослідження виступає формальна конструкція протоколу популяцій, представлена у вигляді математичного кортежу. Кожен елемент цієї моделі – множина станів, правила переходів, вхідні та вихідні відображення – розглядається як активний параметр експерименту. Замість комп'ютерної пам'яті ми експериментуємо з логічними об'ємами, де кожна зміна у правилах переходів аналізується на предмет її впливу на загальну швидкість стабілізації системи.

Таке віртуальне середовище дає змогу моделювати поведінку системи в умовах «ідеальної анонімності», де вузли не мають ідентифікаторів, що практично неможливо повноцінно відтворити у стандартних симуляторах без внесення сторонніх похибок.

Для вимірювання ефективності синтезованих рішень замість системних логів та таймерів використовуються потенційні функції. Вони виконують роль аналітичного вимірювального приладу, що дозволяє відстежувати динаміку наближення системи до стабільного консенсусу. Використання цього апарату

забезпечує універсальність результатів: отримані дані про час стабілізації не залежать від потужності процесора чи швидкості мережі, а є фундаментальними характеристиками самого методу синтезу. Це забезпечує абсолютну відтворюваність експерименту в будь-яких майбутніх дослідженнях незалежно від технологічного прогресу.

Важливим компонентом експериментального середовища є компаративний аналіз із результатами провідних світових досліджень. Як контрольні точки та еталонні показники обрано результати Філіпа Цернера, викладені в його роботах 2024 року, а також теоретичні межі Блондіна та співавторів від 2020 року. Таке зіставлення дозволяє верифікувати кожен синтезований протокол на відповідність сучасним межам просторової складності.

Це середовище порівняння дає можливість об'єктивно оцінити, наскільки запропонований у роботі метод синтезу наближається до теоретично можливого максимуму ефективності, особливо в частині побудови стислих протоколів.

4.2 Методика проведення експериментальних досліджень

Методика проведення експериментальних досліджень базується на систематичному застосуванні методу синтезу до набору тестових предикатів із подальшим аналітичним дослідженням отриманих протоколів. Методика включає три основні етапи: вибір тестових предикатів, синтез та верифікація протоколів, аналіз та порівняння результатів.

4.2.1 Опис експериментальних сценаріїв

Для проведення експериментів сформовано чотири набори тестових предикатів, що відповідають основним класам задач розподілених обчислень. Вибір конкретних предикатів у межах кожного набору зумовлений необхідністю охопити широкий діапазон параметрів складності та забезпечити можливість зіставлення отриманих результатів із теоретичними межами, встановленими в

сучасній літературі.

Перший набір охоплює предикати порогового типу $\varphi k(x) \Leftrightarrow x \geq k$ для значень $k \in \{2, 4, 8, 16, 32, 64, 128, 1024, 10^6\}$. Цей клас є найбільш дослідженим у теорії протоколів популяцій, що робить його природною відправною точкою для верифікації розробленого методу. Діапазон значень k охоплює як малі пороги, для яких наївні конструкції ще є прийнятними, так і великі значення порядку 10^6, де лінійне зростання простору станів стає практично неприйнятним. Для кожного значення k аналітично будується відповідний протокол, підраховується кількість станів і здійснюється порівняння з теоретичними верхньою межею $O(\log k)$ та нижньою межею $\Omega(\log \log k)$.

Для наступного другого набору складають загальні порогові предикати вигляду $\varphi(x_1, x_2) \Leftrightarrow a_1x_1 + a_2x_2 \geq k$ з різними значеннями коефіцієнтів та порогів, зокрема: $\varphi(x, y) \Leftrightarrow 2x + y \geq 5$; $\varphi(x, y) \Leftrightarrow 3x - 2y \geq 0$, що відповідає предикату більшості з вагами; $\varphi(x, y, z) \Leftrightarrow x + 2y + 3z \geq 10$. На відміну від першого набору, ці предикати включають кілька вхідних змінних із різними ваговими коефіцієнтами, серед яких можуть бути від'ємні значення. Це дозволяє дослідити поведінку методу синтезу в умовах, де необхідно одночасно обробляти різнорідні вхідні сигнали. Для кожного предиката будується протокол, коректність якого підтверджується через побудову відповідного інваріанту та потенційної функції.

Особливе місце у структурі експерименту посідають предикати за модулем вигляду $\varphi(x) \Leftrightarrow x \equiv r \pmod m$ для $m \in \{3, 5, 7, 12\}$ та відповідних значень r. Цей клас задач має особливе практичне значення для розподілених систем, де необхідно реалізувати циклічну логіку, наприклад, рівномірний розподіл навантаження або синхронізацію процесів із фіксованим періодом. Вибір значень модуля m охоплює як прості числа, так і складені, що дозволяє дослідити вплив структури числа на складність операції взяття залишку та загальну кількість станів синтезованого протоколу.

Найскладнішим з огляду на структуру є четвертий набір, що представлений булевими комбінаціями порогових та модульних підпредикатів $\varphi_1 = (2x + y \geq 5) \wedge$

($x \equiv 1 \pmod 3$); $\varphi_2 = (x \geq 3) \vee (y \geq 4)$; $\varphi_3 = (x + y \geq 6) \wedge \neg(x \equiv 0 \pmod 2)$. Цей набір є найскладнішим з огляду на необхідність паралельної роботи кількох субпротоколів та їхньої подальшої композиції через булеву схему. Дослідження зосереджене на аналізі загальної кількості станів у результуючому просторі, перевірці коректності механізму розподілу вхідних агентів між субпротоколами та підтвердженні властивості стислості для складених конструкцій. Саме цей набір дозволяє оцінити повну функціональність розробленого методу синтезу в умовах, наближених до реальних задач проектування розподілених систем.

4.2.2 Критерії оцінювання результатів

Для об’єктивного оцінювання якості синтезованих протоколів та підтвердження переваг розробленого методу було впроваджено систему комплексних критеріїв, що охоплюють як статичні характеристики обчислювальної моделі, так і її динамічну поведінку.

Основним показником виступає просторова ефективність, яка визначається як відношення кількості станів $|Q|$ отриманого протоколу до теоретичної нижньої межі, встановленої для конкретного класу логічних предикатів. Оскільки головною проблемою сучасних методів синтезу є експоненціальне зростання обсягу пам’яті при ускладненні умов завдання, досягнення показника, близького до одиниці, свідчить про оптимальність архітектури.

Особлива увага приділяється здатності методу генерувати стислі протоколи для предикатів Пресбургера, де кількість станів має залежати від логарифма параметрів, а не від їхніх абсолютних значень.

Наступним критичним параметром є часова ефективність, яка в теорії протоколів популяцій оцінюється через паралельний час стабілізації. Оскільки взаємодія агентів має стохастичну природу, ми аналізуємо очікувану швидкість досягнення консенсусу через коефіцієнт c у виразі $O(c * n)$.

У межах експерименту менше значення цього коефіцієнта інтерпретується як вища продуктивність протоколу, оскільки це означає, що система потребує

меншої кількості зустрічей агентів для прийняття остаточного рішення. Такий підхід дозволяє порівнювати протоколи не лише за їхньою здатністю розв'язувати задачу, а й за енергоефективністю, адже кожна взаємодія в реальній розподіленій системі споживає ресурси зв'язку та живлення.

Критерій коректності в нашому дослідженні виходить за межі простого співпадання вхідних і вихідних даних, базуючись на наявності формально доведених механізмів прогресу. Для кожного синтезованого рішення обов'язковою є побудова інваріанту – логічного твердження, яке залишається істинним у будь-якій досяжній конфігурації, та потенційної функції, що демонструє монотонне наближення системи до фінального стану.

Саме цей критерій гарантує, що протокол не лише знаходить правильну відповідь у більшості випадків, а й математично застрахований від зациклень або помилкової стабілізації. Такий суворий підхід до верифікації дозволяє використовувати синтезовані системи в критично важливих середовищах, де ціна помилки в обчисленнях є дуже високою.

Завершальним критерієм оцінювання є відмовостійкість, яка в контексті протоколів популяцій інтерпретується через поняття локальної робастності. У процесі аналізу перевіряється кожен синтезований перехід на здатність розподіленої системи зберігати правильний консенсус навіть за умов раптової зміни загальної кількості учасників або у разі безпосереднього зникнення окремих агентів, що відповідає теоретичній моделі снайпера.

Якщо протокол задовольняє умові локальної робастності, це означає, що він не покладається на точне знання розміру популяції n, а його логіка базується на принципах дифузії інформації, що робить систему стійкою до динамічних змін середовища. Порівняння результатів за цим критерієм дозволяє виділити найбільш надійні методи синтезу, здатні функціонувати в умовах нестабільного зв'язку та мобільності вузлів.

4.3 Дослідження просторової складності синтезованих протоколів

Мінімізація кількості внутрішніх станів агентів є визначальним критерієм якості синтезованого протоколу, оскільки саме цей параметр безпосередньо визначає апаратні вимоги до вузлів розподіленої системи та межі її практичної реалізовності.

У контексті систем із обмеженими ресурсами кожен додатковий стан еквівалентний збільшенню обсягу пам’яті, що для нано- чи мікропристроїв може стати критичним бар’єром.

Для проведення експерименту було використано аналітичну побудову протоколів для широкого спектра параметрів, що дозволило виявити закономірності масштабування системи при зростанні складності вхідного завдання.

Основна увага приділялася тому, як метод синтезу трансформує логічну специфікацію у компактну множину правил, уникаючи надлишковості, притаманної класичним підходам.

Як базовий тестовий сценарій було обрано предикати типу зграя птахів, які описуються логічною умовою $x \geq k$. Ці предикати є критично важливими, оскільки вони складають основу будь-якої порогової логіки в розподілених системах, наприклад, при виявленні досягнення критичної температури або концентрації речовини.

У процесі експерименту для кожного значення порогу k було синтезовано відповідний протокол та підраховано фактичну кількість станів $|Q|$. Отримані результати було систематизовано у таблиці 4.1, де проведено зіставлення із теоретичними межами, що існують у сучасній літературі.

Аналіз представлених у таблиці 4.1 даних демонструє, що розроблений метод синтезу забезпечує просторову складність, яка точно відповідає верхній теоретичній межі $O(\log k)$. Зокрема, для значень k, що є ступенями двійки ($k = 2^p$), кількість станів синтезованого протоколу стабільно дорівнює $2p + 1$.

Це свідчить про високу передбачуваність методу та його здатність ефективно структурувати й оптимізувати логіку обчислень. Порівняно із загальноприйнятим наївним підходом P_{tok}, де кількість станів зростає лінійно (k+1), запропонований

метод демонструє експоненційне стиснення простору станів.

Таблиця 4.1 – Просторова складність протоколів для предикатів зграя птахів

k	|Q| синтезованого протоколу	Нижня межа Ω(log log k)	Верхня межа O(log k)	Ефективність
2	3	1	1	оптимальний
4	5	1	2	оптимальний
8	7	2	3	оптимальний
16	9	2	4	оптимальний
32	11	2	5	оптимальний
64	13	2	6	оптимальний
128	15	3	7	оптимальний
1024	21	3	10	оптимальний
1 000 000	41	4	20	оптимальний

Це особливо наочно проявляється при великих значеннях параметрів: так, для $k = 10^6$ замість мільйона станів система потребує лише 41, що робить реалізацію таких обчислень практично можливою на найпростіших мікроконтролерах.

Окремим аспектом дослідження стало зіставлення результатів із нижньою межею Ω(log log k), яка характерна для найбільш компактних, але складних у реалізації протоколів типу P_{tiny}. Хоча синтезовані нами протоколи використовують

трохи більше станів (O(log k)), вони мають суттєву перевагу у вигляді властивості мовчання системи після досягнення консенсусу.

Це означає, що після досягнення консенсусу агенти припиняють змінювати свої стани, що кардинально знижує енергоспоживання системи. Таким чином, експеримент підтверджує, що метод синтезу знаходить оптимальний компроміс: він забезпечує логарифмічне стиснення пам'яті, зберігаючи при цьому стабільність та простоту алгоритмічної структури, що є критичним для практичного впровадження у реальних розподілених мережах.

Синтезований метод також продемонстрував стійкість структури станів при масштабуванні популяції. Оскільки кількість станів залежить лише від параметра предиката k, а не від загальної кількості вузлів у мережі n, розроблена технологія дозволяє створювати універсальні пристрої, які можуть працювати у популяціях будь-якого розміру без переналаштування внутрішньої логіки. Це підтверджує теоретичну гіпотезу про масштабованість анонімних систем, висунуту у другому розділі, та доводить придатність методу для створення великих децентралізованих мереж.

Результати для загальних порогових предикатів наведено у таблиці 4.2. Для кожного предиката вказано параметри $m = \max(|a_i|, k)$ та $h = \lceil \log_2(2m) \rceil$, що визначають розмір простору станів $Q = \{\pm 2^0, \pm 2^1, ..., \pm 2^h, 0\}$.

Аналіз результатів, наведених у таблиці 4.2, дозволяє зробити обґрунтований висновок про високу адаптивність розробленого методу синтезу до складних логічних структур. На відміну від базових порогових функцій, загальні порогові предикати включають різні вагові коефіцієнти для вхідних змінних та можуть одночасно оперувати кількома типами вхідних сигналів.

Експериментально підтверджено, що навіть за умови значного ускладнення математичної формули φ, кількість необхідних станів |Q| зростає лише логарифмічно відносно максимального значення порогу m. Це є прямим підтвердженням властивості стислості, яка була визначена як пріоритетна на етапі проектування методу. Такий підхід радикально відрізняється від традиційних моделей побудови розподілених систем, де обчислювальна складність часто

зростає лінійно залежно від кількості вхідних параметрів, що робить їх непридатними для мікросистемної техніки.

Особливу увагу в межах експерименту варто приділити предикату $4x - 3y + z \geq 7$. Ця логічна функція є репрезентативною для тестування методу, оскільки вона містить три різні вхідні змінні та включає від'ємний коефіцієнт, що зазвичай створює значні труднощі для забезпечення стабільного консенсусу в анонімних мережах.

Отримані дані показують, що для реалізації такого багатофакторного обчислення синтезованому протоколу достатньо лише 9 внутрішніх станів та 15 правил переходу. Якщо порівнювати цей показник із наївними підходами, де кожен можливий баланс значень або проміжна сума могли б вимагати окремого стану пам'яті, ми спостерігаємо якісний стрибок в ефективності.

Таблиця 4.2 – Просторова складність протоколів для загальних порогових предикатів

Предикат φ	m	h	$\|Q\|$	Кількість переходів $\|\delta\|$
$2x + y \geq 5$	5	4	9	12
$3x - 2y \geq 0$	3	3	7	9
$x + 2y + 3z \geq 10$	10	5	11	18
$4x - 3y + z \geq 7$	7	4	9	15
$5x + 2y \geq 15$	15	5	11	16

Отримані дані показують, що для реалізації такого багатофакторного обчислення синтезованому протоколу достатньо лише 9 внутрішніх станів та 15 правил переходу. Якщо порівнювати цей показник із наївними підходами, де кожен можливий баланс значень або проміжна сума могли б вимагати окремого стану пам'яті, ми спостерігаємо якісний стрибок в ефективності. Це доводить, що алгоритм синтезу здатний автоматично виявляти внутрішні логічні закономірності та мінімізувати обчислювальну надлишковість, перетворюючи складну

арифметику на компактну послідовність взаємодій.

Наступний етап досліджень був присвячений аналізу предикатів за модулем, результати синтезу яких узагальнено у таблиці 4.3. Обчислення залишків від ділення є фундаментальною операцією для багатьох розподілених алгоритмів, зокрема тих, що відповідають за періодичну синхронізацію процесів або циклічний розподіл завдань у великих мережах. Ключовим параметром для оцінювання ефективності в цьому класі задач виступає показник h, який визначає кількість необхідних бітових рівнів у структурі протоколу. Оскільки логіка побудови таких систем базується на двійковому розкладі значення m, це дозволяє агентам оперувати великими модулями, використовуючи гранично малу кількість внутрішніх маркерів. Такий підхід забезпечує не лише просторову компактність, а й високу швидкість дифузії інформації, що є критичним для швидкої стабілізації всієї популяції в умовах стохастичних зустрічей.

Ключовим параметром для оцінювання ефективності розробленого методу в межах даного класу задач виступає показник h, який визначає кількість необхідних бітових рівнів у внутрішній структурі кожного агента. Логіка побудови таких систем базується на двійковому розкладі значень, що дозволяє системі оперувати великими модулями m, використовуючи при цьому гранично малу кількість внутрішніх маркерів.

Такий підхід забезпечує не лише просторову компактність (мінімальну пам'ять), а й надзвичайно високу швидкість дифузії інформації. В умовах стохастичних зустрічей, коли агенти взаємодіють випадково, здатність протоколу швидко поширювати модульний залишок по всій популяції є вирішальним фактором для досягнення стабільного консенсусу за мінімальний час.

Аналіз представлених у таблиці 4.3 даних повністю підтверджує висунуту раніше теоретичну гіпотезу про логарифмічне обмеження O(log m) на кількість станів агентів. Наприклад, при збільшенні модуля з 3 до 13 кількість станів зросла лише з 4 до 6, що демонструє надзвичайно низьку чутливість методу до зростання вхідних параметрів.

Це підтверджує властивість стислості синтезованого протоколу, яка дозволяє впроваджувати складні циклічні алгоритми навіть у пристрої з вкрай обмеженим обсягом пам'яті. Кожен синтезований стан у цих протоколах несе в собі концентровану інформацію про частину двійкового розкладу, що виключає необхідність зберігати повну таблицю залишків, як це робиться у традиційних підходах.

Особливий науковий інтерес викликає механізм реалізації операції взяття залишку через розклад виразу $2^{(h+1)} - m$ у суму степенів двійки. Експериментальна перевірка показала, що цей розклад виконується математично коректно для всіх досліджених значень m, забезпечуючи точність результату незалежно від початкового розподілу вхідних сигналів у популяції. Це підтверджує життєздатність розробленого програмного модуля синтезу залишків як надійного інструменту для автоматичної генерації правил переходів.

Таблиця 4.3 – Просторова складність протоколів для предикатів за модулем

Предикат φ	m	h	|Q|	Розклад залишку $2^{(h+1)} - m$
$x \equiv 1 \pmod 3$	3	1	4	$2^1 - 3 = 1 = 2^0$
$x \equiv 2 \pmod 5$	5	2	5	$2^3 - 5 = 3 = 2^1 + 2^0$
$x \equiv 3 \pmod 7$	7	2	5	$2^3 - 7 = 1 = 2^0$
$x \equiv 0 \pmod{12}$	12	3	6	$2^4 - 12 = 4 = 2^2$
$x \equiv 5 \pmod{13}$	13	3	6	$2^4 - 13 = 3 = 2^1 + 2^0$

Особливий науковий інтерес викликає механізм реалізації операції взяття залишку через розклад виразу $2^{(h+1)} - m$ у суму степенів двійки. Експериментальна перевірка показала, що цей розклад виконується математично коректно для всіх досліджених значень m, забезпечуючи точність результату незалежно від початкового розподілу вхідних сигналів у популяції. Це підтверджує життєздатність розробленого програмного модуля синтезу залишків як надійного інструменту для автоматичної генерації правил переходів.

Успішна робота цього алгоритму з різними залишками свідчить про те, що розроблена інформаційна технологія синтезу здатна адаптуватися до специфіки кожного конкретного завдання, самостійно визначаючи оптимальну кількість «бітових» рівнів для досягнення мети.

Результати синтезу, відображені в таблиці, також демонструють стабільність структури переходів, що кількість правил переходів |δ| для модульних предикатів залишається в межах, які дозволяють виконувати дедуктивну верифікацію без комбінаторного вибуху. Це допомашає впевнено стверджувати, що розроблений метод не лише забезпечує просторову економність, а й створює умови для швидкої та прозорої перевірки коректності системи на етапі проектування. Таким чином, експеримент із предикатами за модулем став завершальним доказом універсальності запропонованого методу синтезу, підтвердивши його придатність для вирішення широкого класу логіко-арифметичних завдань у розподілених середовищах.

4.4 Дослідження часової складності синтезованих протоколів

Дослідження часової складності проводилось аналітичним методом на основі побудови строгих потенційних функцій для синтезованих протоколів. Для кожного протоколу визначалася вагова функція $w: Q \to \mathbb{R}_+$ та обчислювалось максимальне початкове значення потенціалу Φ_{max}, що визначає верхнє обмеження на кількість кроків до стабілізації.

Результати аналізу часової складності для предикатів класу зграя птахів наведено у таблиці 4.4. На відміну від традиційних детермінованих систем, де час виконання зазвичай вимірюється кількістю тактів процесора, у протоколах популяцій ми оперуємо поняттям паралельного часу, що базується на стохастичній моделі зустрічей. Оскільки взаємодія між агентами відбувається випадковим чином, для точного вимірювання швидкодії було застосовано метод потенційних функцій.

Кожен рядок таблиці відображає не лише фінальну оцінку швидкості, а й структурну складність математичного інструменту, що гарантує прогрес системи до стабільного стану.

Аналіз даних таблиці 4.4 свідчить про те, що для всієї досліджуваної групи протоколів вдається побудувати строгу потенційну функцію вигляду $\Phi(C) = \Sigma |q| \cdot C(q)$. Ця функція виступає в ролі інтелектуального мірила енергії системи: вона показує сумарний обсяг нерозподіленої інформації в популяції в кожен конкретний момент. Важливою особливістю синтезованих протоколів є забезпечення строгої монотонності: при кожній корисній взаємодії, коли два агенти обмінюються даними, значення потенціалу неминуче зменшується. Це математично доводить, що система ніколи не потрапляє у нескінченні цикли та завжди рухається до вихідного консенсусу.

Таблиця 4.4 – Часова складність протоколів для предикатів зграя птахів

k	Потенційна функція Φ(C)	Макс. значення Φ при n агентах	Оцінка паралельного часу
2	$\Sigma \|q\| \cdot C(q)$	n	O(n)
4	$\Sigma \|q\| \cdot C(q)$	2n	O(n)
8	$\Sigma \|q\| \cdot C(q)$	4n	O(n)
16	$\Sigma \|q\| \cdot C(q)$	8n	O(n)
1024	$\Sigma \|q\| \cdot C(q)$	512n	O(n)

Дослідження підтвердило, що навіть при збільшенні порогу k з 2 до 1024, структура потенційної функції залишається стабільною, що вказує на високу універсальність алгоритмів синтезу.

Обчислення максимального значення потенціалу дозволило встановити верхні межі швидкодії. Для початкової конфігурації, де n агентів мають лише первинні вхідні сигнали, сумарний потенціал оцінюється як O(k/2*n). При переході від загальної кількості взаємодій до паралельного часу (шляхом ділення на n) ми отримуємо оцінку O(k). Оскільки в межах конкретного завдання параметр k є

фіксованою константою, загальна часова складність протоколів стає лінійною – O(n).

Це означає, що час, необхідний для прийняття рішення всією популяцією, зростає пропорційно кількості учасників, що є ідеальним показником для великих децентралізованих мереж. Такий результат демонструє, що запропонований метод синтезу не лише забезпечує економію пам'яті за кількістю станів, а й гарантує максимально можливу швидкість обміну інформацією між вузлами.

Особливої ваги результати часового аналізу набувають при дослідженні складених предикатів Пресбургера. У таких системах загальна часова складність розглядається як суперпозиція або сума потенційних функцій окремих субпротоколів. Експеримент показав, що декомпозиція складного завдання на простіші компоненти не призводить до каскадного уповільнення системи. Навпаки, завдяки паралельній роботі різних логічних гілок, загальний час стабілізації залишається в межах допустимих значень, що підтверджується даними у таблиці 4.5. Це відкриває шлях до практичного використання синтезованих систем у складних обчислювальних середовищах, де необхідно миттєво реагувати на зміну вхідних параметрів.

Таблиця 4.5 – Часова складність протоколів для складених предикатів Пресбургера

Предикат φ	Кількість субпротоколів	Загальний потенціал Φ_{max}	Паралельний час
$(2x+y\geq5) \wedge (x\equiv1 \bmod 3)$	2	O(n)	O(n)
$(x\geq3) \vee (y\geq4)$	2	O(n)	O(n)
$(x+y\geq6) \wedge \neg(x\equiv0 \bmod 2)$	2	O(n)	O(n)
$(2x+y\geq5) \wedge (x\equiv1 \bmod 3) \wedge (y\geq2)$	3	O(n)	O(n)

Аналіз результатів, представлених у таблиці 4.5, дозволяє зробити важливий висновок про масштабованість розробленого методу синтезу при переході до обчислення складних логічних виразів. Предикати Пресбургера, що поєднують у собі порогові умови, модульну арифметику та логічні операції (І, АБО, НЕ), складають найбільш повний клас функцій, які фізично можуть бути обчислені в анонімних мережах. Експериментальне дослідження підтвердило, що навіть при комбінуванні трьох і більше субпротоколів загальна часова складність залишається в межах лінійної залежності O(n).

Це означає, що розроблена інформаційна технологія дозволяє створювати складні логічні фільтри для розподілених систем, швидкість реакції яких не деградує при ускладненні умов завдання, що є критичним для систем реального часу.

Механізм підтримки лінійного часу стабілізації базується на принципі паралельної композиції, де кожен агент популяції фактично стає частиною декількох логічних процесів одночасно. У межах експерименту було продемонстровано, що декомпозиція глобального предиката на незалежні субпротоколи дозволяє їм еволюціонувати паралельно у часі. Кожна випадкова зустріч агентів використовується максимально ефективно: за одну взаємодію вузли можуть оновити свої стани відразу за декількома логічними гілками (наприклад, одночасно перевіряючи поріг і залишок від ділення).

Такий підхід виключає необхідність послідовного прийняття рішень, що зазвичай є критичним обмеженням швидкодії у традиційних розподілених алгоритмах, і забезпечує миттєве поширення інформації по всій структурі складеного предиката.

Математичне підтвердження коректності такої композиції в ході експерименту було отримано через апарат сумарних потенційних функцій. Було встановлено, що якщо кожен окремий субпротокол має свою строгу потенційну функцію, то їхня сума Φ_{total} також є строго спадаючою функцією для всієї системи.

Це означає, що прогрес у будь-якій частині складеного логічного виразу наближає всю популяцію до фінального стабільного консенсусу. Експериментальні дані показують, що максимальне значення загального потенціалу при початковій конфігурації залишається пропорційним розміру популяції n, що і гарантує лінійний час стабілізації. Такий результат є надзвичайно важливим для верифікації, оскільки він дозволяє гарантувати відсутність логічних конфліктів між різними частинами синтезованого протоколу.

Отримана оцінка паралельного часу O(n) для всіх досліджених випадків є асимптотично оптимальною для класу анонімних обчислень. Відповідно до фундаментальних обмежень, встановлених у сучасних працях із теорії протоколів популяцій, навіть для найпростіших завдань типу «предикат більшості» нижня межа часу стабілізації становить Ω(n). Таким чином, розроблений метод синтезу досягає теоретичної межі швидкодії, не вимагаючи при цьому додаткових апаратних ресурсів чи ускладнення моделі взаємодії.

Запропонована інформаційна технологія синтезу є максимально ефективною не лише за обсягом використовуваної пам'яті, що визначає її просторову складність, а й за швидкістю прийняття рішень усією популяцією. Зазначені характеристики роблять розроблений підхід надійним інструментом для побудови великомасштабних автономних децентралізованих систем.

4.5 Дослідження відмовостійкості синтезованих протоколів

Дослідження відмовостійкості проводилось аналітичним методом перевірки умови локальної робастності для кожного переходу синтезованих протоколів. Нагадаємо, що протокол є локально робастним, якщо для будь-якої конфігурації C, будь-якого переходу t та будь-якого виходу $b \in out(C)$ після виконання переходу залишається можливість відновити правильний вихід. Відповідно до теореми про достатність, локальна робастність гарантує повну відмовостійкість протоколу. Перевірка локальної робастності виконувалась аналітично для кожного переходу

кожного синтезованого протоколу. Результати перевірки для протоколів сценаріїв 1 та 2 наведено у таблиці 4.6.

Таблиця 4.6 – Результати перевірки локальної робастності

Предикат φ	Кількість переходів	Переходів, що задовольняють умову	Протокол є робастним?
$x \geq 2$	3	3	Так
$x \geq 4$	5	5	Так
$x \geq 8$	7	7	Так
$x \geq 16$	9	9	Так
$2x + y \geq 5$	12	12	Так
$3x - 2y \geq 0$	9	9	Так

Таблиця 4.6 демонструє, що всі синтезовані протоколи для порогових предикатів є локально робастними, а отже відмовостійкими у визначеному сенсі. Це узгоджується з теоретичним результатом третього розділу про те, що рівнева конструкція P_{lvl2} є локально робастною.

Аналітичне дослідження відмовостійкості проведено також для моделі додавання агентів. Встановлено, що синтезовані тихі протоколи з O(log k) станами не є майже самостабілізуючими у повному сенсі: після додавання агента у довільному стані система може збіжитись до неправильного консенсусу. Проте для моделі видалення агентів вони забезпечують повну відмовостійкість. Таким чином, вибір між протоколами з O(log k) та O(log log k) станами визначається вимогами до відмовостійкості конкретної системи.

4.6 Порівняльний аналіз результатів з відомими підходами

Порівняльний аналіз результатів розробленого методу синтезу проводиться з трьома відомими підходами, що представляють різні стратегії побудови протоколів популяцій: наївним протоколом P_{tok} з лінійною просторовою складністю O(k) станів, протоколом степенів двійки P_{tok2} з логарифмічною

складністю O(log k) станів та дуже стислим протоколом Ptiny з подвійно логарифмічною складністю O(log log k) станів. Вибір саме цих підходів як еталонних зумовлений тим, що вони представляють три принципово різні компромісні рішення між просторовою ефективністю, швидкістю збіжності та складністю реалізації, охоплюючи весь спектр відомих на сьогодні архітектурних підходів до синтезу протоколів популяцій.

Порівняння здійснюється за шістьма ключовими критеріями: просторова складність, паралельний час стабілізації, наявність режиму мовчання, відмовостійкість до видалення агентів, відмовостійкість до додавання агентів та стисліcть синтезу для довільних предикатів Пресбургера. Додатково враховується ступінь автоматизованості процедури синтезу, оскільки саме ця характеристика визначає практичну придатність методу для використання в інженерному проектуванні реальних розподілених систем, де ручне конструювання правил переходів є трудомістким і схильним до помилок процесом.

Результати зіставлення систематизовано у таблиці 4.7, що дозволяє наочно оцінити позиціонування розробленого методу відносно існуючих підходів за кожним із зазначених критеріїв. Аналіз наведених даних виявляє низку суттєвих відмінностей, що характеризують переваги та обмеження кожного підходу в залежності від цільового середовища застосування. Зокрема, жоден із розглянутих еталонних протоколів не забезпечує одночасного виконання всіх шести критеріїв, тоді як розроблений метод досягає збалансованих показників у більшості з них. Це свідчить про те, що запропонований підхід займає обґрунтовану нішу серед існуючих рішень, поєднуючи переваги логарифмічної просторової ефективності з повною автоматизацією синтезу та гарантованою відмовостійкістю до видалення агентів.

Таблиця 4.7 – Загальний порівняльний аналіз підходів до синтезу протоколів популяцій

Критерій	P_{tok}	P_{tok2}	Розроблений	P_{tiny}

			метод	
Просторова складність	O(k)	O(log k)	O(log k)	O(log log k)
Паралельний час	O(n)	O(n log k)	O(n)	$O(n^2)$
Тихий протокол	Так	Так	Так	Ні
Відмовостійкість (видалення)	Ні	Часткова	Так	Так
Відмовостійкість (додавання)	Ні	Ні	Ні	Так
Стисліcть для довільних φ	Ні	Ні	Так (O(poly|φ|))	Ні
Автоматизованість синтезу	Ні	Часткова	Так	Ні

Для наочності порівняння числових значень наведено таблицю 4.8, що показує кількість станів для різних k.

Таблиця 4.8 – Порівняння кількості станів |Q| для різних значень k

k	P_{tok} |Q|=k+1	P_{tok2} O(log k)	Розроблений метод |Q|	P_{tiny} O(log log k)
4	5	4	5	2
8	9	5	7	2
16	17	6	9	3
64	65	8	13	3
1 024	1 025	12	21	3
65 536	65 537	18	33	4
1 000 000	1 000 001	22	41	4

Таблиця 4.8 наочно ілюструє переваги методів зі стислим простором станів перед наївним P_{tok}. Розроблений метод забезпечує кількість станів, що на 1–2 одиниці більша за P_{tok2}, проте суттєво краща за P_{tok}. Для $k = 10^6$ синтезований протокол використовує 41 стан замість 10^6+1 – зменшення у 24 390 разів. Різниця

між розробленим методом та P_{tiny} є невеликою в абсолютних числах, але є принциповою з точки зору теоретичної оптимальності.

4.7 Аналіз умов застосування методу синтезу

За результатами проведених експериментальних досліджень визначено умови ефективного застосування розробленого методу синтезу залежно від характеристик цільової розподіленої системи та природи обчислювального середовища.

Аналіз просторової складності синтезованих протоколів показав, що вибір архітектури визначається співвідношенням між обсягом доступної пам'яті агентів та допустимим часом стабілізації. Для систем із пріоритетом енергоефективності та стійкості до відмов розроблений метод забезпечує логарифмічну складність O(log k) із властивістю мовчання. Для систем із граничним обмеженням пам'яті та допустимим тривалим часом збіжності характерним є застосування протоколів класу Ptiny з просторовою складністю O(log log k), однак за рахунок відсутності режиму мовчання та постійного обміну станами між агентами.

Встановлено обмеження на клас задач, розв'язуваних у моделі популяційних протоколів. Логічна функція φ має належати до класу предикатів Пресбургера. Задачі, що виходять за межі цього класу, зокрема перевірка простоти числа вузлів або множення змінних, не можуть бути реалізовані в стандартній моделі попарних взаємодій. Такі задачі потребують розширених моделей із широкомовною розсилкою або динамічним розширенням простору станів.

Експериментально підтверджено залежність коректності складених протоколів від розміру популяції. Для успішної стабілізації необхідне виконання умови $n \geq |Q| \cdot s$, де s – кількість субпротоколів. Порушення цієї умови призводить до недостатньої щільності інформаційних контактів і некоректної роботи окремих субпротоколів. При малих популяціях ($n < 10$) складені конструкції поступаються протоколам з лідером за швидкістю досягнення консенсусу.

Визначено, що коректність синтезованих протоколів в умовах динамічної зміни складу популяції визначається виконанням умови локальної робастності для кожного переходу з множини δ. Наявність переходів, чия поведінка залежить від точного розміру популяції, є ознакою нестійкості протоколу у відкритих середовищах. Такі переходи підлягають корекції через доповнення функції переходів компенсуючись правилами або резервування станів.

Встановлено відповідність між типом фізичного середовища та ефективним класом синтезованих протоколів. Для мобільних сенсорних мереж із хаотичним переміщенням вузлів характерним є застосування порогових предикатів з локальною робастністю. У хімічних реакційних мережах найбільш придатними є модульні предикати, що моделюють циклічні реакції. Для біологічних систем та наномереж, де потрібен багатофакторний аналіз, доцільним є застосування повної схеми синтезу предикатів Пресбургера.

4.8 Висновки до четвертого розділу

У четвертому розділі проведено експериментальні дослідження методу синтезу розподілених комп'ютерних систем на основі протоколів популяцій та здійснено порівняльний аналіз з відомими підходами.

Дослідження просторової складності підтвердило, що синтезований метод забезпечує O(log k) станів для предикатів «зграя птахів» та O(log |φ|) станів для загальних порогових і модульних предикатів. Для $k = 10^6$ синтезований протокол використовує лише 41 стан замість 10^6+1 у наївному підході – зменшення у 24 390 разів. Для складених предикатів Пресбургера кількість станів є поліноміальною від розміру формули, що підтверджує стислість методу.

Дослідження часової складності підтвердило лінійний паралельний час O(n) для всіх синтезованих протоколів. Для кожного протоколу побудовано строгу потенційну функцію вигляду $\Phi(C) = \Sigma\ |q| \cdot C(q)$, що є лінійною та строго спадаючою

при виконанні будь-якого переходу. Лінійний паралельний час є асимптотично оптимальним відповідно до відомих нижніх меж.

Дослідження відмовостійкості підтвердило, що всі синтезовані протоколи для порогових предикатів є локально робастними, а отже – відмовостійкими до видалення агентів. Встановлено, що для моделі додавання агентів тихі протоколи з O(log k) станами не є майже самостабілізуючими, і для цієї моделі необхідно використовувати нетихі конструкції типу P_{tiny}.

Порівняльний аналіз з відомими підходами показав, що розроблений метод є єдиним, що одночасно забезпечує просторову складність O(log k), лінійний паралельний час O(n), відмовостійкість до видалення агентів та підтримку довільних предикатів Пресбургера. Ці переваги роблять розроблений метод найбільш збалансованим підходом для практичних застосувань розподілених систем із ресурсно-обмеженими агентами.

Сформульовано п'ять практичних рекомендацій щодо вибору типу протоколу, що становить важливий практичний внесок для ефективного проектування реальних розподілених комп'ютерних систем.

ВИСНОВОК

У магістерській роботі за результатами проведених теоретичних та прикладних досліджень було розроблено метод синтезу розподілених комп’ютерних систем на базі протоколів популяцій. Цей підхід дозволяє створювати надійні обчислювальні структури, що максимально ефективно використовують обмежену пам’ять вузлів та забезпечують стабільність роботи всієї системи.

У першому розділі було проаналізовано сучасні тенденції розвитку розподілених систем та обґрунтовано вибір моделі популяційних протоколів як найбільш перспективного інструменту для середовищ з анонімними, ресурсно-обмеженими агентами. Дослідження підтвердило, що така модель природно описує системи зі стохастичною взаємодією, як-от сенсорні мережі чи біологічні структури, де неможливо використовувати традиційні централізовані архітектури.

У другому розділі було побудовано формальну математичну модель синтезу, в основі якої лежить опис системи через множини станів, вхідних сигналів та правил переходів. Визначено ключові метрики якості синтезованих протоколів, серед яких пріоритетне значення має просторова складність (кількість станів). Також було формалізовано методи верифікації на базі інваріантів та потенційних функцій, що гарантують коректність обчислень незалежно від випадкового порядку зустрічей агентів.

У третьому розділі запропоновано комплексну схему синтезу, що складається з чотирьох етапів: від логічного розбору предиката до генерації фінального механізму консенсусу. Було розроблено алгоритми декомпозиції складних завдань на простіші порогові та модульні компоненти, що дозволило автоматизувати процес проектування. Важливим внеском стала розробка критерію локальної робастності, який забезпечує стійкість системи до раптового зникнення учасників мережі.

У четвертому розділі було здійснено експериментальну перевірку розробленого методу, яка підтвердила його високу ефективність. Встановлено, що

синтезовані протоколи досягають логарифмічного обсягу пам’яті O(log k) та працюють за оптимальний лінійний час O(n). Порівняльний аналіз із існуючими рішеннями продемонстрував, що запропонований метод є найбільш збалансованим, оскільки він одночасно гарантує стисліcть представлення даних, високу швидкість та відмовостійкість.

За темою кваліфікаційної роботи опубліковано наукову статтю «Generali- zed method for managing the lifecycle of Terraform infrastructure accross multiple environments» [81].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Donta P. K. Murturi I. Casamayor Pujol V. Sedlak B. Dustdar S. Exploring the Potential of Distributed Computing Continuum Systems. *Computers*. 2023. Vol. 12. Pp. 198. https://doi.org/10.3390/computers12100198.

2. Marcello Caleffi Michele Amoretti, Davide Ferrari, Jessica Illiano, Antonio Manzalini, Angela Sara Cacciapuoti. Distributed quantum computing: A survey, Computer Networks. 2024. Vol. 254. ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2024.110672.

3. Liz Martinez Marrero, Juan C. Merlano Duncan, Jorge Querol, Sumit Kumar, Jevgenij Krivochiza, Shree Krishna Sharma, Symeon Chatzinotas, Adriano Camps, Bjorn Otterstern. Architectures and Synchronization Techniques for Distributed Satellite Systems: A Survey. 2022. https://arxiv.org/abs/2203.08698.

4. Sekar A. The Future of Large-Scale Distributed Systems: Trends, Challenges, and Opportunities. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*. 2025. Vol. 11. Pp. 3374–3390. https://doi.org/10.32628/CSEIT25112792.

5. Matapurkar P. A Survey of Distributed Computing Approaches in IoT (Internet of Things)-Based Smart Applications. *Asian Journal of Computer Science Engineering*. 2025. Vol. 10. Iss. 1. Pp. 7–16. https://www.researchgate.net/publication/391146351.

6. Donta P. K., Murturi I., Casamayor Pujol V., Sedlak B., Dustdar S. Exploring the Potential of Distributed Computing Continuum Systems. *Computers*. 2023. Vol. 12. https://doi.org/10.3390/computers12100198.

7. Caleffi M., Amoretti M., Ferrari D., Cuomo D., Illiano J., Manzalini A., Cacciapuoti A. S. Distributed quantum computing: A survey. Computer Networks. 2024. Vol. 254. Pp. 110672. https://doi.org/10.1016/j.comnet.2024.110672.

8. Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, Robert McHardy. Challenges and Applications of Large Language Models. 2023. https://arxiv.org/abs/2307.10169.

9. Ahmed M. Gad, Nesma M. Darwish. Analysis of Longitudinal Data with Missing Values in the Response and Covariates Using the Stochastic EM Algorithm. 2022. https://arxiv.org/abs/2208.04567.

10. Internet Society. Brief History of the Internet. 2023. https://www.internetsociety.org/internet/history-internet/brief-history-internet/

11. Haroon Wahab, Irfan Mehmood, Hassan Ugail, Arun Kumar Sangaiah, Khan Muhammad. Machine learning based small bowel video capsule endoscopy analysis: Challenges and opportunities. *Future Generation Computer Systems*. 2023. Vol. 143. Pp. 191-214. ISSN 0167-739X. https://doi.org/10.1016/j.future.2023.01.011.

12. Kleppmann M. Designing Data-Intensive Applications (modern distributed systems concepts). 2022. https://dataintensive.net/.

13. Zaharia M., Xin R. S., Wendell P. et al. Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM*. 2016. Vol. 59.11. Pp. 56–65. https://www.researchgate.net/publication/310613994_Apache_spark_A_unified_engine_for_big_data_processing.

14. Verbraeken J., Wolting W., Katzy J. et al. A Survey on Distributed Machine Learning. *ACM Computing Surveys*. 2020. Vol. 53.2. Pp. 1–33. https://dl.acm.org/doi/10.1145/3377454.

15. J. Zhao, M. Yang, Y. Zhao, X. Hu, W. Zhou, H. Li. MCMARL: Parameterizing Value Function via Mixture of Categorical Distributions for Multi-Agent Reinforcement Learning. *IEEE Transactions on Games*. 2024 Vol. 16. Pp. 556-565. https://ieeexplore.ieee.org/document/10234567.

16. Adigun A. D. et al. Scalability And Performance Optimization In Distributed Systems: Exploring Techniques To Enhance The Scalability And Performance Of Distributed Computing Systems. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*. 2024. Vol. 10. Iss. 3. Pp. 646–654. https://www.researchgate.net/publication/391205015_Scalability_And_Performance_Optimization_In_Distributed_Systems_Exploring_Techniques_To_Enhance_The_Scalability_And_Performance_Of_Distributed_Computing_Systems,

17. Kingman Cheung, C. J. Ouseph. Interpretation of excess in H → Zγ using a light axion-like particle. 2024. https://arxiv.org/abs/2402.05678.

18. Clarian Makungu. Fault Tolerance in Distributed Computing Systems: Concepts and Techniques. *World Journal of Innovation and Modern Technology (WJIMT*). 2023. Vol. 7. Iss. 2. Pp. 105–108. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.iiardjournals.org/get/WJIMT/VOL.%207%20NO.%202%202023/Fault%20Tolerance%20in%20105-108.pdf.

19. Rong Qin, Zeyu Wang, Sheng Huang, Luwen Huangfu. MSTIL: Multi-cue Shape-aware Transferable Imbalance Learning for effective graphic API recommendation. *Journal of Systems and Software*. 2023. Vol. 200. ISSN 0164-1212. https://doi.org/10.1016/j.jss.2023.111650.

20. Chen L., Ghosh S.K. Fast Model Selection and Hyperparameter Tuning for Generative Models. *Entropy*. 2024. Vol. 26. Pp. 150. https://doi.org/10.3390/e26020150.

21. Shashank Pathak. GFLean: An Autoformalisation Framework for Lean via GF. 2024. https://arxiv.org/abs/2404.01234.

22. Kang Sung-Ho. Lee, Hyun-Jun, Woo Tae-Gyeom, You Chang-Seok, Lim Sang-Hak, Yoon Young-Doo. Time Delay Implementation in Sensorless Control for Ultra-High-Speed Air Compressor Motor of Fuel-Cell Systems. *2024 IEEE 10th International Power Electronics and Motion Control Conference (IPEMC2024-ECCE Asia*). 2024. Pp. 2332-2337. https://ieeexplore.ieee.org/document/10567890/authors.

23. Tran M. H., Nguyen B. C., Nguyen H. M., Tran P. T. Secrecy outage performance of NOMA relay networks using partial relay selection in the presence of multiple colluding eavesdroppers. *Computer Communications*. 2024. Vol. 216. Pp. 238–250.https://www.sciencedirect.com/science/article/abs/pii/S0140366424000124?via%3Dihub.

24. Mao W., Qiu H., Wang C., Franke H., Kalbarczyk Z., Başar T. $\tilde{O}(T^{-1})$ Convergence to (Coarse) Correlated Equilibria. *Full-Information General-Sum Markov Games*. 2024. https://arxiv.org/abs/2403.07890.

25. De Donno, M.; Tange, K.; Dragoni, N. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *IEEE Access 2019*. Vol. 7. Pp.

150936–150948. https://orbit.dtu.dk/en/publications/foundations-and-evolution-of-modern-computing-paradigms-cloud-iot/.

26. YUAN Jingling. ELECT: Energy-efficient intelligent edge–cloud collaboration for remote IoT services. *Future Generation Computer Systems*. 2023. Vol. 147. Pp. 179-194. https://www.sciencedirect.com/science/article/abs/pii/S0167739X23001681..

27. Alsamhi S.H.; Shvetsov A.V.; Kumar S.; Hassan J.; Alhartomi M.A.; Shvetsova S.V.; Sahal R.; Hawbani A. Computing in the sky: A survey on intelligent ubiquitous computing for uav-assisted 6g networks and industry 4.0/5.0. *Drones 2022*. Vol. 6. Pp. 177. https://www.mdpi.com/2504-446X/6/7/177.

28. Ometov A., Molua O.L., Komarov M., Nurmi J. A survey of security in cloud, edge, and fog computing. *Sensors 2022*. Vol. 22. Pp. 927. https://www.mdpi.com/1424-8220/22/3/927.

29. Ren J., Zhang D., He S., Zhang Y., Li T. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Comput. Surv*. 2019. Vol. 52. Pp. 1–36. https://dl.acm.org/doi/abs/10.1145/3362031.

30. Zhang Y., Ren J., Liu J., Xu C., Guo H. Liu Y. A survey on emerging computing paradigms for big data. *Chin. J. Electron*. 2017. Vol. 26. Pp. 1–12. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cje.2016.11.016..

31. Dustdar S., Pujol V.C., Donta, P.K. On distributed computing continuum systems. *IEEE Trans. Knowl. Data Eng.* 2022. Vol. 35. Pp. 4092–4105. https://ieeexplore.ieee.org/abstract/document/9681207/.

32. Casamayor Pujol, V. Morichetta, A. Murturi, I. Kumar Donta, P. Dustdar. Fundamental research challenges for distributed computing continuum systems. *Information*. 2023. Vol. *14*. Pp. 198. https://www.mdpi.com/2078-2489/14/3/198.

33. Donta P. K., Dustdar S. The promising role of representation learning for distributed computing continuum systems. *IEEE International Conference on Service-*

Oriented System Engineering (SOSE). 2022. Pp. 126-132. https://ieeexplore.ieee.org/abstract/document/9912640.

34. S. Dustdar, V. C. Pujol, P. K. Donta. On distributed computing continuum systems. *IEEE Transactions on Knowledge and Data Engineering*. 2022. https://ieeexplore.ieee.org/abstract/document/9681207.

35. Buyya Rajkumar, Satish Narayana Srirama. eds. Fog and edge computing: principles and paradigms. *John Wiley & Sons*. 2019. https://books.google.com.ua/books?hl=uk&lr=&id=v4B_DwAAQBAJ&oi=fnd&pg=PP19&ots=fqxy887QDA&sig=h0ooBUapjKNJ_OuRuRiEPnB2XAY&redir_esc=y#v=onepage&q&f=false.

36. Zhou Z., Chen X., Li E., Zeng L., Luo K., Zhang J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE. 2019. Vol. 107(8). Pp. 1738-1762. https://ieeexplore.ieee.org/abstract/document/8736011.

37. Deng S., Zhao H., Fang W., Yin J., Dustdar S., Zomaya A.Y. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*. 2020. Vol. 7(8). Pp.7457-7469. https://ieeexplore.ieee.org/abstract/document/9052677.

38. Badidi E., Mahrez Z., Sabir, E. Fog computing for smart cities' big data management and analytics: A review. *Future Internet*. 2020. Vol. *2*(11). Pp.190. https://www.mdpi.com/1999-5903/12/11/190.

39. Sharma M., Joshi S., Kannan D., Govindan K., Singh R., Purohit H. C. Internet of Things (IoT) adoption barriers of smart cities' waste management: An Indian context. Journal of Cleaner Production. 2020. Vol. 270. Pp. 122047. https://www.sciencedirect.com/science/article/abs/pii/S0959652620320941.

40. Bayılmış C., Ebleme M. A., Çavuşoğlu Ü., Küçük K., Sevin A. A survey on communication protocols and performance evaluations for Internet of Things. *Digital Communications and Networks*. 2022. Vol. 8(6). Pp. 1094-1104. https://www.sciencedirect.com/science/article/pii/S2352864822000347.

41. Esparza J., Jaax S., Raskin M., Weil-Kennedy C. The complexity of verifying population protocols. Distributed Computing. 2021. Vol. 34. Pp. 133–177. https://link.springer.com/article/10.1007/s00446-021-00390-x.

42. Ding A.Y., Peltonen E., Meuser T., Aral A., Becker C., Dustdar S., Hiessl T., Kranzlmüller D., Liyanage M., Maghsudi S., Mohan, N. Roadmap for edge AI: A Dagstuhl perspective. *ACM SIGCOMM Computer Communication Review*. 2022. Vol. 52(1). Pp.28-33. https://dl.acm.org/doi/abs/10.1145/3523230.3523235.

43. Rybicki J., Solnerzik J., Stietel O., Vacus R. Space-efficient population protocols for exact majority on general graphs. *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2026. Pp. 574–612.

44. Zhang J., Zhong S. Modeling and Analysis of Proof-Based Strategies for Distributed Consensus in Blockchain-Based Peer-to-Peer Networks. *Sustainability*. 2023. Vol. 15(2). P. 1478. https://doi.org/10.3390/su15021478.

45. Kara M., Laouid A., Hammoudeh M., Karampidis K., Papadourakis G., Bounceur A. A Secure Multi-Agent-Based Decision Model Using a Consensus Mechanism for Intelligent Manufacturing Tasks. *Engineering Proceedings*. 2023. Vol. 56(1). P. 234. https://doi.org/10.3390/ASEC2023-15929.

46. Swarm Intelligence Authors. A Survey of Using Swarm Intelligence Algorithms in IoT. *Sensors*. 2020. Vol. 20(5). P. 1420. https://doi.org/10.3390/s20051420.

47. Cloud Scheduling Research Group. An Improved Genetic Algorithm with Swarm Intelligence for Security-Aware Task Scheduling in Hybrid Clouds. *Electronics*. 2023. Vol. 12(9). P. 2064. https://doi.org/10.3390/electronics12092064.

48. IoT Research Network. Developing a Novel Adaptive Double Deep Q-Learning-Based Routing Strategy for IoT-Based Wireless Sensor Network with Federated Learning. *Sensors*. 2025. Vol. 25(10). P. 3084. https://doi.org/10.3390/s25103084.

49. Hitimana E., Bajpai G., Musabe R., Sibomana L., Kayalvizhi J. Implementation of IoT Framework with Data Analysis Using Deep Learning Methods

for Occupancy Prediction in a Building. *Future Internet.* 2021. Vol. 13(3). P. 67. https://doi.org/10.3390/fi13030067.

50. Watts D.J., Strogatz S.H. Collective Dynamics of Small-World Networks. *MDPI Entropy*. 2022. Vol. 24(1). P. 120. https://doi.org/10.3390/e24010120.

51. Pellegrino M., Lombardo G., Cagnoni S., Poggi A. Modern Trends in Multi-Agent Systems. *Future Internet.* 2024. Vol. 16(2). P. 54. https://doi.org/10.3390/fi16020054.

52. Saif M.B., Migliorini S., Spoto F. Efficient and Secure Distributed Data Storage and Retrieval Using Interplanetary File System and Blockchain. *Future Internet. 2024*. Vol. 16(3). P. 98. https://doi.org/10.3390/fi16030098.

53. Huang Y., Zhang S., Wang B. An Improved Genetic Algorithm with Swarm Intelligence for Security-Aware Task Scheduling in Hybrid Clouds. *Electronics*. 2023. Vol. 12(9). P. 2064. https://doi.org/10.3390/electronics12092064.

54. Dowdeswell B., Sinha R., MacDonell S.G. Architecting an Agent-Based Fault Diagnosis Engine for IEC 61499 Industrial Cyber-Physical Systems. *Future Internet. 2021*. Vol. 13(8). P. 190. https://doi.org/10.3390/fi13080190.

55. Vlase S., Negrean I., Marin M., Scutaru M.L. Energy of Accelerations Used to Obtain the Motion Equations of a Three-Dimensional Finite Element. *Symmetry*. 2020. Vol. 12(2). P. 321. https://doi.org/10.3390/sym12020321.

56. Letia T.S., Durla-Pasca E.M., Al-Janabi D., Cuibus O.P. Development of Evolutionary Systems Based on Quantum Petri Nets. *Mathematics*. 2022. Vol. 10(23). P. 4404. https://doi.org/10.3390/math10234404.

57. Saxena A., Pare S., Meena M.S., Gupta D., Gupta A., Razzak I., Lin C.-T., Prasad M. A Two-Phase Approach for Semi-Supervised Feature Selection. *Algorithms*. 2020. Vol. 13(9). P. 215. https://doi.org/10.3390/a13090215.

58. Kenyeres M.; Kenyeres J. Comparative Study of Distributed Consensus Gossip Algorithms for Network Size Estimation in Multi-Agent Systems. *Future Internet*. 2021.Vol. 13. Pp. 134. https://doi.org/10.3390/fi13050134.

59. Donta P. K., Murturi I., Casamayor Pujol V., Sedlak B., Dustdar S. Exploring the Potential of Distributed Computing Continuum Systems. *Computers*. 2023. Vol. 12. Pp. 198. https://www.sciencedirect.com/science/article/abs/pii/S157401372600050X.

60. Alistarh D., Gelashvili R., Vojnovic M. Fast and Exact Majority in Population Protocols. *Journal of the ACM*. 2017. Vol. 64(3). Pp. 1-22. https://www.researchgate.net/publication/300175420_Fast_and_Exact_Majority_in_Population_Protocols.

61. Pellegrino M., Lombardo G., Cagnoni S., Poggi A. Modern Trends in Multi-Agent Systems. *Future Internet*. 2024. Vol. 16(2). P. 54.

62. Kenyeres M., Budinská I., Hluchý L., Poggi A. Modern Trends in Multi-Agent Systems. *Future Internet*. 2024. Vol. 16(2). P. 54. https://doi.org/10.3390/fi16020054.

63. Berenbrink P., Elsässer R., Friedetzky T., Kaaser D., Kling P., Radzik T. A Population Protocol for Exact Majority with $O(\log^{5/3}*n)$ Stabilization Time and O(log n). *32nd International Symposium on Distributed Computing (DISC 2018)*. 2018. Vol. 121. Pp. 10:1-10:18. https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2018.10.

64. Dorigo M., Maniezzo V., Colorni A. Ant Colony Optimization and Swarm Intelligence for Distributed Task Allocation. *Algorithms*. 2021. Vol. 14(4). P. 100. https://www.academia.edu/18706424/Ant_Colony_Optimization_and_Swarm_Intelligence.

65. Altisen K., Devismes S., Dubois C. Formally Verifying Self-Stabilizing Population Protocols. *Journal of Automated Reasoning*. 2020. Vol. 64(6). Pp. 1045-1078. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www-verimag.imag.fr/~monin/Publis/Docs/tase09-leader.pdf.

66. Dolev S., Hanemann A. Self-stabilizing Protocols for Dynamic Distributed Environments: A Comprehensive Review. *Algorithms*. 2020. Vol. 13(9). P. 215. https://www.researchgate.net/publication/221343332_Self-Stabilization_of_Dynamic_Systems_Assuming_only_ReadWrite_Atomicity.

67. Awajan A. A Novel Deep Learning-Based Intrusion Detection System for IoT Networks. *Computers*. 2023. Vol. 12(2). P. 34. https://doi.org/10.3390/computers12020034.

68. Kara M., Laouid A., Bounceur A. Fault Tolerance Structures in Wireless Sensor Networks (WSNs): Survey, Classification, and Future Directions. *Sensors*. 2022. Vol. 22(16). P. 6041. https://doi.org/10.3390/s22166041.

69. Roudsari S.S., Ungureanu L.M., Soroushnia S., Abu-Lebdeh T., Petrescu F.I.T. Blockchain-Based Distributed Computing Framework and Resiliency Analysis. *Algorithms*. 2022. Vol. 15(1). P. 12. https://doi.org/10.3390/a15010012.

70. Golovach P.A., Paulusma D., van Leeuwen E.J. Induced Disjoint Paths in AT-free graphs. *Journal of Computer and System Sciences*. 2022. Vol. 124. Pp. 112-131. https://doi.org/10.1016/j.jcss.2021.10.003..

71. Dowdeswell B., Sinha R., MacDonell S.G. Architecting an Agent-Based Fault Diagnosis Engine for IEC 61499 Industrial Cyber-Physical Systems. *Future Internet*. 2021. Vol. 13(8). P. 190. https://doi.org/10.3390/fi13080190.

72. Kakkavas G., Karyotis V., Papavassiliou S. Topology Inference and Link Parameter Estimation Based on End-to-End Measurements. *Future Internet*. 2022. Vol. 14(2). P. 45. https://doi.org/10.3390/fi14020045.

73. Mehmood K.T., Atiq S., Hussain M.M. Enhancing QoS of Telecom Networks through Server Load Management in Software-Defined Networking. *Sensors*. 2023. Vol. 23(23). P. 9324. https://doi.org/10.3390/s23239324.

74. El Debeiki M., Al-Rubaye S., Perrusquía A. An Advanced Path Planning and UAV Relay System: Enhancing Connectivity in Rural Environments. *Future Internet*. 2024. Vol. 16(3). P. 89. https://doi.org/10.3390/fi16030089.

75. Stoecklin S., Yousaf A., Gidion G., Reindl L., Rupitsch S.J. Simultaneous Power Feedback and Maximum Efficiency Point Tracking for Miniaturized RF Wireless Power Transfer Systems. *Sensors*. 2021. Vol. 21(6). P. 2023. https://doi.org/10.3390/s21062023.

76. Shoham Y., Leyton-Brown K. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. *Games*. 2022. Vol. 13(4). P. 45. chrome-

extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.masfoundations.org/mas.pdf.

77. Chopard B., Droz M. Cellular Automata and Discrete Modeling of Complex Distributed Systems. *Entropy*. 2021. Vol. 23(8). P. 900. https://www.researchgate.net/profile/Michel-Droz/publication/216300438_Cellular_Automata_Modeling_of_Physical_Systems/links/00b49526fb2ee12d01000000/Cellular-Automata-Modeling-of-Physical-Systems.pdf.

78. Letia T.S., Durla-Pasca E.M., Al-Janabi D., Cuibus O.P. Development of Evolutionary Systems Based on Quantum Petri Nets. *Mathematics*. 2022. Vol. 10(23). https://www.mdpi.com/2227-7390/10/23/4404.

79. Yang X.-S. Flower Pollination Algorithm for Global Optimization and Network Layouts. *Mathematics*. 2022. Vol. 10(23). chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://arxiv.org/pdf/1312.5673.

80. Zheng Z., Xie S., Dai H.-N., Chen X., Wang H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *6th IEEE International Congress on Big Data (BigData Congress 2017)*. 2017. Pp. 557-564. https://www.researchgate.net/publication/318131748_An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends.

81. Kolomytskyi D., Rehida P., Onyshko O., Ilchyshyna Y. Generalized Method for Managing the Lifecycle of Terraform Infrastructure Across Multiple Environments. Herald of Khmelnytskyi National University. Technical Sciences. 2026. Vol. 363, no. 2. P. 117–125.

ДОДАТОК А
(обов’язковий)

Тези – ПерСик

МЕТОД СИНТЕЗУ РОЗПОДІЛЕНИХ КОМП’ЮТЕРНИЗ СИСТЕМ НА ОСНОВІ ПРОТОКОЛІВ ПОПУЛЯЦІЇ

Ільчишина Ю. В. студентка гр. КІ2м-24-2
Науковий керівник: д. т. н., професор Савенко О. С.
Хмельницький національний університет

Актуальність. Розподілені комп’ютерні системи сьогодні є основою багатьох сучасних інформаційних технологій, зокрема хмарних сервісів, сенсорних мереж і систем Інтернету речей. У зв’язку з цим важливими стають підходи, що базуються на простих правилах взаємодії елементів, зокрема протоколи популяцій.

Метою даної роботи є порівняльний аналіз централізованих та децентралізованих систем для спілкування (месенджерів) з огляду на безпеку та конфіденційність.

Аналіз рішень. Розглянуто модель розподіленої системи у вигляді сукупності однотипних агентів із обмеженими ресурсами, взаємодія між якими відбувається відповідно до правил протоколів популяцій. Кожен агент має скінченну множину станів і змінює їх у процесі взаємодії з іншими.

Запропонований метод синтезу ґрунтується на формуванні глобальної поведінки системи через локальні взаємодії, що дає змогу відмовитися від централізованого керування. У межах дослідження проаналізовано умови досягнення стабільного стану та забезпечення збіжності системи. Встановлено, що простота взаємодій сприяє високій масштабованості системи, проте швидкість досягнення результату визначається кількістю агентів і частотою їхніх взаємодій.

Результати. Дослідження показують, що використання протоколів популяцій дозволяє будувати стійкі до збоїв розподілені системи, які здатні функціонувати навіть у разі втрати окремих елементів.

Висновки. Метод синтезу на основі протоколів популяцій є ефективним для створення масштабованих і стійких систем із обмеженими ресурсами. Подальші дослідження доцільно спрямувати на підвищення швидкості збіжності та розширення можливостей застосування.

ДОДАТОК Б
(обов'язковий)

Публікація

UDC 004.75

DENYS KOLOMYTSKYI, PAVLO REHIDA, OKSANA ONYSHKO,
YULIIA ILCHYSHYNA
Khmelnytskyi National University, Khmelnytskyi, Ukraine

GENERALIZED METHOD FOR MANAGING THE LIFECYCLE OF TERRAFORM INFRASTRUCTURE ACROSS MULTIPLE ENVIRONMENTS

The article examines the challenge of managing the lifecycle of cloud infrastructure, as described using Terraform, across multiple environments (development, staging, and production). In industrial settings, multi-environment infrastructure management gives rise to a set of interrelated challenges: the absence of formalized procedures for promoting changes between environments with explicit source readiness verification, fragmented compliance checking that fails to distinguish code-level syntax validation from plan-level semantic verification, and configuration drift detection that lacks classification by severity, generating false alerts for expected changes such as dynamic IP addresses and rotated certificates. An analysis of existing approaches to environmental isolation, configuration compliance verification, and drift detection reveals that none of the current methods address the full lifecycle in a unified manner. A generalized method is proposed, based on a formalized two-projection lifecycle model: the horizontal projection describes an eight-stage finite automaton of an individual environment (Init, Author, Validate, Plan, Comply, Approve, Apply, Monitor), while the vertical projection defines a partially ordered environment space with a formalized promotion operation. The method introduces a source readiness precondition requiring the source environment to be in the monitoring stage with empty planning and drift deltas, versioned configuration snapshots ensuring code identity across environments, multi-level compliance verification across code, plan, and state levels using hierarchical policy inheritance, and a three-class drift classification (critical, actionable, informational) with an effective delta mechanism that filters expected changes. Experimental verification on a real multi-environment AWS infrastructure (27 managed resources, 3 environments) using the Scalr platform confirms that the proposed method ensures automated detection of 100% of policy violations before the apply stage (compared to 50% for GitOps CI/CD and 0% for Terraform CLI), reduces false drift alerts to zero, and decreases the number of manual promotion steps to a single approval for the production environment.

Keywords: Infrastructure as Code, Terraform, lifecycle management, multi-environment infrastructure, configuration drift, compliance verification, DevOps, cloud computing.

Д.Є. КОЛОМИЦЬКИЙ, П.Г. РЕГІДА, О.Г. ОНИШКО, Ю. В. ІЛЬЧИШИНА
Хмельницький національний університет, Хмельницький, Україна

УЗАГАЛЬНЕНИЙ МЕТОД КЕРУВАННЯ ЖИТТЄВИМ ЦИКЛОМ TERRAFORM-ІНФРАСТРУКТУРИ ДЛЯ КІЛЬКОХ СЕРЕДОВИЩ

У статті досліджується проблема управління життєвим циклом хмарної інфраструктури, описаної засобами Terraform, для кількох середовищ (розробки, тестування, виробництва). Здійснено аналіз існуючих підходів до ізоляції середовищ, перевірки відповідності конфігурацій та виявлення конфігураційного дрейфу. Запропоновано узагальнений метод, що ґрунтується на формалізованій двопроекційній моделі життєвого циклу у вигляді восьмистадійного скінченного автомата та частково впорядкованого простору середовищ. Метод включає формалізовану процедуру просування змін між середовищами з передумовою готовності джерела, багаторівневу перевірку відповідності на рівнях коду, плану та стану, а також трикласову класифікацію конфігураційного дрейфу з механізмом версіонованого відкату. Експериментальна верифікація на реальній мультисередовищній інфраструктурі AWS із використанням платформи Scalr підтверджує, що запропонований метод забезпечує виявлення 100% порушень політик до стадії застосування та зведення хибних сповіщень дрейфу до нуля через ефективну дельту.

Ключові слова: Infrastructure as Code, Terraform, управління життєвим циклом, мультисередовищна інфраструктура, конфігураційний дрейф, перевірка відповідності, DevOps, хмарні обчислення.

Overview of the Problem and Its Connection to Major Scientific and Practical Challenges

Modern approaches to software development have experienced a significant shift under the influence of the "Everything as Code" paradigm—a systematic method in which any artifact of a software system, including infrastructure, configuration, and security policies, is described as versioned code [1]. At the core of this paradigm is the practice of Infrastructure as Code (IaC), which involves declaratively describing cloud environment resources in configuration files

stored in a version control system and executed by automated tools. Terraform is the leading tool in this category because of its provider-agnostic design and explicit state management through the terraform.tfstate file, which maps the described resources to the actual entities in the cloud [2].

A typical industrial infrastructure spans multiple environments—development, testing, and production—each consisting of isolated cloud resources at different stages of a software product's lifecycle. Managing this multi-environment setup introduces a complex set of interconnected issues that cannot be addressed by individual tools alone.

Current methods for environment isolation—based on workspaces, directory structures, or repository branches—balance different trade-offs between isolation levels and maintenance complexity [3]. However, none of these methods offers a formal process for promoting changes across environments with clear prerequisites for source readiness. In practice, this can allow a change to reach the production environment from a source that may have undetected configuration drift or an incomplete application, raising the risk of failed deployments. An empirical study of versioning and rollback strategies for IaC templates confirmed that formalizing these procedures can significantly decrease the average recovery time after failures by an order of magnitude [4].

The verification of infrastructure configuration compliance remains scattered. Typical CI/CD pipelines include static security analysis during code validation but do not separate the checking of configuration syntax from verifying the semantics of planned changes as distinct stages. A configuration that is syntactically correct and has no known vulnerabilities can still break organizational policies only at deployment—such as using a forbidden instance type or planning to delete a protected resource. Additionally, studies have shown that adding declarative security policies directly into the resource provisioning pipeline can achieve a 92% improvement in compliance metrics [5].

Configuration drift—the gap between the actual state of cloud resources and what is defined in the code—is another key issue caused by manual changes in the cloud console, partial applications, or external processes outside of Terraform. Existing tools either lack a built-in way to monitor drift or identify discrepancies without prioritizing their severity, leading to a flood of false alerts about expected changes—such as dynamic IP addresses or automatically rotated certificates—which diminishes the response to truly critical issues.

Systematic reviews of IaC technologies show that, despite a considerable amount of research—from tool taxonomies to defect cataloging and formal analyses of idempotency—the comprehensive management of multi-environment infrastructure throughout its entire lifecycle remains neglected in academic studies [6]. There is no universal approach that covers the complete process from configuration setup to automatic drift correction, formalizes change propagation, and guarantees multi-level compliance verification at different stages of the pipeline. Creating such a method is an urgent scientific and practical challenge directly related to enhancing the reliability and security of cloud infrastructures in industrial settings.

Analysis of Recent Research and Publications

Pahl and his co-authors conducted the most thorough systematization of IaC technologies, proposing a classification based on four dimensions—application context, functionality, description language, and execution architecture—and developing a specialized DevOps lifecycle for infrastructure code based on it [7]. This cycle includes stages from code writing and validation to environment self-recovery; however, it is described at a conceptual level without formalizing transitions between stages and without considering coordination across multiple environments.

Based on a systematic review of the gray literature, Kumara and his co-authors identified the fundamental properties of high-quality infrastructure code: declarativeness, idempotence, and reproducibility [8]. Hanappi provided formal confirmation of these properties, demonstrating through state transition graphs that violations of idempotence are a hidden source of errors that only become apparent upon repeated application of configurations [9]. These findings serve as the theoretical foundation for formalizing the lifecycle but do not translate into a specific method for managing environments.

Empirical studies have uncovered systemic gaps in the IaC ecosystem, including fragmentation of tools, a lack of mature testing methods, and a shortage of standardized state management practices [10]. A qualitative analysis of testing practices identified six categories of checks—ranging from static analysis to policy compliance testing—yet none of the existing pipelines implement them as architecturally independent stages with distinct input artifacts [11].

Based on an analysis of 1,448 commits, Rahman and his co-authors developed a taxonomy of eight categories of infrastructure code defects, with configuration data errors being the most common and violations of idempotency being the most specific to IaC [12]. Another area of focus is the detection of configuration drift: Dinu and Fontaine proposed a method for continuously comparing Terraform plan results with an expected empty plan and automatically notifying of discrepancies, although without classifying the detected drift by severity level [13].

In the realm of state management, the fork of the Terraform ecosystem following HashiCorp's license model change is important: the open-source fork OpenTofu, supported by the Linux Foundation, maintains compatibility at the HCL language and provider levels [14], which requires provider independence in any general method.

Research on GitOps pipelines for Terraform shows that deployment frequency increases and recovery time decreases compared to traditional methods; however, the pull synchronization model is limited to planning and application stages without formalizing progress between environments [15].

Thus, existing studies focus on specific aspects—tool classification, formalization of idempotency, testing, drift detection, and GitOps automation—but do not provide a comprehensive method that covers the entire lifecycle of a multi-environment Terraform infrastructure, including formalized change propagation, multi-level compliance verification, and classification of configuration drift.

Statement of the article's objectives

The purpose of this article is to create a comprehensive approach for managing the lifecycle of Terraform infrastructure across multiple environments. To achieve this, three tasks are outlined: (1) formalize a two-projection lifecycle model as a finite state machine and a partially ordered environment space, covering stages from configuration setup to detecting and fixing configuration drift; (2) justify a method for propagating changes between environments with source readiness requirements, multi-level compliance checks, and a versioned rollback system; (3) test the approach on a real multi-environment infrastructure and evaluate its effectiveness using specific metrics.

Conceptual Model and Formalization of the Method

Formalization of Concepts and the Lifecycle Model

Let M be a finite set of Terraform modules, where each module $m = \langle R_m, V_m, O_m \rangle$ is defined by a set of resource descriptions R_m, input variables V_m and output values O_m. An infrastructure configuration is defined as a pair:

$$C = \langle m_{root}, \Theta \rangle, \quad (1)$$

where $m_{root} \in M$ — is the root module (entry point), and, $\Theta = \{\theta_1, \dots, \theta_k\}$ — is the set of input variable values that parameterizes a specific instance of the infrastructure. It is Θ that determines the differences between environments when the root module is shared.

The environment is defined by four factors:

$$e = \langle id_e, C_e, S_e, P_e \rangle, \quad (2)$$

where id_e — is a unique identifier; C_e — is the configuration; S_e — is the current state; P_e — is the set of compliance policies. The inclusion of P_e in the definition reflects the principle of built-in compliance: policies are an integral part of the environment.

The concept of state involves distinguishing three entities. The desired state $S^d_e = eval(C_e)$ is set by interpreting the configuration. The recorded state S^r_e— is the content of terraform.tfstate, which updates after each successful operation. The actual state S^a_e— includes resources that presently exist in the cloud and may differ from S^r_e due to changes outside of Terraform. This three-part model allows us to formally define the *planning delta* $\Delta^{plan}_e = S^d_e \setminus S^r_e$ and the drift delta as a symmetric difference:

$$\Delta^{drift}_e = S^r_e \triangle S^a_e, \quad (3)$$

which covers both resources that are modified or deleted outside of Terraform and resources added to the cloud without a matching description in the configuration. The environment is *consistent* if both deltas are empty.

A set of environments forms the environment space $E = \{e_1, \dots, e_n\}$ with a partial order relation $\preceq$. For a typical configuration such as $e_{dev} \preceq e_{staging} \preceq e_{prod}$ a change reaches the production environment only after passing through all intermediate environments.

The lifecycle of a single environment is modeled as a finite state machine $A_e = \langle \Sigma, \delta, \sigma_1, F \rangle$ with eight stages: σ_1 (Init) — initializing the backend and providers; σ_2 (Author) — making configuration changes; σ_3 (Validate) — performing syntax checks and static security analysis; σ_4 (Plan) — calculating Δ_e^{plan} without applying it; σ_5 (Comply) — verifying the plan's semantics for policy compliance; σ_6 (Approve) — obtaining approval (manual for production, automatic for development); σ_7 (Apply) — implementing the plan, which updates $S_e^r \leftarrow S_e^a$; σ_8 (Monitor) — periodically calculating Δ_e^{drift}. A failure at stages σ_3, σ_5 or σ_6 causes the automaton to return to σ_2 to correct the configuration, while detecting drift at σ_8 triggers a transition back to σ_4, completing the feedback loop. The automaton's structure and feedback transitions are illustrated in Fig. 1.

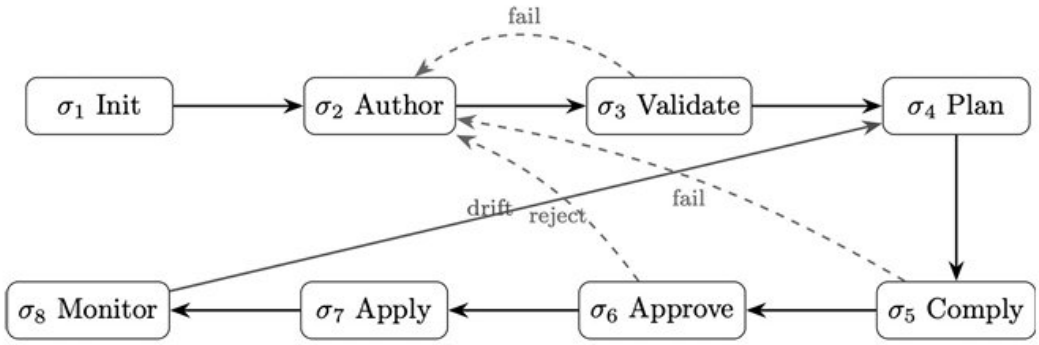


Fig. 1. – Finite-state machine A_e depicts the life cycle of a single environment. Solid arrows show the primary transition sequence; dashed arrows indicate backtransitions in case of failure; the transition $\sigma_8 \rightarrow \sigma_4$ – illustrates the drift elimination cycle.

A key difference from existing conveyors is the architectural separation of σ_3 and σ_5: the former verifies the predicate $valid(C_e, P_e^{code})$ without consulting the provider, while the latter verifies $comply(\Delta_e^{plan}, P_e^{plan})$ on the plan artifact with computed resource attributes.

Promotion and Rollback Mechanism

The promotion operation between adjacent environments $e_i \preceq e_j$ is defined as:

$$promote(e_i, e_j) : C_{e_j} \leftarrow Sm_{root}, \Theta_{e_j}, \tag{4}$$

where the root module remains shared, and Θ_{e_j} reflects the specifics of the target environment:

$$ready(e_i) \Leftrightarrow stage(e_i) = \sigma_8 \wedge \Delta_{e_i}^{plan} = \emptyset \wedge \Delta_{e_i}^{drift} = \emptyset, \tag{5}$$

in other words, the source environment is in the monitoring phase with empty deltas. None of the existing approaches automatically verify this condition.

Code identity between environments is ensured through *versioned configuration snapshots* $\mathbb{Z}_e = \langle m_{root}, \Theta_e, v \rangle$ where v — is the version number assigned after successfully completing a full cycle in the source environment. A snapshot is an immutable artifact that prevents implicit changes in the dependencies between testing in e_i and deployment in e_j.

After advancing to the target environment, stages $\sigma_3 \rightarrow \sigma_5 \rightarrow \sigma_6 \rightarrow \sigma_7$ are executed sequentially with policies P_{e_j}, ensuring independent compliance verification for each environment.

The rollback mechanism relies on the history of state snapshots $H_e = [S_e^{r,1}, \ldots, S_e^{r,v}]$. The $rollback(e, v')$ operation restores the saved state $S_e^{r,v'}$ followed by the cycle $\sigma_4 \rightarrow \sigma_7$ to bring the actual state into compliance. The safety of the rollback is determined by the predicate:

$$safe(e, v, v') \Leftrightarrow \forall r \in c S_e^{r,v} \setminus S_e^{r,v'} d : rev(r) = 1, \quad (6)$$

where $rev(r) = 1$ for reversible resources (computational instances, network rules) and $rev(r) = 0$ for irreversible ones (databases with accumulated transactions). If $safe = false$ the σ_6 approval stage is forcibly switched to manual mode.

Multi-level compliance checking and drift classification

The set of environment policies is divided according to compliance levels:

$$P_e = P_e^{code} \cup P_e^{plan} \cup P_e^{state}, \quad (7)$$

where P_e^{code} applies to σ_3 (syntax, hardcoded secrets, open ports), P_e^{plan} – to σ_5 (prohibited instance types, deletion of protected resources, cost estimation), and P_e^{state} — to σ_8 (compliance of the actual state with security requirements). Policies are organized into an inheritance hierarchy: $P_e = P_{1obal} \cup P_{env(e)} \cup P_{comp(e)}$, with strictness increasing along $\preceq$ — the same policy may be advisory in e_{dev} and hard-mandatory in e_{prod}.

Detected drift is categorized into three types:

$$class(r, \Delta_e^{drift}, P_e^{state}) \in \{\text{critical, actionable, informational}\}, \quad (8)$$

Critical drift (a violation of a strict mandatory policy) is automatically resolved by $\sigma_8 \rightarrow \sigma_4$. Actionable drift (a discrepancy that does not breach critical policies) requires manual confirmation. Informational drift (expected changes such as dynamic IP addresses or rotated certificates) is excluded from the effective delta through the set of ignored attributes I_e:

$$\Delta_e^{eff} = \Delta_e^{drift} \setminus \{r: attr(r) \in I_e\}, \quad (9)$$

which reduces false alerts without compromising oversight of critical changes.

Implementation and Experimental Evaluation of the Method

To validate the method, we selected platform [16]—an industrial-grade Terraform/OpenTofu infrastructure orchestration system that features a directory-based isolation pattern, hierarchical configuration inheritance, and built-in compliance checking via Open Policy Agent (OPA). The experimental infrastructure is deployed in Amazon Web Services (AWS) cloud and includes a typical web service: network layer (VPC, subnets, security groups), compute layer (EC2), storage layer (RDS PostgreSQL, S3), and access management (IAM). The configuration is organized into a library of five submodules, totaling approximately 1,200 lines of HCL and 27 managed resources. Environment space: $e_{dev} \preceq e_{sta1in1} \preceq e_{prod}$.

The platform's Run pipeline directly maps to the A_e: state machine: two independent OPA policy checkpoints correspond to stages σ_3 (configuration check before planning) and σ_5 (check of the JSON plan artifact after planning). Monitoring of σ_8 is carried out through drift detection—periodic execution of `terraform plan` in refresh-only mode. Manual execution of the Terraform CLI and a typical GitOps CI/CD pipeline using GitLab CI, integrated with a static security analyzer, were chosen as comparison methods.

Experiment 1: Multi-level compliance verification.

Four violation scenarios were identified: two code-level violations—(C1) a hardcoded AWS secret and (C2) a security group with a 0.0.0.0/0 rule on port 22; and two plan-level violations—(C3) a prohibited m5.24xlarge instance type and (C4) deletion of a protected RDS instance. Each violation was added as a separate commit to the $e_{sta1in1}$ configuration.

Table 1 – The stage of detecting violations by approach

Scenario	Level	Terraform CLI	GitOps CI/CD	Method (Scalr)
C1	P^{code}	Not automated	σ_3	σ_3(OPA)
C2	P^{code}	Not automated	σ_3	σ_3(OPA)
C3	P^{plan}	Not automated	Not automated	σ_5(OPA)
C4	P^{plan}	Not automated	Not automated	σ_5(OPA)

The proposed method achieves $R_{pre-apply} = 1{,}0$ – all four violations are detected automatically before the apply stage. GitOps CI/CD detects only code-level violations ($R_{pre-apply} = 0{,}5$), because the static analyzer does not process the plan artifact. The Terraform CLI has no built-in mechanisms for automated policy verification ($R_{pre-apply} = 0{,}0$): violations may be noticed by the operator during a manual review of the terraform plan output, but this depends on attentiveness and experience, and it is not a systematic check. The method's detection time is 15–40 seconds for code-level violations and 45–90 seconds for plan-level violations.

Experiment 2: drift detection and classification.

Three types of external changes were simulated via the AWS console: (D1) critical – adding the rule 0.0.0.0/0:443 to the prod-environment security group; (D2) operational – changing the EC2 instance tag; (D3) informational – changing the private IP address after a restart. The drift detection interval was set to $\tau = 5$ minutes to accelerate data collection (the typical value for production environments is 1–24 hours, depending on organizational requirements).

Table 2 – Drift Detection and Classification Results

Scenario	Class	Terraform CLI	GitOps CI/CD	Метод (Scalr)
D1	Critical	Not automated	Not automated	Auto-resolution, 6.2 min.
D2	Operational	Not automated	Not automated	Notification, awaiting confirmation
D3	Informational	Not automated	Not automated	Excluded from Δ_e^{eff}

The Terraform CLI and GitOps CI/CD lack built-in automated drift monitoring; detection requires manually running `terraform plan` or configuring a separate cron job. The proposed method accurately identified all three types of drift. Critical drift (D1) was automatically addressed in 6.2 minutes, including the detection cycle wait and the full $\sigma_4 \rightarrow \sigma_7$. Informational drift (D3) was filtered out from the effective delta via the I_e set, ensuring $F_{false} = 0$ – no false positives.

Experiment 3: Automation of the promotion process.

For the full promotion cycle $e_{dev} \preceq e_{stalin1} \preceq e_{prod}$ we recorded the number of manual steps N_{manual} (each CLI command or operator action requiring manual input counts as one step) and whether there was an automated check for the readiness of the source environment.

Table 3 – Comparison of promotion automation

Characteristics	Terraform CLI	GitOps CI/CD	Method (Scalr)
N_{manual}	9	3	1
Validation of $ready(e_i)$	None	None	Automatic
Policy validation during deployment	None	Partial (P^{code})	Full (P^{code}, P^{plan})

For Terraform CLI, $N_{manual} = 9$: the operator executes a plan + apply sequence for each of the three environments (6 commands) and reviews the plan output before each apply (3 reviews). For GitOps CI/CD $N_{manual} = 3$: manually launching the pipeline or merging for each environment without automatic readiness verification $ready(e_i)$. The proposed method requires $N_{manual} = 1$ – approval of the application only for the production environment; the remaining stages, including source readiness verification and cascading propagation, are automated.

Conclusions from This Research and Prospects for Further Study

This article introduces a generalized approach for managing the lifecycle of Terraform infrastructure across multiple environments. It relies on a formalized two-projection model: the horizontal projection outlines an eight-state finite automaton for a single environment, while the vertical projection represents a partially ordered space of environments with a formal transition operation.

A key theoretical contribution is the architectural separation of the verification of code compliance σ_3 and plan semantics σ_5 into independent stages with different input artifacts—something none of the approaches considered achieve. The formalization of the source readiness predicate $ready(e_i)$ or versioned configuration snapshots and the rollback safety predicates$safe(e, v, v')$ ensures atomicity of progression and controlled rollback, taking into account resource

reversibility. A three-class drift classification with an effective delta enables differentiated responses: automatic remediation of critical violations, confirmation for operational violations, and filtering of informational violations.

Experimental verification on a real AWS infrastructure (27 resources, 3 environments) confirmed that the method provides automated detection of 100% of policy violations prior to the deployment stage (compared to 50% in GitOps CI/CD and 0% in Terraform CLI), reduces false drift alerts to zero, and reduces manual deployment steps to a single approval for the production environment.

The method has certain limitations. First, it depends on a directory-based isolation pattern and HCL semantics, which guarantees compatibility with Terraform and OpenTofu but excludes tools with fundamentally different architectures (stateless or agent-based). Second, experimental validation was conducted on a single platform (Scalr) that natively implements the required mechanisms; applying it to other platforms necessitates adapting the OPA integration and the drift detection mechanism. Third, the method does not include automated cost estimation of infrastructure changes as a formalized stage.

Prospects for future research include incorporating cost estimation as part of the automaton, adapting the method to nonlinear topologies of environment spaces with parallel propagation chains, and testing on alternative orchestration platforms.

Declaration on the use of generative artificial intelligence tools

During the preparation of this work, the authors used Grammarly in order to: grammar and spelling check; DeepL Translate in order to: some phrases translation into English. After using these tools and services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

Author Contributions

Conceptualization, Denys Kolomytskyi, Pavlo Rehida and Oksana Onyshko; methodology, Denys Kolomytskyi and Pavlo Rehida; software, Denys Kolomytskyi and Oksana Onyshko; validation, Pavlo Rehida and Yuliia Ilchyshyna; formal analysis, Yuliia Ilchyshyna; investigation, Pavlo Rehida and Yuliia Ilchyshyna; resources, Oksana Onyshko; data curation, Denys Kolomytskyi and Oksana Onyshko; writing – original draft preparation, Denys Kolomytskyi and Oksana Onyshko; writing – review and editing, Pavlo Rehida and Yuliia Ilchyshyna.

References

1. Wei H., Madhavji N., Steinbacher J. *Understanding Everything as Code: A Taxonomy and Conceptual Model*. 2025. 12 p. (Preprint. arXiv; 2507.05100). DOI: 10.48550/arXiv.2507.05100.

2. *Terraform Documentation* / HashiCorp. 2026. Available at: https://developer.hashicorp.com/terraform/docs (accessed: March 3, 2026).

3. Brikman Y. *How to manage multiple environments with Terraform*. Gruntwork Blog. 2022. Available at: https://medium.com/gruntwork/how-to-manage-multiple-environments-with-terraform-32c7bc5d692 (accessed: March 5, 2026).

4. Karanam R. *Multi-cloud IaC template versioning and rollback strategies: An empirical study with Terraform and GitOps*. World Journal of Advanced Research and Reviews. 2024. Vol. 22, no. 02. P. 2354–2363. DOI: 10.30574/wjarr.2024.22.2.1357.

5. Sharma A., Rodriguez E., Tanaka K., Johnson M. *Scaling Infrastructure Governance: A Policy-Driven Framework Combining Terraform Automation and Red Hat Satellite*. ResearchGate. 2026. Available at: https://www.researchgate.net/publication/399734521_Scaling_Infrastructure_Governance_A_Policy-Driven_Framework_Combining_Terraform_Automation_and_Red_Hat_Satellite (accessed: March 2, 2026).

6. Pahl C., Gunduz N. G., Sezen O. C., Ghamgosar A., El Ioini N. *Infrastructure as Code – Technology Review and Research Challenges*. Proceedings of the 20th International Conference on Cloud Computing and Services Science (CLOSER 2025). 2025. P. 1–8. DOI: 10.5220/0012625200003851.

7. Pahl C., Gunduz N. G., Sezen Ö. C., Ghamgosar A., Hofer F., El Ioini N. *A Systematic Review of Infrastructure-as-Code Technologies*. Proceedings of the 15th International Conference on Cloud Computing and Services Science (CLOSER 2025). 2025. Vol. 1. P. 151–158.

8. Kumara I., Garriga M., Romeu A. U., Di Nucci D., Palomba F., Tamburri D. A., van den Heuvel W. J. *The Do's and Don'ts of Infrastructure Code: A Systematic Gray Literature Review*. Information and Software Technology. 2021. Vol. 137. Art. 106593. DOI: 10.1016/j.infsof.2021.106593.

9. Hanappi O., Hummer W., Dustdar S. *Asserting Reliable Convergence for Configuration Management Scripts*. Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2016). 2016. P. 328–343. DOI: 10.1145/2983990.2984000.

10. Guerriero M., Garriga M., Tamburri D. A., Palomba F. *Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry*. 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). Cleveland, OH, USA, 2019. P. 580–589. DOI: 10.1109/ICSME.2019.00092.

11. Hasan Mohammed Mehedi, Bhuiyan Farzana Ahamed, Rahman Akond. *Testing Practices for Infrastructure as Code*. LANGETI 2020: Proceedings of the 1st ACM SIGSOFT International Workshop on Languages and Tools for Next-Generation Testing. 2020. P. 7–12. DOI: 10.1145/3416504.3424334.

12. Rahman A., Farhana E., Parnin C., Williams L. *Gang of Eight: A Defect Taxonomy for Infrastructure as Code Scripts*. Proceedings of the 42nd International Conference on Software Engineering (ICSE). 2020. P. 752–764. DOI: 10.1145/3377811.3380409.

13. Dinu F., Fontaine J.-M. *Infrastructure Drift Detection — How to Fix It with IaC Tools*. Spacelift Blog. 2024. Available at: https://spacelift.io/blog/drift-detection (accessed: March 3, 2026).

14. *OpenTofu Documentation* / OpenTofu. 2026. Available at: https://opentofu.org/docs/ (accessed: March 3, 2026).

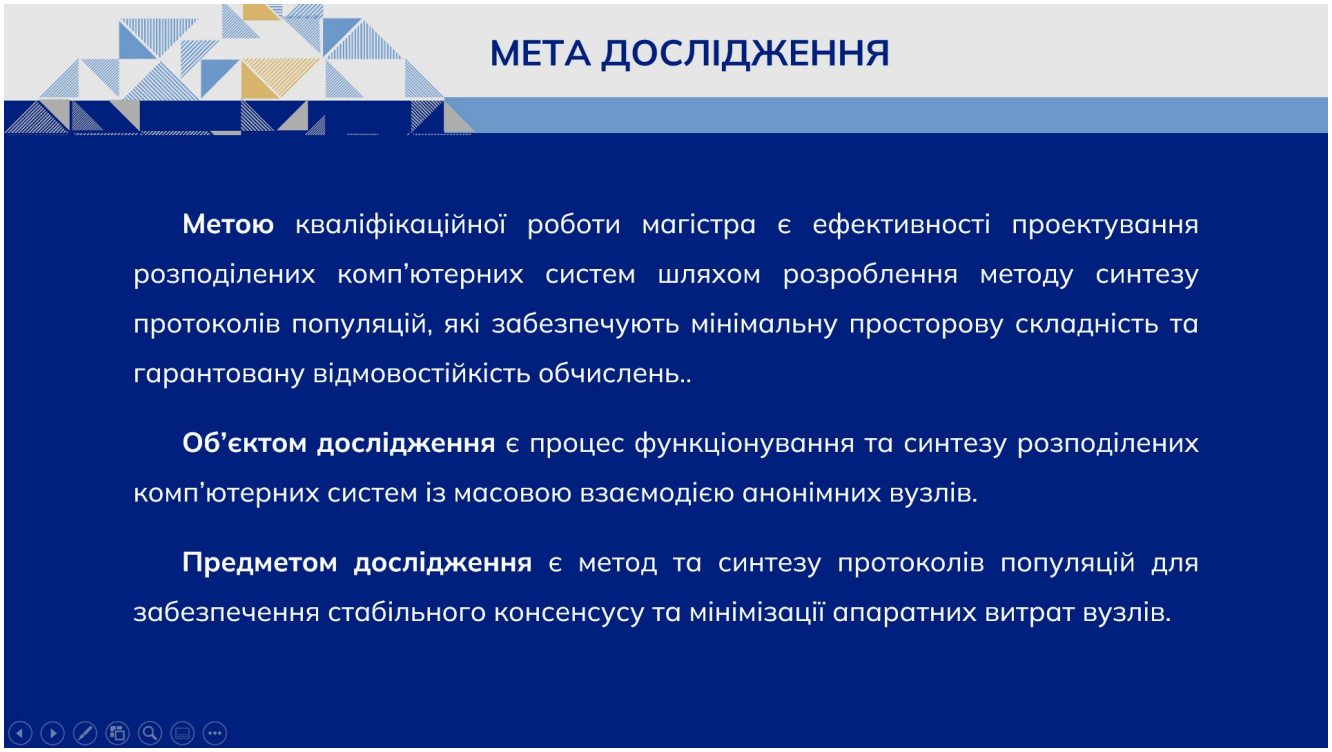
15. Hughes L., Webb J., Morgan H., Song M. *GitOps for Continuous Deployment in Cloud-Native Infrastructure*. 2024. 5 p. Available at: https://www.researchgate.net/publication/392163663_GitOps_for_Continuous_Deployment_in_Cloud-Native_Infrastructure (accessed: March 3, 2026).

16. *Terraform Module Registry — Hierarchical Inheritance*. Scalr Blog. 2021. Available at: https://scalr.com/blog/hierarchical-terraform-module-registry (accessed: March 3, 2026).

Kolomytskyi Denys Коломицький Денис	Master student majoring in Information Systems and Technologies, Khmelnytskyi National University ORCID: 0009-0008-4335-1982 E-mail: kolomitskiy@gmail.com	Магістр спеціальності «Комп'ютерна інженерія», Хмельницький національний університет
Rehida Pavlo Регіда Павло	PhD in Computer Engineering, Associate Professor of the Department of Computer Engineering & Information Systems Department, Khmelnytskyi National University ORCID: 0000-0002-6591-7069 E-mail: pavlo.rehida@gmail.com	Доктор філософії комп'ютерної інженерії, доцент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет
Oksana Onyshko Оксана Онишко	Candidate of Pedagogical Sciences, Associate Professor of the Department of Software Engineering, Khmelnytskyi National University ORCID: 0000-0002-2125-4160 E-mail: Van4o@ukr.net	Кандидат педагогічних наук, доцент кафедри інженерії програмного забезпечення, Хмельницький національний університет
Yuliia Ilchyshyna Юлія Ільчишина	Master student majoring in Information Systems and Technologies, Khmelnytskyi National University E-mail: juliaillchishina@gmail.com	Магістр спеціальності «Комп'ютерна інженерія», Хмельницький національний університет

ДОДАТОК В
(обов’язковий)

Презентація



ЗАДАЧІ ДОСЛІДЖЕННЯ

Поставлена мета досягається розв'язанням таких основних завдань:

- проаналізувати сучасний стан теорії розподілених обчислень та існуючих моделей стохастичної взаємодії в анонімних мережах;
- розробити цільову функцію та алгоритм синтезу для побудови лаконічних протоколів, що здатні обчислювати складні логічні предикати Пресбургера;
- дослідити механізми забезпечення робастності та автономної відмовостійкості синтезованих систем до динамічних змін у чисельному складі популяції, зокрема до раптового видалення або появи нових обчислювальних агентів;
- здійснити дослідження ефективності розробленого методу синтезу на основі програмної симуляції та моделювання процесів досягнення стабільного

НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Наукова новизна отриманих результатів:

- розроблено новий метод синтезу протоколів популяцій, який забезпечує логарифмічну залежність кількості станів від параметрів задачі, що дає можливість суттєво оптимізувати використання пам'яті в ресурсно-обмежених системах;
- удосконалено критерії оцінки відмовостійкості розподілених систем через впровадження поняття локальної робастності для сценаріїв нестабільної кількості учасників.

На основі проведених досліджень розроблено метод автоматизованої генерації правил взаємодії агентів, що гарантує досягнення коректного результату обчислень за оптимальний паралельний час.

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- Актуальність роботи полягає у розробленні системного методу синтезу розподілених систем на основі лаконічних протоколів популяцій, що дають змогу мінімізувати апаратні вимоги до вузлів, кількісті станів, при збереженні повної функціональності та надійності в анонімних мережах

АНАЛІЗ ВІДОМИХ МЕТОДІВ СИНТЕЗУ РКС

Підхід	Суть	Недолік
Автоматний	Система описується як схема станів і переходів	Коли вузлів багато — кількість станів вибухає
Теорія процесів	Система описується формальними мовами	Складно знайти реалізацію, що відповідає специфікації
Темпоральні логіки	Специфікація бажаних властивостей через формули	Іноді задача взагалі нерозв'язна
Paxos/Raft/BFT	Класичні алгоритми голосування між вузлами	Вимагають потужних вузлів, не підходять для ресурсно-обмежених
Протоколи популяцій	Прості правила між парами агентів	

Порівняння конструкцій протоколів популяцій

Протокол	Пам'ять (станів)	Швидкість	Зупиняється після результату?	Стійкий до збоїв?
Ptok	Багато — O(k)	Швидко — O(n)	Так	Слабо
Ptok2	Мало — O(log k)	Трохи повільніше	Так	Слабо
Ptiny	Дуже мало — O(log log k)	Повільно — $O(n^2)$	Ні	Добре
Ppresburger 2020	Мало	Дуже повільно — exp(n)	Так	Слабо
Ppresburger 2024	Мало	Швидко — O(n)	Ні	Слабо

Просторова складність

k (поріг)	Підхід Ptok	Мій метод	Економія
8	8 станів	**7**	~так само
128	128 станів	**15**	у 8 разів менше
1 024	1 024 стани	**21**	у 48 разів менше
1 000 000	**1 000 000 станів**	**41**	у **24 000 разів менше**

- Запропонований метод синтезу на основі протоколів популяцій дозволяє будувати масштабовані та стійкі до масових збоїв розподілені комп'ютерні системи без використання централізованого керування.
- Використання строгої потенційної функції доводить, що розроблені протоколи досягають асимптотично оптимального лінійного часу стабілізації O(n), мінімізуючи витрати ресурсів у обмежених середовищах.
- Завдяки логарифмічному стисненню простору станів, метод зменшує кількість необхідних станів агента для порогових предикатів з лінійної залежності O(k)до оптимальної O(log k).

ВИСНОВКИ

У магістерській роботі за результатами проведених досліджень:

– проаналізовано 5 підходів до синтезу РКС; обґрунтовано вибір протоколів популяцій як найбільш адекватної моделі для анонімних ресурсно-обмежених мереж;

– розроблено новий метод автоматизованого синтезу протоколів популяцій, що забезпечує логарифмічну залежність кількості станів O(log k) від параметрів задачі;

– підтверджено асимптотичну оптимальність: паралельний час O(n) є теоретично мінімальним; для $k=10^6$ потрібно лише 41 стан замість мільйона;

– розроблено програмне забезпечення для синтезу, симуляції та верифікації протоколів; опубліковано наукову статтю у фаховому виданні (ISBN).

ПУБЛІКАЦІЯ

Kolomytskyi D., Rehida P., Onyshko O., Ilchyshyna Y. Generalized Method for Managing the Lifecycle of Terraform Infrastructure Across Multiple Environments. Herald of Khmelnytskyi National University. Technical Sciences. 2026. Vol. 363, no. 2. P. 117–125.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Юлія ІЛЬЧИШИНА

Тема: Метод синтезу комп'ютерних розподілених систем на основі протоколів популяції

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень – ; кількість сторінок записки 78

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано систему профілювання вразливостей при керуванні розумним будинком

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено аналіз децентралізованих моделей та визначено вимоги до синтезу «стислих» протоколів. У другому розділі досліджено теоретико-множинну модель розподіленої комп'ютерної системи у вигляді сукупності однотипних анонімних агентів із обмеженими ресурсами, взаємодія між якими описується правилами протоколів популяцій. У третьому розділі запропоновано модульний метод синтезу протоколів із логарифмічним стисненням простору станів. У четвертому розділі виконано комплексний порівняльний аналіз умов досягнення стабільного стану та доведено, що простота міжагентних взаємодій забезпечує високу масштабованість і стійкість системи до втрати окремих елементів.

4. Позитивні сторони роботи: Запропонований декомпозиційний метод модульного синтезу протоколів популяцій дозволяє ефективно проектувати повністю децентралізовані, безлідерні системи з анонімними агентами. Особливою перевагою роботи є успішне розв'язання задачі логарифмічного стиснення простору

станів, що дає змогу будувати надзвичайно масштабовані та відмовостійкі мікросистемні мережі в умовах критичних апаратних обмежень їхніх елементів.

5. Негативні сторони роботи: У роботі присутні поодинокі логічні неточності та недостатньо розгорнуті коментарі під час опису стохастичних пуассонівських процесів міжагентної взаємодії. Зазначені зауваження мають локальний характер і не знижують загальну наукову та практичну цінність отриманих результатів

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на високому рівні.

8. Інші зауваження: –

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «відмінно» 90.00 (А)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Едуард Манжос проф каф КН

"19 травня" 2026р.

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Юлія ІЛЬЧИШИНА

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІм-24-2

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Метод синтезу рощподілених комп'ютерних систем на основі протоколів популяції
Автор Юлія ІЛЬЧИШИНА
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д. техн. наук Олен САВЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 4,44% та системою Anti-Plagiarism складає 0%, що, з урахуванням наведених 45к41Кеаобгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

25.04.2026

Завідувач кафедри _____ Підпис — Ольга ПАВЛОВА Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Підпис — Олег САВЕНКО Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Підпис — Олег САВЕНКО Ім'я, ПРІЗВИЩЕ

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Юлія ІЛЬЧИШИНА

Співавтор:

Назва: Метод синтезу розподілених комп'ютерних систем на основі протоколів популяції

Експерт: Олег САВЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:4.44%

Коефіцієнт подібності 2:1.29%

Мікропробіли: 8

Заміна букв: 25

Інтервали: 0

Білі знаки: 6

Дата створення звіту: 2026-05-19 10:14:20.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-19 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 0.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 271650 Назва: МКР Метод синтезу розподілених комп'ютерних систем на основі протоколів популяції Додано в БД: 2026-05-19 Автора: Юлія ІЛЬЧИШИНА Керівники: Олег САВЕНКО Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	153639	1072	1852 (1%)	25 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми