МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова праця
на правах рукопису

ЗАКУТИНСЬКИЙ ІГОР ВОЛОДИМИРОВИЧ

УДК 621.372.621.396

ДИСЕРТАЦІЯ

СИСТЕМА ІНТЕРНЕТУ РЕЧЕЙ ДЛЯ МОНІТОРИНГУ ТА УПРАВЛІННЯ ГРОМАДСЬКИМ ТРАНСПОРТОМ

172 «Електронні комунікації та радіотехніка»
17 «Електроніка та телекомунікації»
Подається на здобуття ступеня доктора філософії

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

_____І.В.Закутинський

Науковий керівник:
Сібрук Леонід Вікторович
доктор технічних наук, професор

Київ – 2023

АНОТАЦІЯ

Закутинський І.В. Система Інтернету речей для моніторингу та управління громадським транспортом. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії з галузі знань 17 «Електроніка та телекомунікації» за спеціальністю 172 «Телекомунікації та радіотехніка». – Національний авіаційний університет, 2023 рік.

Дисертаційна робота присвячена побудові інтелектуальних транспортних систем на основі концепції Інтернету речей, розробці системи моніторингу та управління громадським транспортом з використанням запропонованих методів.

За даними Департаменту економіки та соціальних справ ООН, у 2020 році 56% населення світу проживало в міських районах, і ця кількість зростає щорічно. Зростання населення міських регіонів призводить до збільшення кількості транспортних засобів на дорогах, що в свою чергу призводить до загострення транспортних проблем, зокрема до заторів, затрат на пальне і викидів шкідливих речовин в атмосферу, що має негативний вплив на якість життя і здоров'я мешканців міста. Незалежно від розміру регіону, оптимізація роботи системи громадського транспорту стає важливою задачею, яка визначає ефективність роботи транспортної системи та міста в цілому. У цьому контексті концепція Інтернету речей (IoT) стає однією з найбільш перспективних технологій для розробки нових та вдосконалення вже існуючих систем громадського транспорту. IoT надає можливість побудувати інтелектуальні мережі, в якій різні об'єкти транспортної інфраструктури можуть обмінюватися даними та взаємодіяти з операторами та пасажирами в режимі реального часу. Це в свою чергу надає можливості для моніторингу та аналізу даних, а отже і можливості для створення більш безпечних, ефективних та зручних систем громадського транспорту. Тому, розробка інтелектуальних систем громадського транспорту, а також наукове обґрунтування методів їх побудови є актуальною темою дослідження.

У вступі наведено мету та завдання дисертаційного дослідження, а також обґрунтовано актуальність даної теми. Також визначено наукову новизну, серед яких 4 нових наукових методи та 1 вдосконалений, сформульовано практичне значення отриманих результатів. Продемонстровано зв’язок дослідження з науковими темами. Крім того, надано інформацію про 13 наукових публікацій автора, серед яких 7 статей, а також апробацію результатів роботи на 5-ти науково-технічних конференціях.

В першому розділі проведено аналіз поточного стану розвитку інтелектуальних систем для моніторингу та управління громадським транспортом. Також проведено аналіз наукових методів побудови систем Інтернету речей в цілому, а також в контексті їх застосування для розробки інтелектуальних систем громадського транспорту.

На основі проведеного аналізу виділено основні тренди наукових досліджень, а також ряд невирішених проблем та напрямків які потребують розробки нових рішень, або вдосконалення вже існуючих. А саме: вибір та обґрунтування технологій передачі даних, побудова оптимальної архітектури телекомунікаційної мережі, розробка методів моделювання та симуляції поведінки IoT системи, розробка методів для обробки та збереження даних, розробка методів для підвищення ефективності використання обчислювальних ресурсів. Проаналізувавши переваги та недоліки комерційних рішень, було виділено основні функціональні компоненти, які мають бути реалізовані в розробленій системі, а також запропоновано її загальну схему.

На основі проведеного аналізу, було сформовано основне завдання дисертаційного дослідження, а також ряд науково-технічних задач, які необхідно виконати для побудови досліджуваної системи. Крім того, розроблено методику проведення дослідження, в якій сформульований алгоритм виконання вищенаведених задач.

В другому розділі розробляється структура телекомунікаційної мережі, яка має забезпечити передачу даних в досліджуваній інтелектуальні системі громадського транспорту. На першому етапі виділяються інформаційні потоки

даної мережі. Після цього аналізуються існуючі технології та протоколи передачі даних, та обґрунтовується оптимальний вибір для даної системи. Обґрунтування проводиться на основі проведених науково - практичних експериментів. Зокрема, для дослідження бездротових технологій передачі даних LTE та NB-IoT проводиться експеримент в міських умовах, де визначається вплив рівня сигналу на параметри передачі даних, та енергоспоживання системи. Для дослідження протоколів програмного рівня, розроблено середовище тестування, в якому симулюються різні сценарії поведінки IoT мережі.

На основі визначеної структури телекомунікаційної системи, а також обґрунтованих технологій передачі даних розроблено її програмну реалізацію. В основі реалізації покладена концепція розподіленої мікросервісної архітектури. Для оцінки впливу архітектури системи, а також конфігурації її окремих компонентів на пропускну здатність та загальну продуктивність, розроблено математичну модель з метою моделювання функціонування IoT системи на основі розподіленої архітектури.

У третьому розділі дослідження проводиться розробка системи збереження та обробки даних. Спочатку було сформульовано основні завдання даної системи, а також побудовано її функціональну схему та архітектуру. На наступному етапі розглядалися проблеми забезпечення надійності, масштабованості системи, а також проблеми оптимального використання обчислювальних ресурсів. Для розв’язання вищенаведених проблем було розроблено та апробовано комплекс методів та алгоритмів, які на них базуються. А саме: метод визначення оптимальної кількості обчислювальних контейнерів в розподілених системах Інтернету речей, а також метод балансування навантаження в розподілених IoT системах на основі багатопараметричного моніторингу. Далі було розглянуто можливість застосування методів нейронних мереж для обробки даних телекомунікаційної мережі в рамках запропонованої системи збереження та обробки даних. На основі розробленого алгоритму формування даних телекомунікаційної мережі для навчання нейронних мереж,

отримано моделі для прогнозування пасажиропотоку та визначення ймовірності виникнення ДТП.

В четвертому розділі проводиться реалізація інтелектуальної системи громадського транспорту на основі розроблених методів, а також виконується її тестування та оцінка. На першому етапі розроблено загальну архітектуру системи, сформульовано її основні модулі та зв’язки між ними. На основі розробленої архітектури системи виконано програмну реалізацію її компонентів. Крім того, розроблено алгоритм процесу розгортання програмного забезпечення на серверах. Також в даному розділі проаналізовано та обґрунтовано відповідність розробленої системи основним стандартам захисту інформації.

Для проведення оцінки ефективності розробленої системи, а також її порівняння з існуючими комерційними рішеннями запропоновано методику тестування. Суть даної методики полягає у відтворенні навантаження, яке створює транспортна система міста Києва, для двох систем – розробленої на основі запропонованої архітектури та методів, а також системи на основі існуючих комерційних рішень. Оцінка ефективності вищенаведених систем проводилась по ключових характеристиках робастності системи, які були виділенні в завданні даного дослідження. Результати оцінки свідчать про переваги запропонованої IoT системи у порівняні з аналогами.

Ключові слова: інтернет речей, громадський транспорт, телекомунікаційна мережа, xG системи мобільного зв'язку, 5G, LTE, стільникові мережі, програмно визначене радіо, моніторинг, інтелектуальні системи, системи управління, вимірювання сигналів, бездротові технології, мобільний зв’язок, розподілені системи, балансування навантаження, мережеві системи, обчислювальні контейнери, протоколи передачі даних, сенсори, датчики, глибоке навчання, машинне навчання, нейронні мережі, інформаційні потоки, передача інформації, захист інформації, програмне забезпечення, хмарні обчислення, обробка даних, аналіз даних, виявлення аномалій, симуляційні моделі, автоматизована система, мережева архітектура, індустріальні системи, програмні

алгоритми, моделі оцінки ефективності, M2M комунікації, електронні система, радіоелектронні системи, оцінка ефективності, безпека даних, кібербезпека.

ABSTRACT

Zakutynskyi I.V. IoT System for Monitoring and Management Public Transport. - Qualifying scientific topic as a manuscript.

Dissertation for the degree of Doctor of Philosophy in the field of knowledge 17 "Electronics and Telecommunications", specialty 172 "Electronics communications and Radio Engineering". - National Aviation University, 2023.

The dissertation focuses on the development of intelligent transport systems based on the Internet of Things (IoT) concept and the creation of a system for monitoring and managing public transport using the methods proposed.

According to the United Nations Department of Economic and Social Affairs, in 2020, 56% of the global population resided in urban areas, and this number continues to grow annually. The population growth in urban regions leads to an increase in the number of vehicles on the roads, exacerbating transportation problems such as traffic congestion, fuel costs, and emissions of harmful substances into the atmosphere. These factors have a negative impact on the quality of life and the health of city residents. Regardless of the region's size, optimizing the operation of the public transport system becomes a crucial task that significantly influences the efficiency of the transport system and the city as a whole. In this context, the Internet of Things (IoT) concept emerges as one of the most promising technologies for enhancing existing public transport systems and developing new ones. IoT offers the opportunity to create intelligent networks where various elements of the transportation infrastructure can exchange data and interact with operators and passengers in real-time. This, in turn, facilitates data monitoring and analysis, offering opportunities to create safer, more efficient, and convenient public transport systems.

Therefore, the development of intelligent public transport systems and the scientific justification of their construction methods represent a relevant research topic.

The introduction outlines the research's objectives and tasks, substantiates the relevance of the topic, and identifies the scientific novelty, including four new scientific methods and one improved method. It also emphasizes the practical significance of the obtained results and establishes a connection with related scientific topics. Additionally, the author's 13 scientific publications, including seven articles, and the presentation of work results at five scientific and technical conferences are provided.

The first section analyzes the current state of intelligent systems for monitoring and managing public transport. It also explores scientific methods for building IoT systems, both in general and within the context of their application to intelligent public transport systems.

Based on the analysis, the main trends in scientific research are identified, along with several unsolved problems and areas requiring new solutions or improvements. These include the selection and justification of data transmission technologies, the development of optimal telecommunication network architecture, methods for modeling and simulating IoT system behavior, data processing and storage methods, and techniques for optimizing computing resource utilization. After analyzing the advantages and disadvantages of commercial solutions, the main functional components to be implemented in the developed system are selected, and its general scheme is proposed.

Building upon this analysis, the main task of the dissertation research is formulated, along with scientific and technical tasks necessary for constructing the studied system. A research methodology is developed, including an algorithm for performing these tasks.

In the second section, the telecommunications network structure is developed to ensure data transmission in the intelligent public transport system under study. Initially, the information flows within this network are defined. Subsequently, existing technologies and data transmission protocols are analyzed, and the optimal choices for this system are justified based on scientific and practical experiments. These

experiments include a study of LTE and NB-IoT wireless data transmission technologies in the urban environment, where the influence of signal level on data transmission parameters and system energy consumption is determined. A test environment for simulating various IoT network behavior scenarios is created to study software-level protocols.

Building upon the determined telecommunications system structure and well-founded data transmission technologies, software implementation is developed based on the concept of distributed microservice architecture. To evaluate the impact of system architecture and individual component configurations on throughput and overall performance, a mathematical model is developed to simulate the functioning of an IoT system based on distributed architecture.

In the third section of the study, the data storage and processing system is developed. Initially, the primary tasks of this system are formulated, and its functional scheme and architecture are designed. Subsequently, issues related to ensuring system reliability, scalability, and optimal utilization of computing resources are addressed. A set of methods and algorithms, including a method for determining the optimal number of computing containers in distributed IoT systems and a method for load balancing based on multi-criteria monitoring, are developed and tested to solve these problems. Additionally, the potential application of neural network methods for processing telecommunication network data within the proposed data storage and processing system is explored. Using the developed algorithm for forming telecommunication network data for training neural networks, models for predicting passenger traffic and determining the probability of accidents are obtained.

In the fourth chapter, an intelligent public transport system is implemented based on the developed methods, and its testing and evaluation are conducted. The initial stage involves developing the general architecture of the system, formulating its main modules, and establishing connections between them. Software implementation of the system components is performed based on the developed architecture. Furthermore, an algorithm for the software deployment process on servers is created.

This section also includes an analysis and justification of the compliance of the developed system with basic information protection standards.

To assess the effectiveness of the developed system and compare it with existing commercial solutions, a testing methodology is proposed. This methodology involves reproducing the load generated by the transport system of Kyiv using two systems: one developed based on the proposed architecture and methods, and the other based on existing commercial solutions. The evaluation of these systems' efficiency is conducted based on key characteristics of system robustness identified in the study. The results of the evaluation demonstrate the advantages of the proposed IoT system compared to its counterparts.

Keywords: internet of things (iot), public transport, telecommunication network, xG mobile communication systems, 5G, LTE, cellular networks, software defined radio, monitoring, intelligent systems, control systems, signal measurement, wireless technologies, mobile communication, distributed systems, load balancing, network systems, computing containers, data transfer protocols, sensors, sensors, deep learning, machine learning, neural networks, information flows, information transfer, information protection, software, cloud computing, data processing, data analysis, anomaly detection, simulation models, automated system, network architecture, industrial systems, software algorithms, performance evaluation models, M2M communications, electronic system, radio electronic systems, performance evaluation, data security, cyber security.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Sibruk, L., & Zakutynskyi, I. (2022). Recurrent Neural Networks for Time Series Forecasting. Choosing the best Architecture for Passenger Traffic Data. In Electronics and Control Systems (Vol. 2, Issue 72, pp. 38–44). National Aviation University.

2. Zakutynskyi, I., & Rabodzei, I. (2022). Microservice Communication for IoT-based Systems. Architecture Review and Performance Test. In Electronics and Control Systems (Vol. 4, Issue 74, pp. 73–78). National Aviation University.

3. Zakutynskyi, I., Sibruk, L., & Kokarieva, A. (2023). IoT System for Monitoring and Managing Public Transport Data. In WSEAS TRANSACTIONS ON SYSTEMS (Vol. 22, pp. 242–248). World Scientific and Engineering Academy and Society (WSEAS).

4. Zakutynskyi, I. (2023). Finding the Optimal Number of Computing Containers in IoT Systems: Application of Mathematical Modeling Methods. In Electronics and Control Systems (Vol. 2, Issue 76, pp. 9–14). National Aviation University.

5. Закутинський, І. (2022). ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ПЕРЕДБАЧЕННЯ ТА АНАЛІЗУ ДОРОЖНЬО-ТРАНСПОРТНИХ ПРИГОД. In Наука і техніка сьогодні (Issue 13(13)). Ukrainian Assembly of Doctors of Science in Public Administration

6. Zakutynskyi, I., & Rabodzei, I. (2023). IoT system architecture for monitoring and analyzing public transport data. Multidisciplinary Science Journal, 5, 2023.

7. Zakutynskyi, I., & Sibruk L, Rabodzei, I. (2023). Performance evaluation of the cloud computing application for IoT-based public transport systems. Eastern-European Journal of Enterprise Technologies, 4, pp. 6-13.

8. Zakutynskyi, I., Rabodzei, I., Burmakin, S., Kalishuk, O., Nebylytsia, V. (2023).Improving a procedure of load balancing in distributed IoT systems. Eastern-European Journal of Enterprise Technologies, 5 (2 (125)).

9. Zakutynskyi, I., Sibruk, L. Recurrent neural networks for passengers traffic data forecasting. Architecture overview and comparison. (2022). Aviation in the XXI-st century. Safety in Aviation and Space Technologies: Proceeding of The Tenth World Congress (Kyiv, 28 –30 September 2022), 2022. P. 2.1.9-2.1.13.

10. Zakutynskyi, I. Neural networks for road accident predictions and analysis. AUTOMATION, COMPUTER-INTEGRATED TECHNOLOGIES, AND PROBLEMS OF ENERGY EFFICIENCY IN INDUSTRY: Conference proceeding, Kropyvnytskyi, 10-11 November 2022. P. 166-168.

11. Закутинський І.В. IoT система для моніторингу та аналізу даних громадського транспорту. Інноваційні технології розвитку та ефективності функціонування автомобільного транспорту: Збірник матеріалів Міжнародної науково-практичної інтернет-конференції, Кропивницький, 17 – 19 листопада 2022 р. С. 34-37.

12. Zakutynskyi, I., & Rabodzei, I. IoT system architecture for monitoring and analyzing public transport data. 4th International Conference on Multidisciplinary Innovation in Academic Research (ICMIAR-2023): Conference proceedings, 17th & 18th March 2023, Chennai, India. P. 18.

13. Zakutynskyi, I., Sibruk, L. Security framework for IoT-Based public transport systems: A comprehensive analysis and design. International Conference on Research in Engineering, Technology and Science (ICRETS): Abstract Book July 6-9, 2023 - Budapest, Hungary. P. 37.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IoT	Internet of Things (Інтернет речей)
IPTS	Intelligent Public Transport System (Інтелектуальні системи міського транспорту)
ITC	Інтелектуальні транспортні системи
AVL	Automatic Vehicle Location (Система локації транспортного засобу)
PIS	Passenger Information System (Інформаційна пасажирська система)
EFC	Electronic Fare Collection (Електронна система оплати проїзду)
TMS	Traffic Management System (Система управління рухом)
API	Application Programming Interface (Прикладний програмний інтерфейс)
MQTT	Message Queue Telemetry Transport (Протокол телеметрії)
HTTP	Hypertext Transfer Protocol (Протокол передачі тексту)
CoAP	Constrained Application Protocol (Спеціалізований протокол веб-передачі для систем Інтернету речей)
gRPC	Google Remote Procedure Call (Система віддаленого виклику процедур)
БД	База даних
NB-IoT	Narrow Band Internet of Things (Стандарт стільникового зв'язку для пристроїв телеметрії з низькими обсягами обміну даними)
ОС	Операційна система
ПЗ	Програмне забезпечення

MILP	Mixed-Integer Linear Programming (Змішане цілочисельне лінійне програмування)
LB	Load Balancing (Балансування навантаження)
LSTM	Long short-term memory (Довга короткочасна пам'ять)
GRU	Gated Recurrent Units (Керований рекурентний блок)
IAM	Identity and Access Management (Управління обліковими даними)
JWT	JSON Web Token (Стандарт токена доступу)
TDD	Test Driven Development (Методологія тестової розробки)
LXC	Linux Containers (Система контейнеризації)
CI/CD	Continuous integration, Continuous delivery (Методологія неперервної інтеграції та неперервної доставки)
DNS	Domain name system (Система доменних імен)
SQL	Structured Query Language (Мова структурованих запитів)
NOSQL	non-SQL (нереляційна структура даних)
ДТП	Дорожньо-транспортна пригода

ЗМІСТ

ВСТУП

Урбанізація – це процес, коли все більша і більша частина населення країни переїжджає жити в міські регіони. Це відбувається в усьому світі, і це відбувається дуже швидко. Більше половини населення світу зараз живе в містах, і за прогнозом Світового банку до 2050 року цей показник зросте майже до 70% [1]. В Україні понад 70% населення проживає в містах [2], що робить її однією з найбільш урбанізованих країн Європи. Найбільшими містами України є Київ, Харків, Одеса, Дніпро. Більшість сучасних міст стикаються з такими проблемами, як старіння інфраструктури, забрудненість повітря через перенасичення персональним транспортом, а також нерозвинені та неоптимізовані системи громадського транспорту. Громадський транспорт є критично важливими компонентом міської інфраструктури, який значною мірою формує якість життя в сучасному місті.

Системи громадського транспорту — це системи, які забезпечують пересування людей у межах міста чи регіону за допомогою різних видів транспорту, включаючи автобуси, потяги та трамваї, та ін. Розумний громадський транспорт може зіграти значну роль у вирішенні сучасних викликів урбанізації. Використовуючи аналіз даних в реальному часі та розширену аналітику, ці системи можуть оптимізувати маршрути та розклади руху, скоротити час очікування та підвищити якість обслуговування пасажирів. Ще однією перевагою розумного громадського транспорту є те, що він може покращити доступність існуючого транспорту для всіх мешканців, у тому числі людей з обмеженими можливостями чи обмеженою мобільністю. Ці системи можуть надавати інформацію в реальному часі про варіанти громадського транспорту, включаючи доступні маршрути та послуги, в тому числі інклюзивні, такі як низькі підлоги, пандуси та автоматичні оголошення.

Це позитивно вплине на привабливість громадського транспорту для жителів сучасного міста, а отже допоможе зменшити кількість приватних автомобілів на дорозі, що в свою чергу дозволить зменшити затори та

забруднення повітря. Тому на сьогоднішній день проблеми побудови та оптимізації Інтелектуальної транспортної системи є надзвичайно актуальними.

Аналіз результатів досліджень, проведених в напрямку методів розробки та побудови Інтелектуальних транспортних систем, показує, що наразі питання наукового обґрунтування та побудови до кінця не є вирішеними. Особливо це стосується такого важливого компоненту, як телекомунікаційні мережі, які забезпечують зв'язок в режимі реального часу між транспортними засобами, інфраструктурою та центрами управління. Також не до кінця дослідженими є проблеми обробки та збереження даних, для подальшого аналізу та моніторингу.

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

Тема дисертаційної роботи відповідає Національній транспортні стратегії України на період до 2030 року, від 30 травня 2018 р. № 430-р. Крім того, тематика дисертаційного дослідження безпосередньо пов'язана з рішенням науково-технічних завдань, які відповідають положенням Закону України Про пріоритетні напрями розвитку науки і техніки (стаття 3, п.2) від 11 липня 2001; Закону України «Про пріоритетні напрями інноваційної діяльності в Україні» (стаття 4, п.2 і п.7) від 5 грудня 2012; «Концепції розвитку телекомунікацій в Україні» затвердженої розпорядженням Кабінету Міністрів України № 316-р від 7 червня 2006.

В ході проведення наукового дослідження були враховані положення Закону України «Про телекомунікації», Закону України «Про використання радіочастотного ресурсу», Закону України «Про інноваційну діяльність», Постанови Кабінету Міністрів України «Про затвердження Національної таблиці розподілу смуг радіочастот України» та Постанови Кабінету Міністрів України «Про затвердження Плану використання радіочастотного ресурсу України».

Мета і завдання дослідження.

Метою дисертаційної роботи є підвищення ефективності методів побудови інтелектуальних транспортних систем, а також розробка на основі них системи Інтернету речей для моніторингу та управління громадським транспортом.

Для цього сформульовано комплекс наступних науково – технічних задач:

1. Провести аналіз поточного стану існуючих Інтелектуальних систем громадського транспорту, а також методів побудови. Визначити їх недоліки та обмеження.

2. Розробити загальної архітектури інтелектуальної системи громадського транспорту, виділення основних інформаційних потоків, розробка методів та алгоритмів збереження та обробки даних.

3. Розробити метод та алгоритм побудови телекомунікаційної мережі системи Інтернету речей. На основі нього розробити структуру мережі, яка дозволить підтримувати передачу даних в режимі реального часу та забезпечить зв’язок між різними компонентами інтелектуальної системи громадського транспорту.

4. Розробити методи для визначення оптимальної кількості обчислювальних ресурсів в системах Інтернету речей з динамічним навантаженням.

5. Розробити метод підвищення ефективності балансування навантаження в розподілених системах Інтернету речей, а також запропонувати модель оцінки рівномірності розподілу навантаження.

6. Розробити методику оцінки продуктивності запропонованої інтелектуальної системи громадського транспорту на основі симуляційних експериментів.

7. Надати практичні рекомендації щодо прийняття та впровадження запропонованої системи у конкретних містах та регіонах.

Об'єкт дослідження – процес побудови Інтелектуальної системи для моніторингу та правління міським транспортом на основні концепції Інтернету речей.

Предмет дослідження – моделі та методи розробки Інтелектуальних транспортних систем та систем Інтернету речей.

Наукова новизна отриманих результатів полягає в тому, що:

1. **Вперше** запропонований метод балансування навантаження в розподілених системах Інтернету речей, який на відмінну від існуючих виконує балансування навантаження в мережі на основі багатопараметричного моніторингу стану активних обчислювальних контейнерів, дозволяє підвищити ефективність балансування навантаження на 40% - 65%.

2. **Вперше** запропонований метод визначення оптимальної кількості обчислювальних ресурсів. Даний метод реалізований на основі розробленої математичної моделі, яка на відмінну від існуючих враховує динамічне навантаження, яке генерується IoT пристроєм, що дозволяє покращити ефективність серверних обчислень при нерівномірному навантаженні.

3. **Вперше** розроблено математичну модель для моделювання підсистеми збереження та обробки даних в системах Інтернету речей. Розроблена математична модель дозволяє оцінити вплив архітектури та конфігурації компонентів підсистеми на її пропускну здатність, а також оцінити загальну ефективність системи на основі функцій розподілу швидкості обробки даних.

4. **Отримав подальший розвиток** метод прогнозування часових рядів за рахунок попередньої обробки вхідних даних телекомунікаційної мережі за запропонованим алгоритмом на основі LSTM архітектури для прогнозування пасажиропотоку міського транспорту і виявлення аномалій, а також метод рекурентних нейронних мереж для оцінки ризиків виникнення дорожньо-транспортних пригод (ДТП).

Практичне значення одержаних результатів. Практично вагомими вважаються такі результати:

1. Запропоновано методику визначення оптимальної технології передачі даних в телекомунікаційній IoT мережі, який полягає в проведенні комплексу експериментальних досліджень передачі даних між пристроєм Інтернету речей та обчислювальним сервером. На основі якого, проведено дослідження NB-IoT стандарту для побудови телекомунікаційної мережі в

інтелектуальних транспортних системах. Дослідження дозволили оцінити переваги та недоліки даного стандарту в порівнянні з класичним LTE з'єднанням в міських умовах.

2. Розроблено програмний комплекс для дослідження впливу протоколів програмного рівня на швидкість передачі та пропускну здатність в мережі системи Інтернету речей.

3. Розроблено програмний продукт згідно математичної моделі процесу функціонування IoT мережі, який дає можливість оцінити можливе навантаження, та правильно спроектувати серверну архітектуру та конфігурацію системи.

4. Розроблено алгоритм автоматичного розгортання розподіленої архітектури IoT мережі для інтелектуальної системи міського транспорту.

5. Розроблено алгоритм оцінки продуктивності системи із застосуванням програмних емуляторів IoT пристроїв. Виконано програмну реалізацію емуляторів, а також середовища для проведення тестування, на основі запропонованої моделі.

6. Запропоновано алгоритм автоматичного формування вхідних даних для навчання нейронних мереж на основі телеметрії IoT пристроїв. Виконано програмну реалізацію алгоритму, а також на основі сформованих даних виконано навчання моделей нейронних мереж.

Особистий внесок здобувача.

Основні положення та результати дисертаційної роботи отримані автором самостійно. Автор виконав усі теоретичні та практичні дослідження, що становлять основу дисертаційної роботи. В опублікованих роботах у співавторстві, згідно з списком опублікованих праць за темою дисертації (с. 11-12), здобувачу належать такі результати: [1], [8] – запропоновано алгоритм формування даних телекомунікаційної мережі для навчання нейронних мереж, розроблено моделі для прогнозування пасажиропотоку, виконано програмну реалізацію; [2] – запропоновано методику оцінювання продуктивності

протоколів програмного рівня в системах Інтернету речей, розроблено середовище тестування, проведено практичний експеримент оцінки протоколів передачі даних; [3] - розроблено архітектуру системи управління та керування даними телекомунікаційної мережі громадського транспорту, виконано програмну реалізацію запропонованих методів, проведено аналіз отриманих результатів; [6], [11] - розроблено структуру інтелектуальної системи громадського транспорту, запропоновано методику оцінки пропускної здатності, проведено практичні експерименти; [7] - розроблено методику оцінки методів хмарних обчислень для транспортних систем на основі концепції Інтернету речей, виконано програмну реалізацію емулятора IoT модуля, виконано програмну реалізацію системи збереження та обробки даних для Інтелектуальної системи громадського транспорту; [12] - проведено аналіз методів захисту інформації в системах Інтернету речей, запропоновано метод, та розроблено програмне забезпечення для оцінки вразливості системи.

Апробація результатів дисертації. Основні результати дисертаційної роботи були представлені та обговорені на семи міжнародних науково-технічних та науково-практичних конференціях:

1. Aviation in the XXI-st century. Safety in Aviation and Space Technologies: Proceeding of The Tenth World Congress. Kyiv, 28 –30 September 2022.

2. AUTOMATION, COMPUTER-INTEGRATED TECHNOLOGIES, AND PROBLEMS OF ENERGY EFFICIENCY IN INDUSTRY. Kropyvnytskyi, 10-11 November 2022.

3. Інноваційні технології розвитку та ефективності функціонування автомобільного транспорту: Збірник матеріалів Міжнародної науково-практичної інтернет-конференції, Кропивницький, 17 – 19 листопада 2022 р.

4. 4th International Conference on Multidisciplinary Innovation in Academic Research (ICMIAR-2023), 17th & 18th March 2023, Chennai, India.

5. International Conference on Research in Engineering, Technology and Science (ICRETS). July 6-9, 2023 - Budapest, Hungary.

6. 12th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. 7-9 September 2023. Dortmund University of Applied Sciences and Arts, Dortmund, Germany.

7. VII Міжнародна науково-практична конференція: Прикладні системи та технології в інформаційному суспільстві. Вересень 2023. Київський національний університет імені Тараса Шевченка, Київ, Україна.

Додатково практичні результати даного дослідження були апробовані в рамках круглого столу «Європейський досвід для підвищення стійкості критично важливих об'єктів в Україні, Проєкт №: 101085825 - ERASMUS-JMO-2022-MODULE.», а також хакатону «Volyn Cybersecurity Summer Training Camp 2023».

Публікації. Основні результати дисертаційного дослідження були опубліковані в 12 наукових працях. Зокрема 7 статей в наукових журналах, 3 з яких проіндексовані міжнародними наукометричними базами даних. Також отримані результати були представлені в 5 публікаціях у матеріалах міжнародних науково-технічних та науково-практичних конференцій.

Структура та обсяг дисертації.

Дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, додатків і містить 84 рисунки, 26 таблиць, 15 сторінок додатків. Список використаних джерел містить 108 найменувань і займає 16 сторінок. Загальний обсяг роботи складає 184 сторінки.

РОЗДІЛ 1. СТАН РОЗВИТКУ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ ДЛЯ МОНІТОРИНГУ ТА УПРАВЛІННЯ ГРОМАДСЬКИМ ТРАНСПОРТОМ

Інтелектуальні транспортні системи (ІТС) все частіше застосовуються в багатьох містах по всьому світу для вирішення проблем, пов’язаних із заторами, забрудненням повітря, безпекою руху а також підвищення рівня комфорту пасажирів. У цьому розділі буде розглянуто поточний стан існуючих інтелектуальних транспортних систем та їхній вплив на розвиток міст та регіонів.

Азійські міста були в авангарді впровадження ІТС-технологій, що зумовлено високою щільністю населення та швидкою урбанізацією. Такі міста, як Токіо, Сінгапур і Сеул та Пекін, запровадили комплексні системи, які включають моніторинг руху в реальному часі, розумне паркування та автоматизований громадський транспорт. Наприклад, у Сінгапурі є повністю інтегрована система ІТС, яка включає мережу датчиків і камер для моніторингу транспортного потоку та коригування дорожніх сигналів у режимі реального часу. Впровадження такої системи призвело до скорочення проведеного пасажиром часу в дорозі в час-пік на 12% [3].

Такі міста, як Нью-Йорк, Лос-Анджелес і Торонто, запровадили інтелектуальні системи керування дорожнім рухом, які використовують дані в режимі реального часу для коригування сигналів світлофора та зменшення заторів [4-7]. Міста Південної Америки також починають застосовувати ІТС-технології, що спонукає до вирішення проблеми заторів і забруднення [8-11]. Такі міста, як Сантьяго та Богота, запровадили системи швидкого автобусного транспорту, які забезпечують швидкий і надійний громадський транспорт. Ці системи скоротили час у дорозі та розширили доступ жителів до громадського транспорту.

1.1 Об’єкти дослідження

Сучасне суспільство надає особливу увагу розвитку сталої та ефективної системи транспорту, спроможної задовольняти потреби міського населення. Наземний громадський транспорт є об’єктом даного дослідження. IoT системи, є основою концепції «Розумного міста». Тому другим об’єктом дослідження є система Інтернету речей, та можливості її застосування для побудови інтелектуальних систем громадського транспорту. Останнім об'єктом дослідження будуть методи побудови систем Інтернету речей. Це включає в себе розгляд архітектурних рішень, протоколів зв'язку, способів збору та аналізу даних, а також розробку оптимальних стратегій взаємодії між "розумними" пристроями.

Рис. 1.1. Об’єкти дослідження

1.1.1. Громадський транспорт як об’єкт дослідження

Наземний громадський транспорт, що включає автобуси, трамваї, метро, тролейбуси та інші види громадського перевезення, є складною системою, що забезпечує рух пасажирів в сучасних міських регіонах та на прилеглих територіях. Об’єктами дослідження є не лише транспортні засоби самі по собі, а й вся інфраструктура, пов’язана з їх рухом та обслуговуванням. А саме:

- Моніторинг руху транспорту: Сучасні системи моніторингу дозволяють в реальному часі відстежувати місцезнаходження транспортних

засобів. Це дозволяє збирати дані про рух та зупинки, а також аналізувати часові затримки та пропускну здатність.

- Системи оповіщення та інформування: На основі інтеграції сучасних методів комунікації, можливо передавати пасажирам актуальну інформацію через мобільні додатки, електронні табло на зупинках та інші канали. Це дозволяє пасажирам отримувати оновлену інформацію про розклади, затримки, маршрути та доступність транспорту.
- Моніторинг стану транспорту: Важливим аспектом є відстеження технічного стану транспортних засобів. Системи на базі Інтернету речей дозволяють в режимі реального часу відслідковувати параметри, такі як рівень пального, тиск у шинах, стан двигуна тощо. Це дозволяє вчасно виявляти поломки та планувати профілактичний ремонт.

1.1.2. Системи Інтернету речей як об’єкт дослідження

Системи Інтернету речей є складними інтегрованими структурами, що об’єднують багато пристроїв, датчиків та сенсорів для збору та обміну даними через мережу Інтернет. У контексті даного дослідження IoT систем для громадського транспорту, об’єктом є концепція реалізації таких систем для оптимізації управління та надання якісних послуг пасажирам. В контексті системи Інтернету речей, можна виділити наступні групи, які є об’єктами дослідження:

- Сенсори та датчики: Сенсори та датчики є основними елементами IoT системи, які здійснюють збір даних з оточуючого середовища. У випадку громадського транспорту це можуть бути геолокаційні дані, температура, вологість, вібрація інформація про стан транспортного засобу, тощо.
- Збір та передача даних: Зібрані дані передаються через мережу Інтернет до центральної системи для подальшої обробки та аналізу. Це дозволяє здійснювати моніторинг руху транспорту, відстежувати затримки та виявляти проблеми та аномалії.

- Аналітика та прийняття рішень: Однією з важливих характеристик систем Інтернету речей є можливість аналізу та обробки великих обсягів даних. Застосування алгоритмів машинного навчання та штучного інтелекту дозволяє виявляти закономірності, прогнозувати події та приймати обґрунтовані рішення для покращення та оптимізації роботи системи.

1.1.3. Методи побудови систем Інтернету речей для моніторингу та управління громадським транспортом як об’єкт дослідження

Розробка систем Інтернету речей для громадського транспорту включає в себе використання широкого спектру методів та технологій. В даному дослідження методи побудови є окремим об’єктом дослідження. Ці методи включають в себе використання бездротових комунікацій, хмарних обчислень, а також використання методів машинного навчання та штучного інтелекту для обробки даних.

- Бездротові комунікації: Сучасні системи інтернету речей потребують ефективного способу обміну даними між транспортними засобами, інфраструктурою та центральною системою обробки. Бездротові технології забезпечують можливість збору та передачі даних у реальному часі на значні відстані.
- Хмарні обчислення: Для зберігання, аналізу та обробки великого обсягу даних використовуються хмарні платформи. Це дозволяє забезпечити доступ до інформації з будь-якої точки та робити аналітику в реальному часі.
- Машинне навчання та штучний інтелект: Застосування моделей машинного навчання дозволяє прогнозувати маршрути та рух транспортних засобів на основі зібраних даних. Використання алгоритмів машинного навчання дозволяє автоматизувати процеси аналізу та прийняття рішень. Наприклад, можна розробити моделі для прогнозування руху та затримок, а також оптимізації маршрутів.

1.2. Підсистеми інтелектуальних систем громадського транспорту

Для проведення комплексного аналізу та порівняння існуючих інтелектуальних систем громадського транспорту, необхідно виділити їх основні підсистеми, та визначити основні характеристики. На сьогоднішній день виділяють декілька основних типів таких підсистем, зокрема: система автоматичного визначення місцезнаходження транспортного засобу, система інформування пасажирів, система моніторингу технічного стану транспортного засобу, система управління рухом, система безпеки руху, система електронної оплати проїзду.

Система автоматичного визначення місцезнаходження транспортного засобу. У великих містах важливо ефективно відстежувати та керувати рухом громадського транспорту для забезпечення плавності руху та мінімізації затримок. Дана підсистема (рис. 1.2) дозволяє в реальному часі відстежувати місцезнаходження транспортних засобів, контролювати їх рух, визначати затримки та надавати оперативну інформацію для пасажирів та операторів. Система складається з двох основних компонентів: *сенсори та GPS-приймачі*, які встановлюються на транспортних засобах і забезпечують збір і передачу даних про місцезнаходження, швидкість, напрямок руху тощо, а також *системи бездротового зв'язку*, які забезпечують передачу даних з сенсорів та приймачів на центральний сервер.

На сьогодні існує багато систем автоматичного визначення місця розташування транспортного засобу. Наприклад транспортна система у м.Лондон, Велика Британія - "Transport for London" використовує систему сенсорів та GPS на автобусах та поїздах для відстеження їх руху в реальному часі.

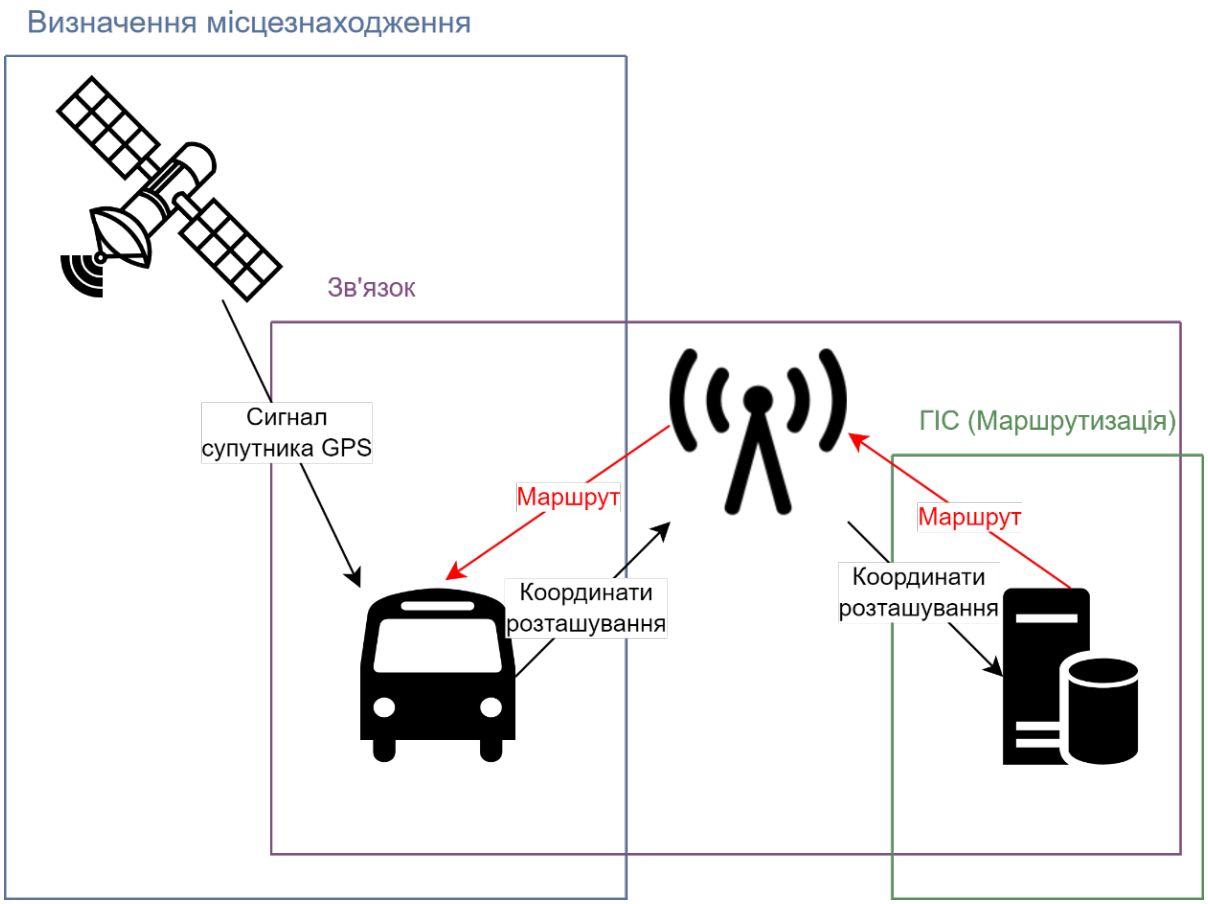


Рис. 1.2. Елементи системи автоматичного визначення місцезнаходження транспортного засобу [12]

Інформація про місцезнаходження передається до центральної системи, яка аналізує дані та регулює рух транспорту для мінімізації затримок та забезпечення точних розкладів.

Система інформування пасажирів. Дані системи (рис. 1.3) надають пасажирам інформацію в режимі реального часу про час прибуття, затримки та іншу інформацію, пов’язану з громадським транспортом. Може використовувати веб-інтерфейси, мобільні додатки, електронні табло на зупинках та інші засоби для передачі інформації пасажирам. На сьогоднішній день є багато прикладів реалізації таких систем. Наприклад система «Київ Цифровий», Київ Україна, яка дозволяє пасажирам отримувати інформацію через мобільний додаток, який надає розклади, затримки та оптимальні маршрути.

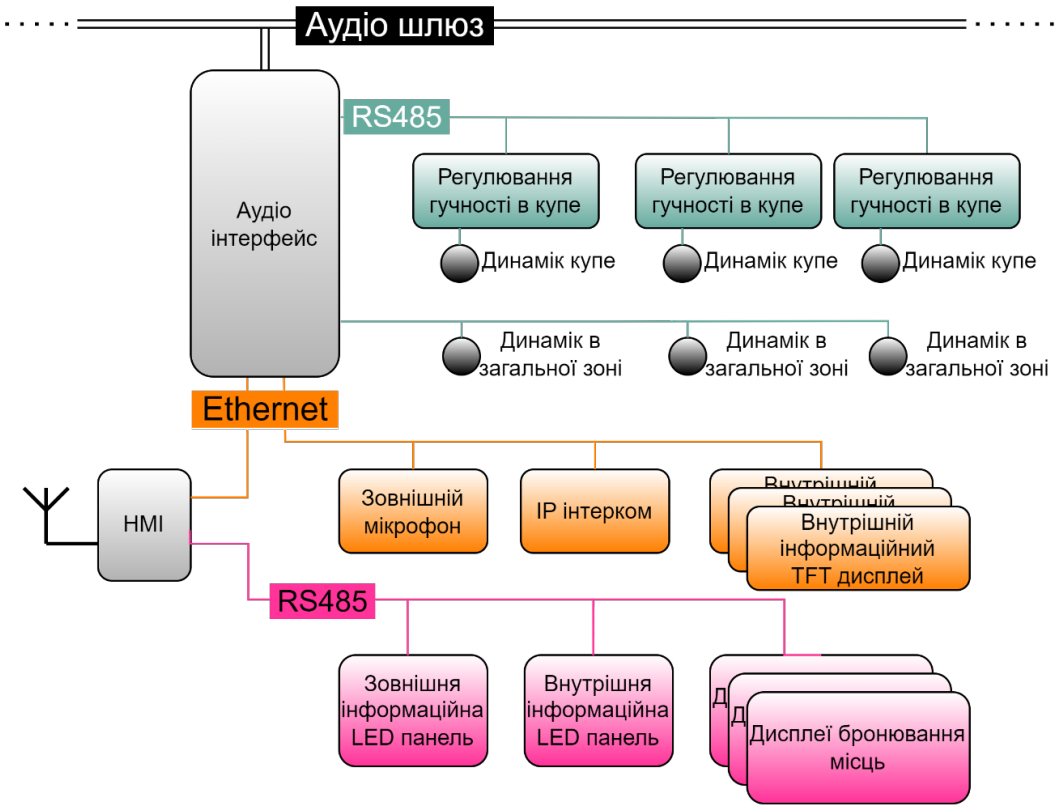


Рис. 1.3. Блок-схема системи інформування пасажирів [13]

Система моніторингу технічного стану транспортного засобу. Дана система виконує моніторинг технічного стану транспортних засобів, для виявлення поломок та несправностей. Основними компонентами даної системи є датчики та сенсори для вимірювання параметрів транспортних засобів, наприклад: тиск в шинах, температура двигуна, рівень пального, а також автоматизовані системи, що виявляють несправності в роботі транспорту та передають інформацію для подальшого обслуговування. Прикладом такої системи, може бути система моніторингу технічного стану автобусів м. Нью-Йорк, США. Дана система виявляє контролює ряд параметрів, таких як: стан гальмівних колодок, низький рівень пального, температура двигуна, тощо, та надсилає автоматичні сповіщення про можливі несправності для забезпечення швидкого реагування.

Система керування рухом. Важливим завданням сучасної транспортної системи є ефективне управління рухом для мінімізації заторів та забезпечення

швидкого та зручного перевезення пасажирів. Дана підсистема (рис. 1.4) використовує алгоритми оптимізації та прогнозування для керування рухом транспорту. Вона враховує інформацію про затримки, велику кількість пасажирів та інші фактори для вирішення питань щодо графіку руху та розподілу транспорту за маршрутами.

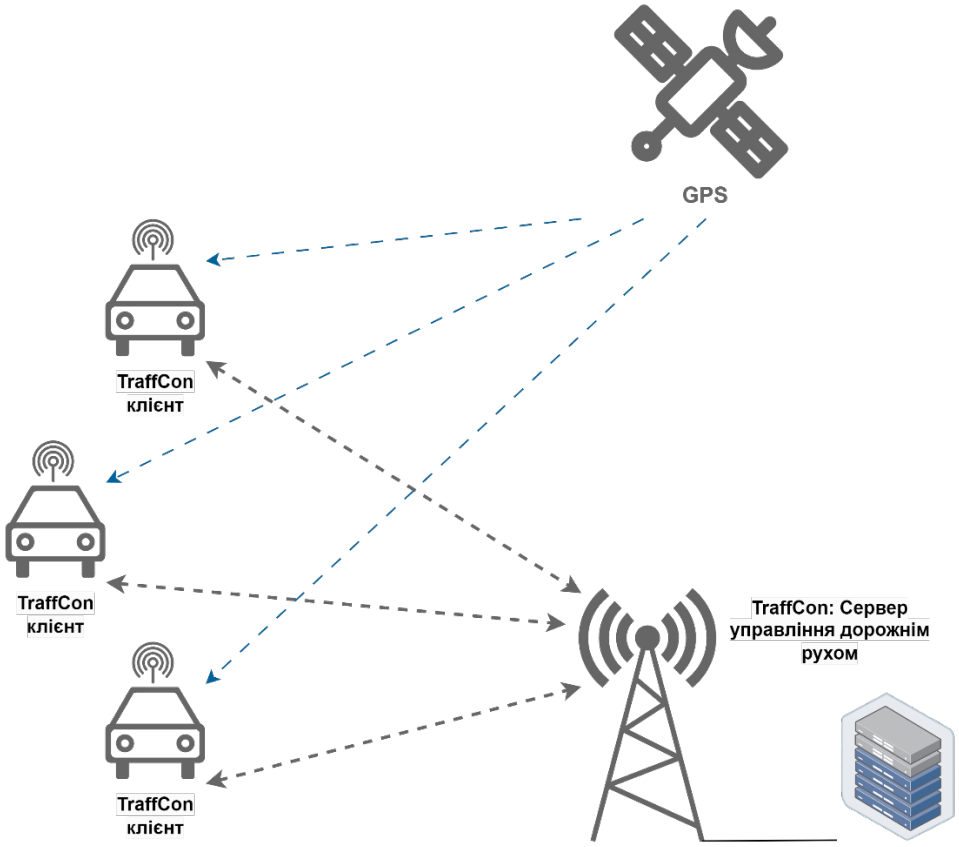


Рис. 1.4. Приклад реалізації системи керування дорожнім рухом [14]

Прикладом реалізації такої системи, є підсистема аналітики та планування транспортних маршрутів Сінгапуру. Дана система використовує алгоритми та моделі на основі машинного навчання для планування оптимальних маршрутів та розкладів руху, враховуючи обсяги пасажирського потоку та дорожні умови.

Система безпеки руху. Забезпечення безпеки пасажирів та оперативна реакція на аварійні ситуації є критичним аспектом, який повинні забезпечувати сучасні транспортні системи. Дані системи включають в себе ряд підсистем, таких як: системи виявлення аварій, відеоспостереження, аварійна зупинка та

інші засоби для забезпечення безпеки пасажирів та персоналу. В деяких реалізаціях, транспортні системи може автоматично реагувати на надзвичайні ситуації та сповіщати служби екстреної допомоги. Наприклад, вищезгадана транспортна система м. Лондон, Велика Британія підключена до служб екстреної допомоги для оперативної реакції на можливі надзвичайні ситуації.

Система електронної оплати проїзду. В даних системах (рис. 1.5) використовуються такі технології, як смарт-картки або мобільні платежі, щоб спростити процес оплати в громадському транспорті.

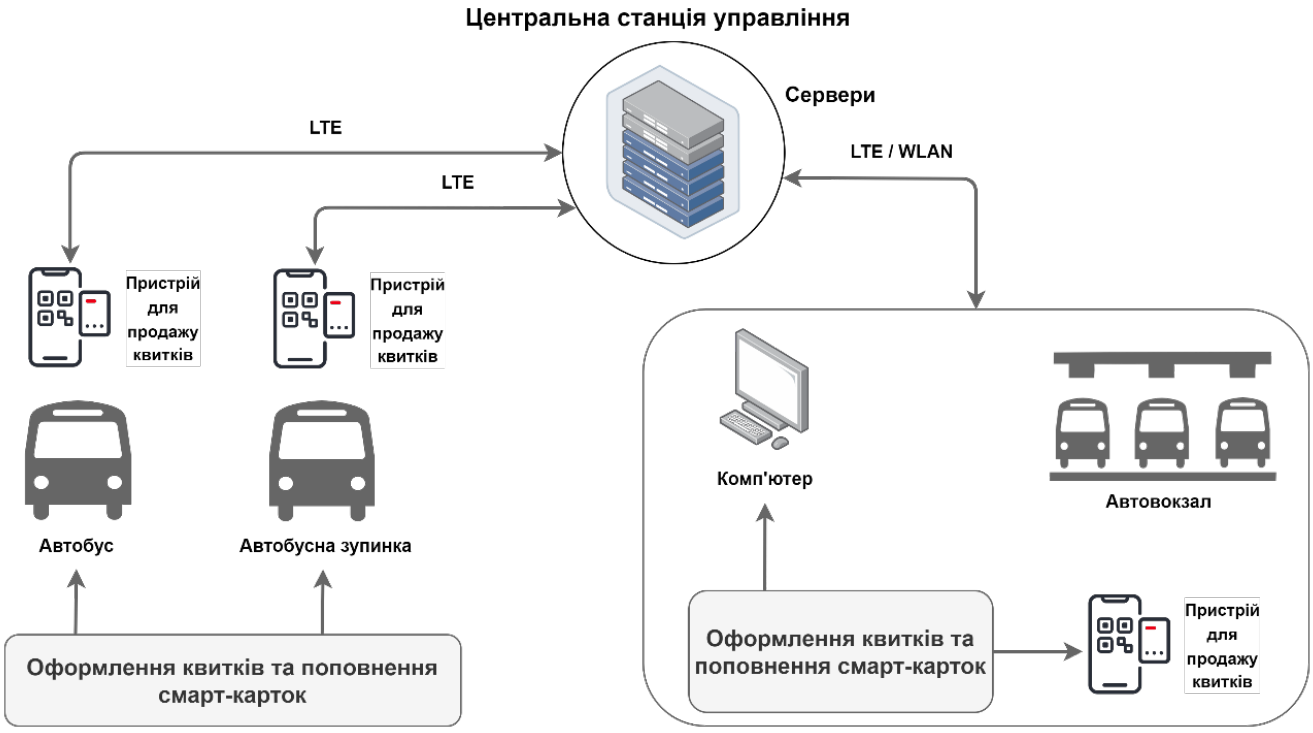


Рис. 1.5. Компоненти системи електронної оплати проїзду [15]

Також на основі цієї інформації можуть накопичуватися дані про кількість пасажирів, які можна використовувати для покращення планування та оптимізації маршрутів, а також для динамічного ціноутворення та інших тарифних політик, щоб оптимізувати дохід а також покращити надання транспортних послуг. Такі підсистеми є основним компонентом інтелектуальних транспортних систем та присутні в усіх розглянутих вище прикладах.

1.3. Аналіз існуючих інтелектуальних транспортних систем громадського транспорту

У сучасному містобудуванні та транспортному плануванні існує необхідність в пошуку та впровадженні інноваційних технологічних рішень для покращення ефективності та якості громадського транспорту. Ріст населення у містах, погіршення екологічної ситуації ставлять перед суспільством та бізнесом завдання покращити управління громадським транспортом та забезпечити максимально зручний, доступний та сталий рух. Розробка та впровадження інтелектуальних транспортних систем стає ключовим напрямком у розв'язанні зазначених викликів. А тенденція зростання інтересу до них, створює новий технологічний ринок. В Таблиці 1.1 наведено основних виробників вищенаведених систем, а також міста їх впровадження.

Таблиця 1.1. Виробники інтелектуальних систем громадського транспорту

Виробник	**Система**	**Функціонал**	**Приклад реалізації, функціонал системи**
Siemens Mobility	Siemens Intelligent Traffic Systems	Моніторинг руху, оптимізація маршрутів, пасажирська інформація	Мюнхен, Німеччина: *оптимізація маршрутів та розкладів руху*
Cubic Transportation Systems	NextBus, Cubic Mobile Suite	Системи оплати, керування доступом, аналіз даних	Сан-Франциско, США: *платежі та управління рухом*
Trafi	Trafi Platform	Мобільні додатки для пасажирів, планування маршрутів	Вільнюс, Литва: *оптимізація маршрутів*

Moovit	Moovit App	Мобільний додаток для планування маршрутів та інформації	Лос-Анджелес, США: *планування маршрутів*
IBM	IBM Intelligent Transportation	Аналіз даних, оптимізація руху	Копенгаген, Данія: *оптимізація розкладів руху*
Transdev	Transdev Intelligent Transport System	Моніторинг, оптимізація, диспетчеризація	Ліон, Франція: *оптимізація розкладів руху*
INIT	MOBILE-ITCS, MOBILE-AVMS	Моніторинг, диспетчеризація, планування	Абу-Дабі, Об'єднані Арабські Емірати: *оптимізація маршрутів та розкладів руху*
Optibus	Optibus Platform	Планування маршрутів, розкладів, аналіз даних	Тель - Авів, Ізраїль: *оптимізація маршрутів та розкладів руху*
Conduent	ATLAS	Моніторинг, оптимізація, пасажирська інформація	Австралія *оптимізація маршрутів та розкладів руху*

На сьогоднішній день лідерами ринку є компанії Siemens та Cubic Transportation Systems, які пропонують комплексні системи на основі концепції Інтернету речей.

- Siemens Mobility є дочірньою компанією Siemens AG і спеціалізується на розробці та впровадженні інтелектуальних рішень для управління та оптимізації транспортної інфраструктури, включаючи громадський транспорт.

Siemens Intelligent Traffic System є комплексною інтегрованою системою для управління рухом громадського транспорту. Дана система надає функціонал моніторингу місцезнаходження та технічного стану транспортних засобів [16].

На основі цих даних система проводить аналіз для визначення оптимальних маршрутів та розкладів, а також надає інформацію пасажирам через інтерфейс мобільного додатку. Також є опціональна можливість інтеграції з системою світлофорів. На Рисунку 1.6 наведені основні компоненти Siemens Intelligent Traffic System.

Рис. 1.6. Компоненти системи Siemens Intelligent Traffic System

Недоліками даного рішення є те, що Siemens Intelligent Traffic System є закритою екосистемою, без можливості сторонніх інтеграцій, як от публічні API, тощо. Це значно звужує функціонал системи а також ускладнює адаптацію для умов конкретних регіонів. Також до недоліків можна віднести те, що в даній системі в якості основного протоколу комунікації використовується [16] стандарт ITS-G5, що значно ускладнює інтеграцію даної системи у більшості Європейських містах, оскільки вимагає реалізацію додаткової інфраструктури.

- Cubic NextBus - це інтелектуальна система моніторингу та управління громадським транспортом, розроблена компанією Cubic Transportation Systems. Дана система спрямована на покращення ефективності та зручності громадського транспорту шляхом використання технологій та аналізу даних на

основі апарату нейронних мереж. Система забезпечує постійний моніторинг руху автобусів та інших транспортних засобів за допомогою GPS-датчиків (рис. 1.7).

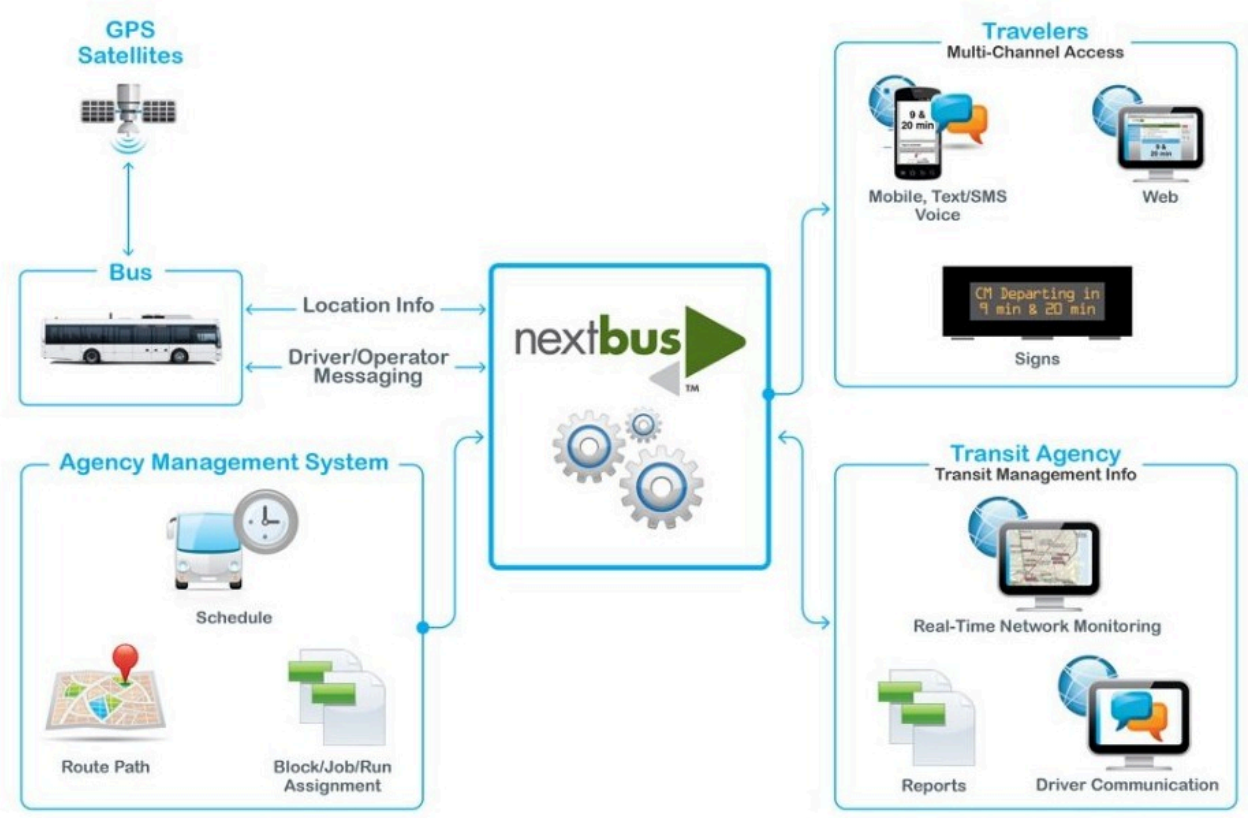


Рис. 1.7. Структура системи NextBus [17]

Також система збирає дані про пасажиропотоки, завантаження транспорту та інші параметри. На основі зібраних даних, система здатна прогнозувати час прибуття транспорту на конкретну зупинку. Cubic NextBus може бути інтегрована з платіжними системами, що дозволяє пасажирам здійснювати оплату проїзду онлайн, використовуючи мобільні додатки чи картки. До недоліків даної системи слід віднести її обмежений функціонал, а також відсутність можливості додаткових сторонніх інтеграцій, так само, як це було і в попередній системі. Це обмеження призводить до неможливості адаптації та розширення системи. Ще одним значущим недоліком даної системи є використання програмного забезпечення з монолітною архітектурою, що ускладнює масштабування та підтримку.

1.4. Аналіз існуючих методів побудови систем громадського транспорту на основі концепції Інтернету речей та їх недоліків

Для побудови Інтелектуальної транспортної системи на основі концепції Інтернету речей, та вдосконалення вже існуючих необхідно проаналізувати методологічні аспекти побудови IoT систем в цілому. Ефективність використання ресурсів та масштабованість є основними характеристиками IoT-систем, в тому числі і в контексті Інтелектуальних транспортних систем.

Протягом останніх 10 років спостерігається тенденція зростання кількості наукових досліджень спрямованих на покращення методів побудови систем Інтернету речей, а також їх застосування в Інтелектуальних транспортних системах. На основі проведеного аналізу, можемо виділити основні проблеми досліджень: методи побудови архітектури системи та технологій передачі інформації, методи обробки та збереження даних, проблеми безпеки та конфіденційності.

Окремо можна виділити дослідження, в яких пропонуються комплексні рішення для систем громадського транспорту [18-44], що корелюють з темою даного дослідження. В роботах [25, 26, 27] розглядається процес побудови Інтелектуальної транспортної системи на основі концепції Інтернету речей для столичного регіону. Автори пропонують систему з таким функціоналом як: визначення часу прибуття транспортного засобу, побудову оптимальних маршрутів, а також аналіз на основі історичних даних. Але методи та технології, які використовуються в даній системі, є необґрунтованими, а також не визначається стійкість даної системи до високого навантаження та масштабування. Схоже рішення пропонується авторами в дослідженнях [28, 29], де автори пропонують прототипи інтелектуальних транспортних систем. До недоліків даного дослідження можна віднести те, що архітектура яка пропонується не досліджена з точки зору масштабування, та можливості застосування в реальних системах з динамічним навантаженням. У дослідженні [45] автори розглядають можливість застосування методів нейронних мереж в

транспортному секторі. Ідея використання нейронних мереж для обробки даних, що генеруються в системах Інтернету речей, є актуальною та перспективною, що підтверджують результати досліджень [46-54]. Проте необхідно провести ширший аналіз та оцінити результати застосування вищенаведених методів у реалізованій системі.

Одним з головних компонентів, які забезпечуються стійкість та надійність IoT системи, є телекомунікаційна мережа. На сьогоднішній день, недостатньо дослідженими є проблеми обґрунтованого вибору технологій та протоколів передачі даних а також їх вплив на поведінку системи в цілому.

При розробці архітектури системи збереження та обробки даних важливим аспектом є вибір технологій для серверних обчислень. На сьогоднішній день, є декілька основних парадигм: обчислення на виділених серверах, а також хмарні та туманні обчислення. Обчислення на виділених серверах вважаються класичним підходом, який застосовується у більшості існуючих систем. У той же час, концепції хмарних та туманних обчислень з'явилися нещодавно, привертаючи значну увагу дослідників у контексті їхнього потенційного застосування в системах Інтернету речей.

Наприклад, в роботі [55] автори розглядають загальні парадигми серверних обчислень, та роль хмарних технологій для концепції Інтернету речей. Наведені результати досліджень показують, що потенціал сучасної IoT системи не може бути повною мірою реалізованим на основі класичної інфраструктури з виділеними серверами. Причиною цього є проблеми з масштабованістю, пропускною здатністю мережі, а також неефективним використанням обчислювальних ресурсів. Особливо ці недоліки проявляються в системах з динамічним навантаженням, як от інтелектуальні транспортні системи. В роботі [56] автори розглядають концепції високорівневих фреймворків для інтелектуальних транспортних систем. В роботі [57] автори також пропонують високорівневу концепцію системи, без аналізу комунікаційних технологій та методів обробки даних. В роботі [58] запропонована практична реалізація системи з використанням хмарного сховища та бази даних. До недоліків

вищенаведених досліджень можна віднести те, що в них не проведено тестування, та не підтверджено стійкість такої архітектури до навантаження.

Автори досліджень [59-60] запропонували підхід для аналізу та обробки великого об’єму даних, який генерується в сучасних транспортних системах на базі IoT. В запропонованому підході хмарні обчислення використовуються як основна серверна інфраструктура, але доцільність такого підходу не доводиться.

Ще одним важливим аспектом при побудові IoT системи є забезпечення захищених каналів комунікації та безпеки даних. Останні кілька років з’явилося багато досліджень, в які висвітлюють цю тему. Зокрема, в роботі [61] автори проводять дослідження безпеки за допомогою моделювання на фізичному та мережевому рівнях. В роботі [62] проводиться комплексне дослідження проблем кібербезпеки в хмарних обчисленнях на основі концепції Інтернету речей. Автори розглядають проблеми та вразливості, які виникають внаслідок запровадження хмарних обчислень в IoT системах, а також визначають майбутні напрямки досліджень. Також, схоже дослідження проводиться у роботі [63], де автори проводять аналіз потенційних атак в IoT системах, а також відомих методів протидії. У цих дослідженнях розглядаються протоколи безпеки, методи шифрування та механізми контролю доступу для розв'язання проблем безпеки та захисту конфіденційних даних. В той же час, вищенаведені дослідження показують ряд невирішених проблем в сфері безпеки даних. Також суттєвим недоліком є те, що в даних роботах не розглядається вплив того, чи іншого методу захисту даних на швидкість передачі даних.

Іншим невирішеним питанням, є масштабуванням та ефективне використання обчислювальних ресурсів. Першим кроком до його вирішення є визначення та аналіз високонавантажених компонентів системи. Саме такий підхід використаний в роботі [64], де наводиться загальна архітектура системи громадського транспорту, з повним аналізом комунікаційних технологій та протоколів передачі даних. Однак у дослідженні [64] не розглядається поведінка системи при динамічній змінні навантаження, а отже її здатність до масштабування.

Це дає підстави стверджувати, що доцільним є проведення дослідження, присвяченого аналізу застосування технологій серверних обчислень в системах Інтернету речей загалом та в контексті інтелектуальних транспортних систем.

Ще одним важливим аспектом для побудови стійких до навантаження систем, є правильне балансування навантаження. Актуальність проблеми балансування навантаження на сучасному етапі важко переоцінити. Ця ситуація вимагає проведення досліджень у даній області, з метою знаходження нових підходів та удосконалення існуючих методів. Останні кілька років основний фокус дослідників був спрямований на балансування навантаження в системних хмарних та туманних обчислень. В роботі [65]. автори пропонують метод для балансування вхідного навантаження між віртуальними машинами, на основі зміни завантаженості центрального процесора. Однак в даного підходу є обмеження, оскільки вхідне навантаження може бути орієнтованим не на ресурси процесора, а на пам’ять, або мережеві ресурси. Таким чином, даний підхід до балансування буде не ефективним. В роботі [66] автори також використовують концепцію віртуальних машин у хмарних системах. Однак ефективність запропонованого розподілу навантаження між ними, не є до кінця дослідженим. Авторами дослідження [67] запропоновано метод розподілу навантаження на основі «нарізання» мережевих ресурсів в контексті туманних обчислень. Однак, концепція туманних обчислень, а отже і запропоновані методи не завжди можуть бути застосовані в IoT мережі.

Ще одним підходом у вирішенні завдань розподілу навантаження, є балансування на основі геокоординат. В дослідженні [68] автори пропонують модель розподілу навантаження, на основі географічного розташування підключених серверів (Geography-Aware). В контексті системи Інтернету речей, дані методи є обмеженими, оскільки підключені пристрої часто не є георозподіленими. Також дослідження розподілу навантаження на основі даних алгоритмів проведено у роботі [69]. В цьому дослідженні автори пропонують підходи для балансування навантаження в системах зберігання даних.

Також алгоритми на основі георозподілу застосовуються в дослідженні [70], для розв'язання задачі розподілених датацентрів. Така модель є ефективною для вищенаведених систем, але недостатньо ефективною для систем з динамічним та важкопрогнозованим навантаженням, таких якими є IoT мережі.

В дослідженні [71] авторами запропоновано комплексну систему розподілу навантаження в IoT мережі. В основі концепції даної системи, є динамічне виділення ресурсів для кожної частини мережі, та процес сегментації. До недоліків надої системи можна віднести складність реалізації, через необхідність забезпечення додаткового рівня туманних обчислень.

В роботі [72] проводиться комплексний аналіз методів балансування навантаження в хмарних та туманних обчисленнях, а також можливість їх застосування в IoT системах. Однак для оцінки ефективності розглянутих методів в контексті IoT систем не проведено науково-практичних експериментів, що є обмеженням даного дослідження. В роботі [73] автори розглядають енергоефективні методи балансування навантаження в системах туманних обчислень. Даний підхід є цікавим та перспективним, оскільки орієнтується на загальну енергоефективність системи, але водночас обмеженим, через необхідність реалізації додаткового рівня туманних обчислень. Схоже дослідження проведено в роботі [74], де розглядається застосування існуючих методів балансування в системах туманних обчислень. Результати отримані в даному досліджені, є також обмеженими в контексті систем Інтернету речей, через необхідність додаткового обчислювального рівня.

Ще одним підходом до розв'язання проблеми балансування навантаження є застосування методів машинного навчання. Автори роботи [75] пропонують схему балансування навантаження на основі нейронних мереж для систем Інтернету речей. Для визначення ефективності даного методу, авторами проведено математичне моделювання, однак не досліджено роботу даних методів в контексті динамічних систем з розподіленою архітектурою. Схожий метод пропонується в роботі [76], де детально розглядається топологія мережі, та способи розподілу навантаження в ній. До недоліків даного дослідження можна

віднести складність реалізації та ускладнення архітектури системи. Автори дослідження [77] пропонують методи балансування навантаження при багатошляховій маршрутизації. Однак в даному дослідженні немає практичних експериментів, які б підтверджували чи спростовували ефективність наведених методів.

Ще одним напрямом проведення досліджень проблеми балансування навантаження, є застосування методів математичного моделювання. Авторами роботи [78] запропонована математична теорія динамічного балансування навантаження в стільникових мережах. Серед недоліків даного підходу можна виділити обмежену адаптивність теорії до зміни інтенсивності вхідного навантаження мережі. В роботі [79] автори розглядають квазіопозиційний алгоритм для балансування навантаження в середовищі хмарних обчислень. В роботі [80] пропонуються математичні моделі для визначення оптимальної кількості обчислювальних контейнерів в системах інтернету речей. Основним недоліком цього підходу може бути складність адаптації моделей до змін в конфігураціях та навантаженнях систем. В дослідженні [81] розглядається модель оптимізації для планування завдань у хмарних обчисленнях, проте дана модель не враховує динамічної зміни навантаження. Автори роботи [82] пропонують математичну модель, для підвищення ефективності використання ресурсів бази даних у хмарних обчисленнях. Головним обмеженням зазначеного дослідження є спрямування на класичні веб-додатки та комерційні рішення, які на враховують динамічне навантаження, що виникає у системах Інтернету речей.

Тому є необхідність в розробці ефективної методики балансування навантаження, в розподілених IoT системах. Дана методика має забезпечити рівномірний розподіл завдань між доступними серверами, та максимально ефективне використання обчислювальних ресурсів. Розробка вищезазначеної методики є ще одним завданням даного дослідження.

1.5. Постановка завдання дослідження

Аналіз існуючих комерційних рішень Інтелектуальних транспортних систем, а також методів побудови таких систем показав ряд невирішених проблем, які лягли в основу цього дослідження. Завданням дисертаційного дослідження є розробка масштабованої та відмовостійкої системи громадського транспорту на основі концепції Інтернету речей. Для вирішення основного завдання дослідження необхідно виконати ряд науково-технічних задач. А саме:

1. Провести аналіз існуючих методів побудови, а також існуючих реалізацій Інтелектуальних транспортних систем.

2. Розробити структуру телекомунікаційної мережі системи Інтернету речей для забезпечення стабільного та захищеного каналу передачі даних. Провести експериментальні дослідження та обґрунтувати вибір оптимального варіанту для побудови системи.

3. Провести аналіз існуючих протоколів для передачі даних програмного рівня, а також розробити комплекс моделей для оцінювання їх продуктивності. Провести експериментальні дослідження та обґрунтувати вибір оптимального варіанту для побудови системи.

4. Розробити математичну модель для моделювання поведінки IoT системи, а також влив архітектури мережі на загальну продуктивність системи.

5. Розробити архітектуру системи збереження та обробки даних, а також моделі та методи для оцінки її продуктивності та здатності до масштабування.

6. Розробити метод для визначення оптимальної кількості обчислювальних контейнерів в розподілених системах Інтернету речей.

7. Розробити математичну модель для оцінки рівномірності розподілу навантаження

8. Розробити вдосконалений метод балансування навантаження в системах Інтернету речей на основі MQTT протоколу.

9. Вдосконалити існуючі методи обробки та аналізу даних на основі апарату нейронних мереж.

10. Виконати практичну реалізацію системи на основі запропонованої архітектури та методів. Провести експериментальне дослідження для оцінки масштабованості, відмовостійкості та ефективності використання ресурсів.

В результаті даного дослідження, має бути реалізована Інтелектуальна система громадського транспорту, основні компоненти якої виділені на Рис. 1.8.

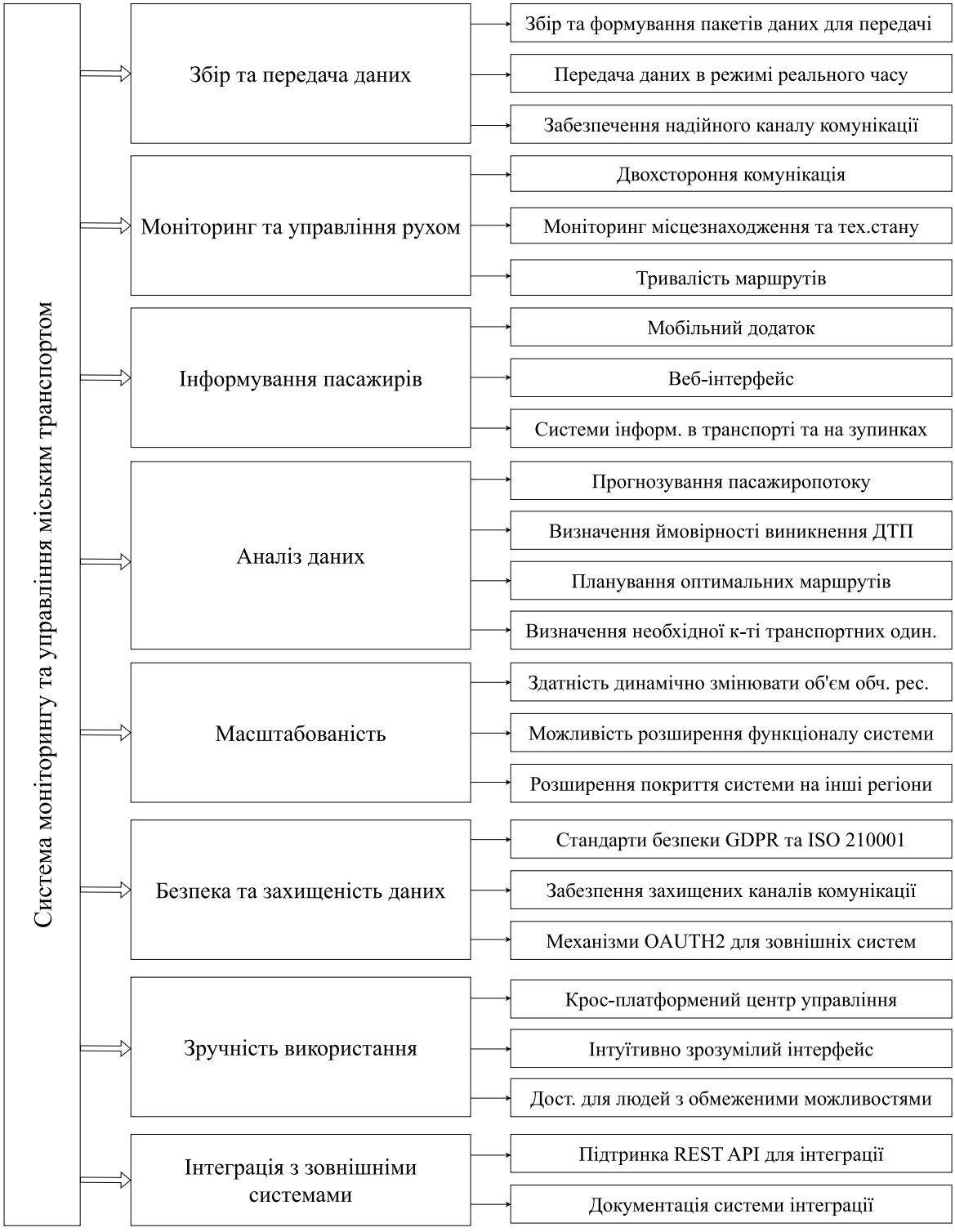


Рис. 1.8. Компоненти системи моніторингу та управління міським транспортом

Рис. 1.9. Блок-схема досліджуваної системи

1.6. Методологія проведення дослідження

На першому етапі проводиться комплексний аналіз та визначаються технічні вимоги, які ставляться до досліджуваної системи . На основі визначених технічних вимог будується структурна схема системи та визначаються основні її компоненти. Після визначення вимог, розглядаються існуючі методи реалізації систем Інтернету речей, а також визначаються їх обмеження та недоліки в контексті застосування для реалізації інтелектуальних систем громадського транспорту.

На наступному етапі виділяються основні інформаційні потоки , а також розробляється структурна схема телекомунікаційної мережі, яка буде забезпечувати передачу даних в досліджуваній системі. Також проводяться експериментальні дослідження для визначення оптимальних технологій та методів передачі даних, а також методи оцінки їх продуктивності.

Після розробки структури телекомунікаційної мережі розробляється архітектура системи збереження та обробки даних, а також методи експериментальної оцінки її масштабованості та доступності. Для цього розробляється математична модель, яка дозволяє описати досліджувану систему, та оцінити вплив архітектури системи на її продуктивність.

На наступному етапі, більш детально розглядаються проблеми оптимального використання ресурсів та балансування навантаження. Розглядаються методи обробки даних на основі апарату нейронних мереж.

Після розробки архітектури системи, обґрунтування технологій та протоколів передачі даних, а також вищенаведених методів, розробляється прототип системи, а також проводиться її тестування на відповідність до поставлених завдань.

На Рис. 1.10 наведено методологію проведення дисертаційного дослідження.

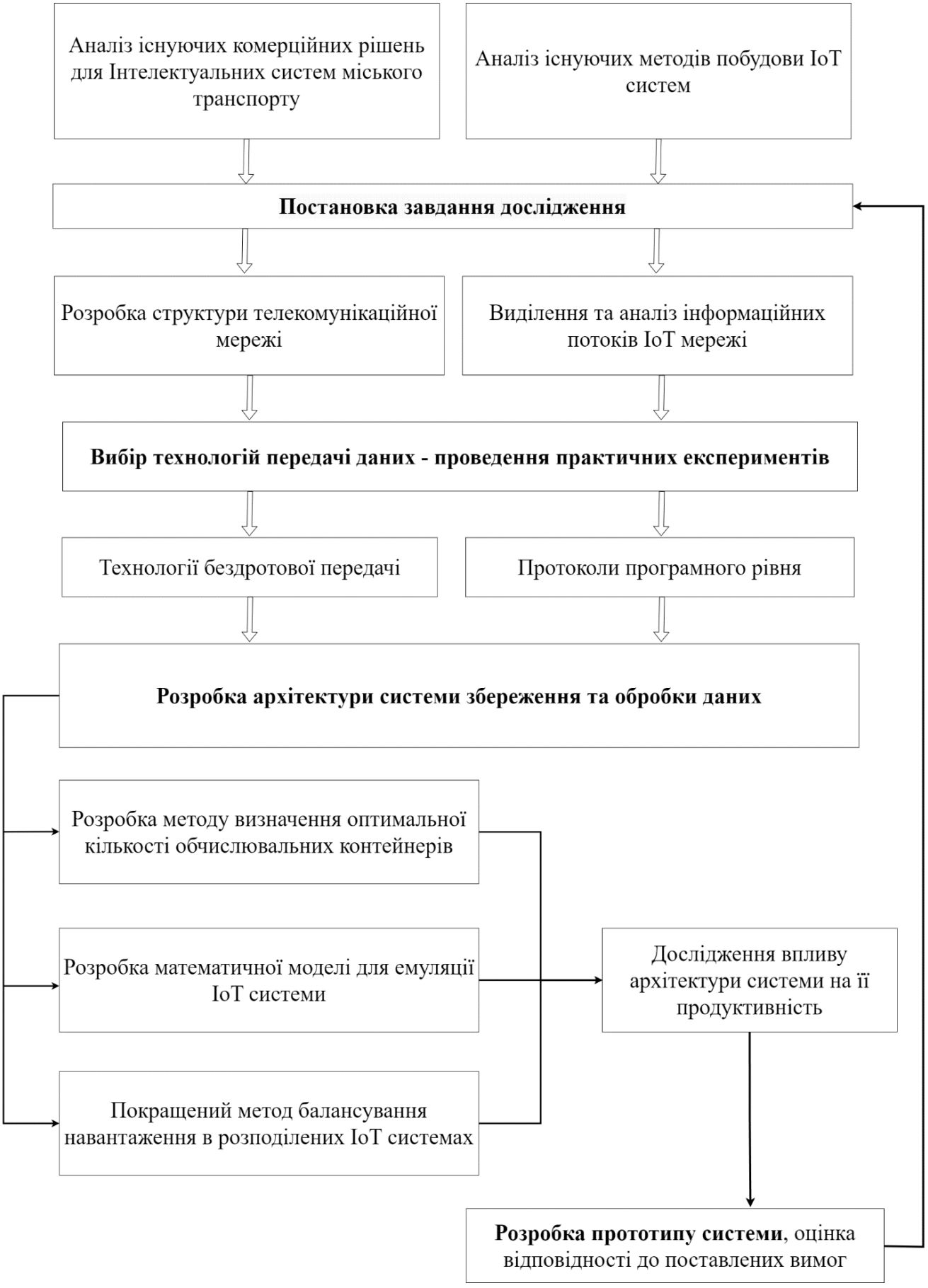


Рис. 1.10. Методологія проведення дослідження

1.7. Висновки

1. На підставі класифікації та багатокритеріального аналізу наявних комерційних рішень для інтелектуальних транспортних систем було встановлено їх основні технічні характеристики і функціональні можливості, а також визначено переваги та недоліки.

2. Проведено аналіз існуючих наукових методів побудови систем на основі концепції Інтернету речей, а також можливості та умови їх застосування в контексті розробки систем громадського транспорту. На основі проведеного аналізу виділено основні недоліки, а також виокремлено напрямки, які потребують проведення подальших наукових досліджень. А саме найбільш недослідженими напрямками є методи моделювання та побудови розподілених систем з динамічним балансуванням навантаження. Крім того, даний аналіз дозволяє стверджувати, що методи комплексного моделювання телекомунікаційних мереж для IoT систем у наукових дослідженнях є недостатньо обґрунтованими в цілому.

3. На підставі проведеного аналізу сформульовано завдання дослідження, а також продемонстровано методику проведення дисертаційного дослідження. В основі якого лежить вирішення комплексу науково-технічних завдань, що спрямовані на розробку нових методів побудови IoT систем та покращення вже існуючих.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТЕЛЕКОМУНІКАЦІЙНОЇ МЕРЕЖІ ТА ІНФОРМАЦІЙНИХ ПОТОКІВ ДЛЯ СИСТЕМИ МОНІТОРИНГУ ТА УПРАВЛІННЯ ГРОМАДСЬКИМ ТРАНСПОРТОМ

Телекомунікаційна мережа є ключовим елементом системи Інтернету речей, зокрема досліджуваної системи громадського транспорту. Оскільки вона забезпечує зв’язок в режимі реального часу між транспортними засобами, обчислювальною інфраструктурою та центрами управління. Побудова телекомунікаційної мережі для IoT системи складається з декількох етапів:

- Визначення вимог: Перший етап - це визначення вимог до телекомунікаційної мережі IoT системи. Необхідно визначити, які типи пристроїв та датчиків будуть використовуватись, які дані потрібно передавати, яка відстань передачі та пропускна здатність потрібні, а також які вимоги до безпеки та надійності.
- Вибір технологій: На другому етапі необхідно вибрати відповідні технології для побудови мережі. Це може включати бездротові технології, такі як Wi-Fi, Bluetooth, Zigbee, NFC або мережі операторів мобільного зв'язку (GSM, 3G, 4G, 5G). Вибір технологій залежить від потреб у зоні покриття, пропускній здатності та енергоефективності.
- Розробка структури мережі: Після вибору технологій потрібно розгорнути необхідну інфраструктуру мережі. Це може включати встановлення мережевих пристроїв, таких як маршрутизатори, комутатори, базові станції, антени, сенсори тощо. Крім того, необхідно встановити і налаштувати мережеве програмне забезпечення для управління мережею та забезпечення безпеки передачі даних.
- Підключення пристроїв: Після побудови інфраструктури мережі необхідно підключити IoT пристрої до мережі. Це може включати реєстрацію та налаштування пристроїв, надання їм необхідних ідентифікаторів та ключів

безпеки, а також встановлення зв'язку з центральними серверами або хмарними платформами.

- Тестування та впровадження: Після підключення пристроїв потрібно провести тестування мережі. Це включає перевірку зв'язку, надійності, пропускної здатності та безпеки мережі. При необхідності вносяться корективи та оптимізації.
- Управління та підтримка: Після впровадження мережі важливо забезпечити її ефективне управління та підтримку. Це включає моніторинг мережі, виявлення та усунення відмов, оновлення програмного забезпечення, забезпечення безпеки та надання технічної підтримки.

2.1. Інформаційні потоки системи

Для побудови архітектури мережі Інтернету речей, на першому етапі було виділено чотири основні інформаційні потоки (таб. 2.1), а також розглянуто та проаналізовано комунікаційні технології для передачі даних між ними.

Таблиця 2.1. Інформаційні потоки

	Мережевий рівень	**Опис**
Сенсор – Контролер	Фізичний	Комунікація між IoT пристроєм та підключеними сенсорами. UART, I2C, SPI, CAN-BUS.
Контролер – Сервер	Транспортний / Програмний	Комунікація між IoT контролером та обчислювальним сервером. TCP/IPv4/ IPv6, MQTT, HTTP, COAP

Сервер – Сервер	Програмний	Міжсерверна комунікація. MQTT, HTTP, COAP
Сервер – База даних	Транспортний / Програмний	TCP/IPv6

Передача даних між IoT сенсорами та сервером (рис. 2.1) у системі громадського транспорту включає в себе збір і передачу інформації про рух транспортних засобів, стан обладнання і пасажирську інформацію через мережу зв'язку на віддалений сервер для подальшої обробки та аналізу.

Рис. 2.1. Передача даних між IoT сенсором та сервером

Передача команди від сервера до IoT контролера (Рис. 2.2) в системі громадського транспорту полягає в передачі віддалених інструкцій через мережу зв'язку для керування функціями транспортного засобу. На сервері створюється команда відповідно до вимог задачі. Ця команда містить інструкції та параметри, необхідні для виконання дії.

Рис. 2.2. Передача зовнішньої команди на IoT контролер

Команда передається через мережу зв'язку до IoT контролера, який знаходиться на відповідному транспортному засобі. Цей процес може

використовувати протоколи зв'язку, такі як MQTT, HTTP або інші, залежно від конфігурації системи. IoT контролер на транспортному засобі отримує команду, розшифровує її і виконує необхідні дії. Наприклад, якщо команда полягає в зміні швидкості, контролер регулює роботу двигуна, відповідно до вказівок. Цей процес передачі команд забезпечує двосторонній обмін інформацією між сервером і транспортними засобами в реальному часі.

У системі громадського транспорту може бути декілька серверів, які виконують різні функції, наприклад, моніторинг, управління, обробку платежів тощо. Інформація про транспортні засоби, маршрути, розклади та інші дані зберігаються в спільній базі даних. Ця БД служить як центральне сховище для інформації. Сервери взаємодіють один з одним (рис. 2.3) через мережу для обміну даними та командами.

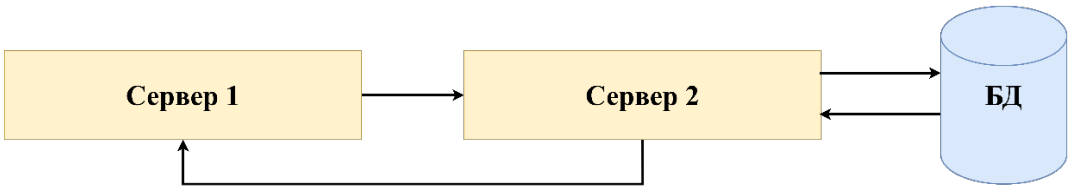


Рис. 2.3. Міжсерверна комунікація з доступом в БД

Це може включати в себе передачу інформації про стан транспорту, замовлення пасажирів, оновлення розкладів тощо.

Пасажири громадського транспорту, або інші користувачі, які мають доступ до системи, ініціюють команду (рис. 2.4) або запит на керування певною функцією транспортного засобу. Це може бути, наприклад, вимога на зупинці змінити маршрут або звернутися до водія. Система приймає команду від користувача та обробляє її. Це може включати в себе перевірку прав доступу та аутентифікацію користувача.

Також, система визначає конкретний транспортний засіб, до якого адресована команда. Це може відбуватися на основі ідентифікатора транспорту

або інших параметрів. IoT контролер, який отримав команду, інтерпретує її та виконує відповідні дії. Наприклад, якщо це команда на зміну маршруту, контролер може надати водію нові напрямки.

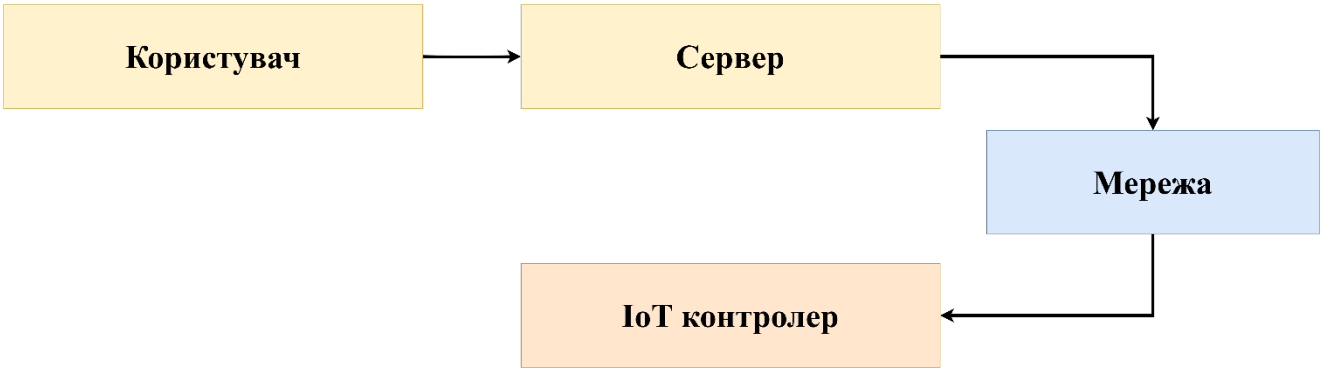


Рис. 2.4. Передача команди від користувача до IoT контролера

Після виконання команди IoT контролер повідомляє систему про статус виконання. Узагальнену схему телекомунікаційної мережі наведено на Рисунку 2.5.

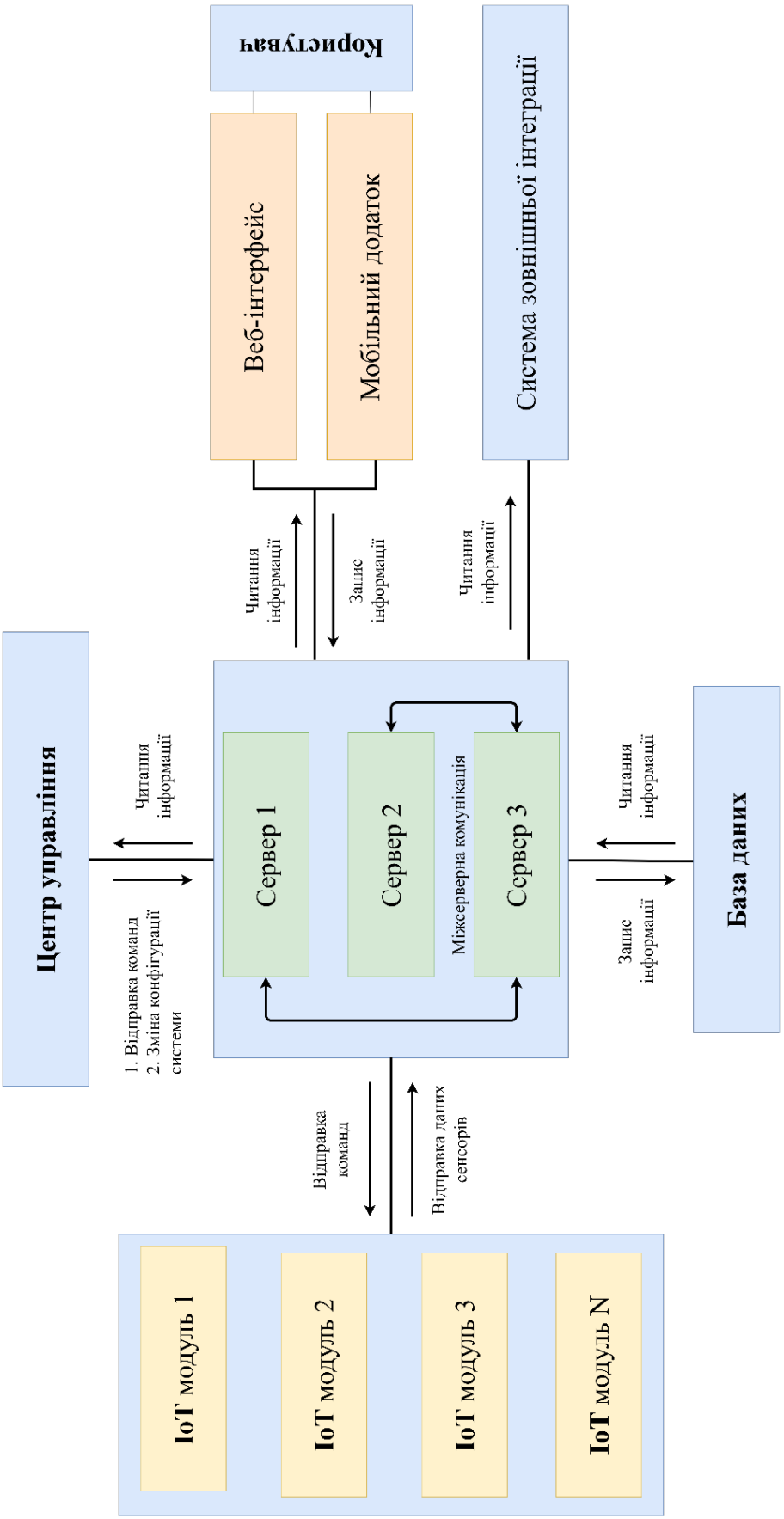


Рис. 2.5. Узагальнена схема телекомунікаційної мережі для системи моніторингу та управління громадським транспортом

2.2. Аналіз та вибір технологій для передачі даних

Важливим етапом проектування та реалізації телекомунікаційної мережі є аналіз та вибір оптимальних технологій передачі даних, які забезпечать комунікацію між усіма компонентами мережі. У цьому розділі проведено аналіз наявних технологій передачі даних в IoT системах, а також обґрунтовано вибір для телекомунікаційної мережі моніторингу міського транспорту. Оскільки система громадського транспорту є динамічною системою Інтернету речей, де підключенні IoT пристрої постійно змінюють своє розташування, то необхідно розглянути бездротові технології передачі даних.

2.2.1 Огляд бездротових технологій передачі в системах Інтернету речей

Бездротові технології передачі можна розділити на дві групи згідно з їх діапазоном дії: ближньої (таб. 2.2) та дальньої (таб. 2.3) дії.

Таблиця 2.2. Технічні характеристики технологій ближньої дії

Технічні характеристики	Wireless RF	Bluetooth	Wi-Fi	Zigbee
Відстань дії	До 100 м за прямої видимості	10 м	До 120 м	100 м
Частота	315/433 МГц	2.4 ГГц	2.4 ГГц. 5 ГГц	915 МГц / 2.4 ГГц
Швидкість передачі даних	10-115.2 кбіт/с	< 1 Мбіт/с	7 Гбіт/с	250 кбіт/с
Підтримка мобільних датчиків	-	Так	Так	Так
Відомості про місцезнаходження датчика	-	Ні	Так	Так
Енергоспоживання	Наднизьке	Низьке	Високе	Низьке

Таблиця 2.3. Технічні характеристики технологій дальньої дії

Технічні характеристики	LoRaWan	SigFox	GSM/LTE
Відстань дії	2-5 км в міській зоні, та 45 км в приміській зоні	10 км в міській зоні, та до 50 км в приміській зоні	10-28 км
Частота	< 1 ГГЦ	Частотно незалежна технологія	700-2600 МГц
Швидкість передачі даних	0.3 - 50 Кбіт/с	100 – 1000 біт/с	0.1 – 1 Гб/с
Підтримка мобільних датчиків	Так	Ні	Так
Відомості про місцезнаходження датчика	Так	Ні	Так
Енергоспоживання	Низьке	Низьке	Високе

Для визначення оптимальної технології для інтелектуальної системи громадського транспорту, розглянемо таку систему у середньостатистичному європейському місті та необхідну дальність покриття. Середньостатистичне європейське місто може мати площу у межах кількох сотень квадратних кілометрів до кількох тисяч квадратних кілометрів. Для системи громадського транспорту, необхідно відстежувати транспортні засоби на всій території міста.

На основі аналізу параметрів технологій, наведених у Таблиці 2.2 та. 2.3, можна зробити висновок, що для побудови системи громадського транспорту в середньостатистичному європейському місті, де важливо забезпечити ефективне покриття великої території та надійний зв'язок з багатьма транспортними засобами, застосування технологій дальньої дії, таких як LoRaWAN, SigFox, LTE, є найбільш обґрунтованим та оптимальним рішенням. На Рисунку 2.6 наведено співвідношення між швидкістю передачі даних та дальністю дії [83].

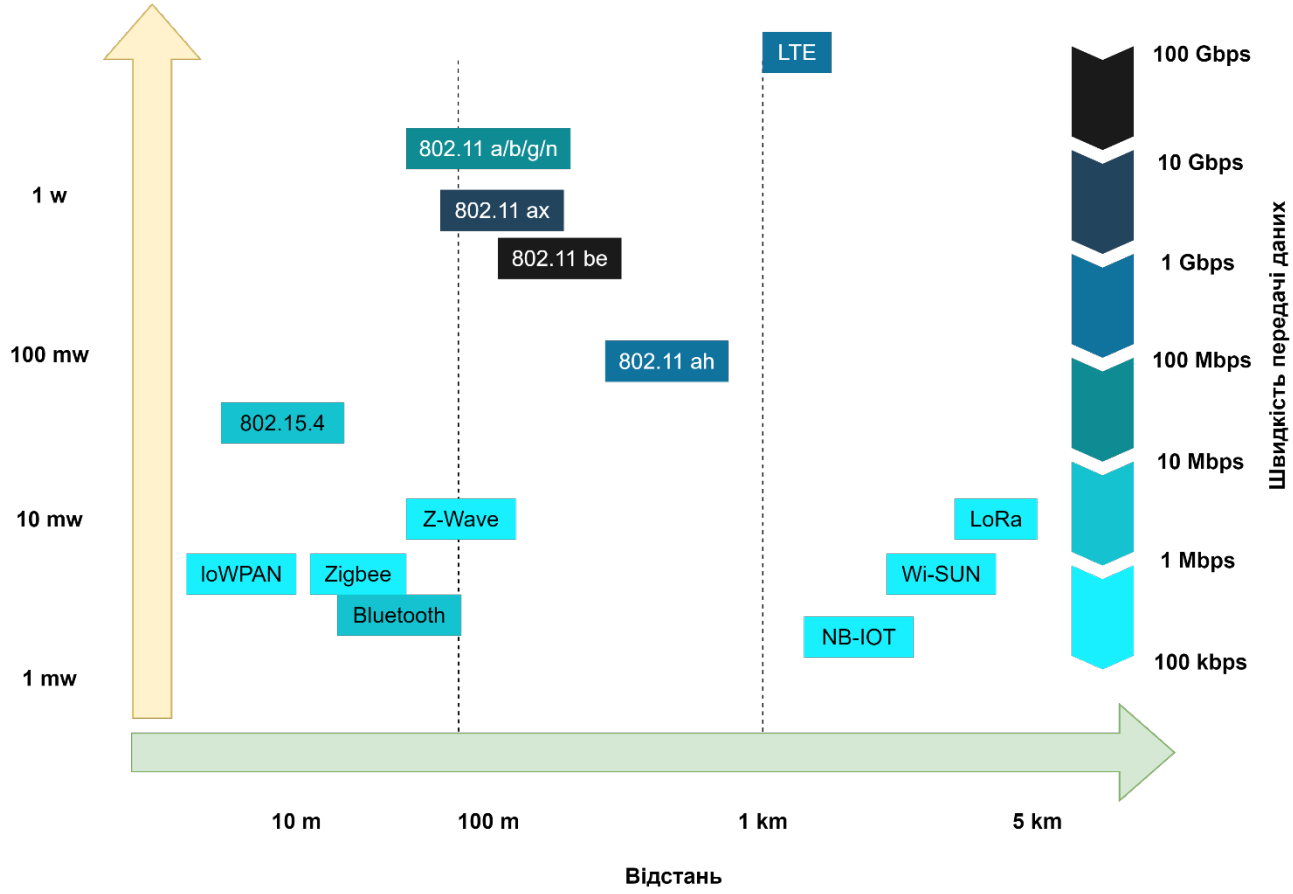


Рис. 2.6. Залежність швидкості передачі даних та енергоспоживання від відстані дії технології

Для системи громадського транспорту в середньостатистичному європейському місті, де вже існує розгорнута мережа LTE (4G), вибір технології LTE є оптимальним. Основними перевагами LTE є наявність широкого покриття мережі без потреби будувати додаткову інфраструктуру (як у LoRaWAN і SigFox), що робить її більш витрато-ефективною і практичною для впровадження у вже в існуючих міських середовищах.

Четверте покоління стільникового мобільного зв'язку, відоме як LTE технологія (Long Term Evolution), створено у відповідності до вимог Міжнародного союзу електрозв'язку партнерською асоціацією груп телекомунікаційних компаній 3GPP [84, 85, 86, 87]. Робота почалась у 2004 році, в грудні 2008 року вийшов Реліз 8 3GPP (таб. 2.4) [88], а перше застосування LTE відбулося в 2009 році.

Таблиця 2.4. Еволюція LTE поколінь

3GPP Реліз	99/4	5/6	7	8-9	10-12
Роки	2003/4	2005-2008	2008-2009	2010	
Швидкість завантаження даних	384 kbps	14 Mbps	28 Mbps	150 Mbps	1 Gbps
Швидкість передачі даних	128 kbps	5.7 Mbps	11 Mbps	75 Mbps	
Час затримки (RTT)	~ 150 ms	<150 ms	<50 ms	LTE ~10 ms	

Реліз 8 3GPP визначив параметри та характеристики технології LTE, а саме:

- Високу пікову швидкість передачі до 300 Мб/с в каналі вниз та 75 Мб/с - вгору при використанні 4×4 MIMO (англ. Multiple Input Multiple Output) з шириною смуги пропускання 20 МГц;
- Високу спектральну ефективність в змінних смугах частот 1.4 МГц, 3 МГц, 5 МГц, 10 МГц, 15 МГц, 20 МГц;
- Підтримка дуплексних каналів FDD (англ. Frequency-division duplexing) і TDD англ. Time-division duplexing);
- Затримка до 5 мс для IP пакетів при поширенні радіохвиль в каналах з ідеальними умовами;
- Модуляція OFDM (англ. Orthogonal frequency-division multiplexing);
- Технологія MIMO 2×2 та 4×4;

Спектральна ефективність LTE відповідно до Шеннона забезпечується розширенням смуги пропускання та функціонуванням з модуляцією QAM високого рівня в умовах. Спектральна ефективність дорівнює добутку швидкості передачі бітів і порядку модуляції (для QAM (англ. Quadrature Amplitude Modulation) - 64 це 6 - один символ переносить інформації шести бітів). В мережах мобільного зв’язку використовується середня спектральна

ефективність, яка визначається як пропускна здатність всіх користувачів поділена на ширину смуги пропускання та кількість стільників (біт/с/Гц/стільник). Пропускна здатність мережі (біт/с/Гц) оцінюється даними від не менше 95% користувачів.

Еволюція LTE дозволила отримати нову якість системи у вузькосмуговому спектрі для застосування в мережах Інтернету речей. Стандартизовані такі технології для IoT систем, як: LTE-MTC (англ. LTE-Machine-Type Communication), EC-GSM-IoT (англ. Extended Coverage-GSM-IoT), NB-IoT (англ. Narrrow Band-IoT). Основна мета LTE розширень для IoT систем полягає в створенні спеціальних конфігурацій стандарту мобільного зв'язку, спрямованих на забезпечення енергоефективності та довготривалого функціонування пристроїв від автономних джерел живлення. На сьогоднішній день технологією з найбільш поширеним покриттям є NB-IoT [89]. Це пояснюється тим, що для впровадження даної технології не потрібні значні інфраструктурні зміни в існуючих LTE мережах.

Розгортання інфраструктури NB-IoT зводиться до оновлення програмного забезпечення на наявних базових станціях [90]. Тому в даному дослідженні ми розглядаємо саме цю технологію і проводимо практичний експеримент в міських умовах для визначення її переваг і недоліків в порівнянні з класичним LTE з'єднанням.

У мережі 4G LTE смуга передачі даних розділяється на окремі канали (піднесучі), кожен з яких має ширину 15 кГц. На кожен канал припадає ресурсний блок (РБ), що складається з 12 піднесучих по 15 кГц. Отже, загальна ширина смуги 4G LTE складає 180 кГц. Кожен ресурсний блок поділяється на 84 ресурсних елементи (сітку 12 на 7) [90]. У випадку з мережею NB-IoT для передачі використовується лише один ресурсний блок шириною 180 кГц (рис. 2.7). Для забезпечення дальнього охоплення сигнал усилюється на 6 дБ. Це пояснює більший радіус дії мережі Інтернету речей в порівнянні з LTE, незважаючи на однаковий частотний діапазон використання [90].

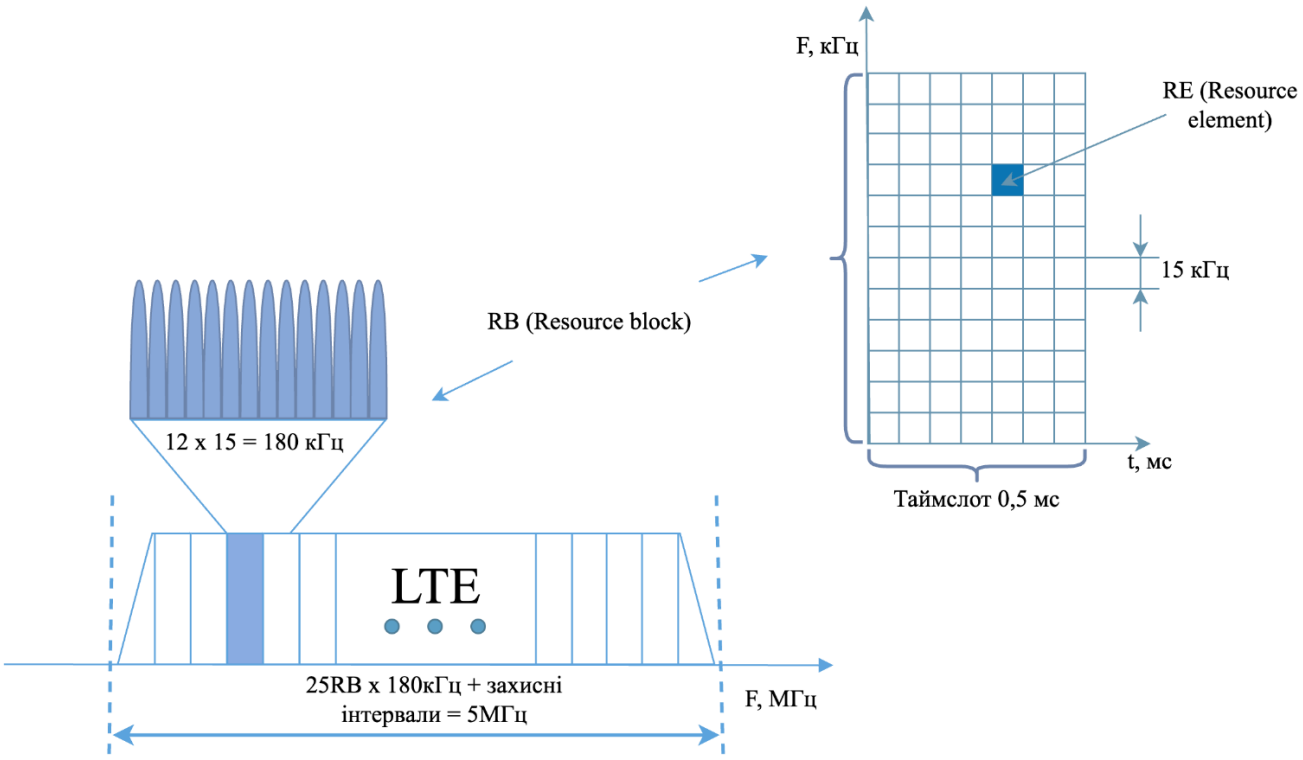


Рис. 2.7. Структура NB-IoT мережі [90]

Запуск мережі може здійснюватися трьома способами [87, 90]:

1. На відведеному каналі шириною 200 кГц, який також включає захисний спектр у діапазоні 300-600 кГц, для уникнення можливих перешкод. Даний метод є неактуальним для України через недостатність частотних ресурсів;

2. З використанням захисних смуг частот: в кожній смузі, де оператори розгортають комунікаційні системи, виділяється додатковий захисний інтервал завширшки 500 кГц з кожного боку;

3. В межах вже виділеної смуги (цей варіант актуальний для України).

2.2.2 Проведення практичного експерименту. Тестування NB-IoT в міських умовах.

Метою даного експерименту є визначення та порівняння продуктивності LTE та NB-IoT з точки зору швидкості передачі даних, стійкості до завад та енергоспоживання в реальних умовах. Тестування проводилось на основі

розробленого прототипу IoT модуля який передає дані на сервер через LTE та NB-IoT модуль в режимі реального часу. Експеримент проводився на тролейбусному маршруті у м. Київ.

Завдання експерименту:

- Оцінити та порівняти рівень потужності сигналу (RSSI - Received Signal Strength Indication), а також рівень сигнал/шум (SNR, dB) в залежності від відстані від базової станції.
- Оцінити та порівняти швидкість передачі даних.
- Визначити та порівняти коефіцієнт втрачених пакетів в залежності від RSSI.
- Оцінити та порівняти споживання енергії: виміряти енергоспоживання для різних сценаріїв та в різних режимах передачі.
- Оцінити вищенаведені параметри в русі, при зміні відстані від базової станції.

Прототип автомобільного IoT модуля було розроблено на базі мікрокомп'ютера Raspberry PI 3 (рис. 2.8) з ОС Debian а також модему Teltonika TRM 250 (рис. 2.9), який підтримує LTE та NB – IoT стандарти для передачі даних.

Рис. 2.8. Raspberry PI 3

Рис. 2.9. Модем Teltonika TRM 250

В ході експерименту було застосовано 2 вищенаведених модулі: з активним LTE, та з активним NB-IoT з'єднанням які в реальному часі передавали

значення координат, температури, енергоспоживання, а також рівень RSSI. На Рисунку 2.10 наведено загальну схему експерименту.

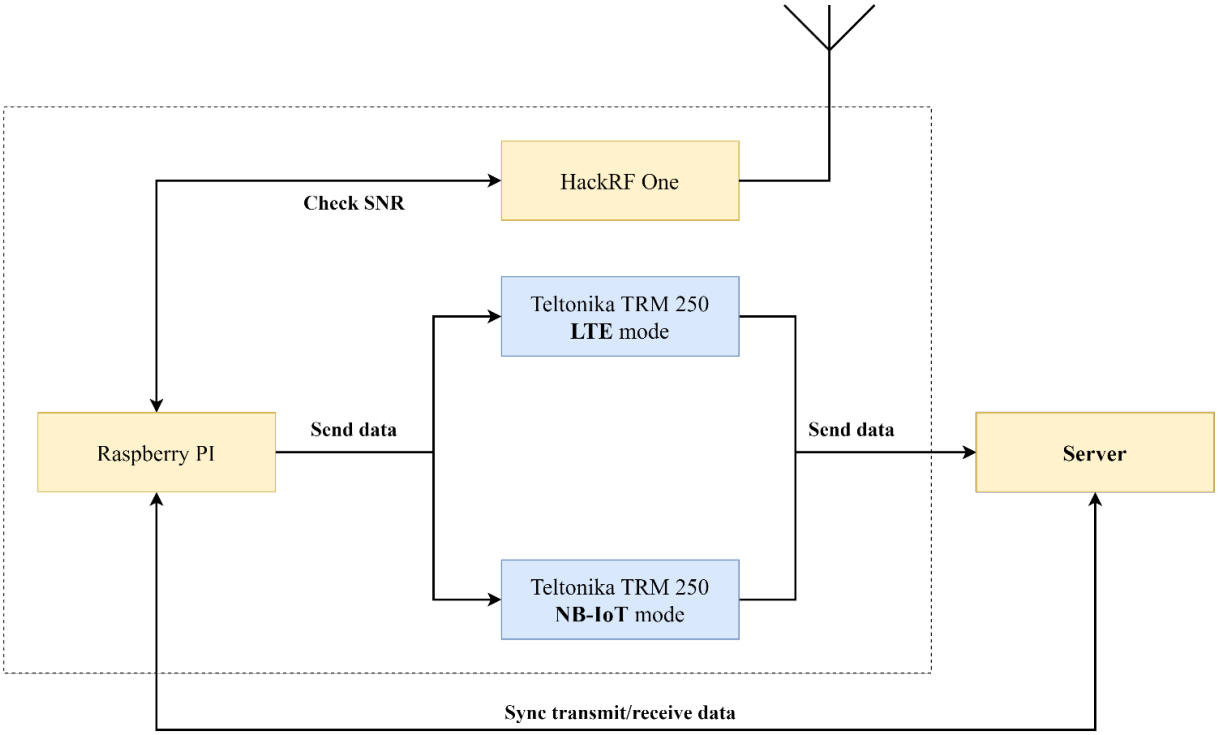


Рис. 2.10. Загальна схема експерименту

Для оцінки рівня сигнал-шум в експерименті була використана платформа HackRF One (рис. 2.11).

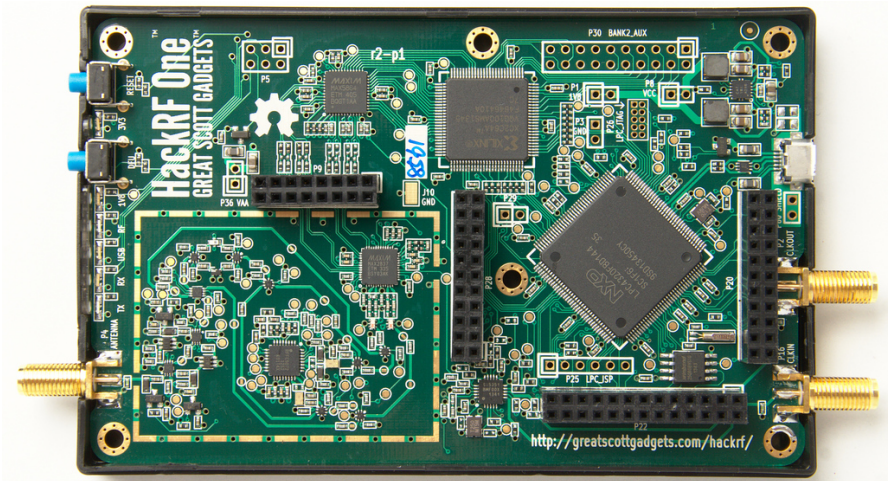


Рис. 2.11. HackRF One

Для проведення експерименту дані про рівень сигналу були фіксовані та записувалися безпосередньо на IoT модулі. Після завершення експерименту ці

дані були об'єднані з вимірюваннями, які були отримані на центральному сервері. Алгоритм роботи IoT модуля наведено на Рисунку 2.12.

Algorithm 1: Automotive controller firmware

```
while Network is not connected do
    Init Nb-IoT connection...
end
Connect to MQTT Gateway...
if mqtt connection then
    Subscribe to MQTT topics → {geolocation, climate, fuel, passengers}
end
while true do
    if intervalEvent then
        Read / Publish data → {geolocation, climate, fuel} topic
    end
    if stopEvent then
        if stopLocation in (currentPosition + precisionDistance) region then
            Listen passengers module delay(60s) then
            Publish data → passengers topic
        end
    end
end
```

Рис. 2.12. Алгоритм роботи IoT модуля

Результати експерименту

В ході експерименту було отримано рівень сигналу та відношення сигнал/шум на двох тролейбусних маршрутах. Зміна рівня сигналу на даних маршрутах наведена на рисунках 2.13 і 2.14.

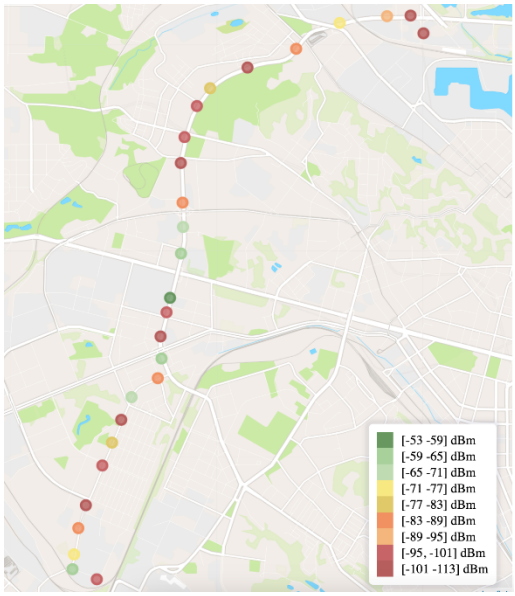


Рис. 2.13. Маршрут 1 - Співвідношення сигнал/шум.

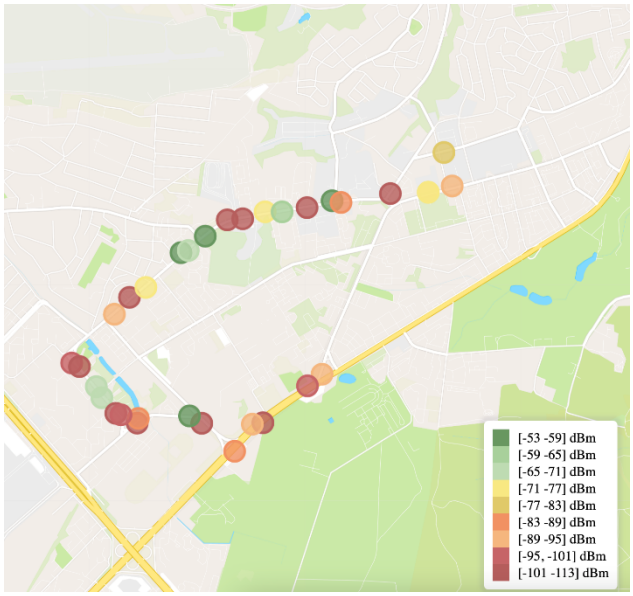


Рис. 2.14. Маршрут 2 - Співвідношення сигнал/шум.

Результати експерименту (таб. 2.5) показали, що енергоспоживання в LTE та NB-IoT з'єднаннях є лінійно залежним від рівня сигналу. Ця залежність характеризується тим, що зі збільшенням рівня сигналу споживана передавачем енергія також зростає.

Таблиця 2.5. Залежність енергоспоживання від рівня сигналу

Номер вимірювання	Характеристика		
	RSSI (dBm)	Енергоспоживання LTE, (мА)	Енергоспоживання NB-IoT, (мА)
1	-58	54.577	16.977
2	-92	35.382	13.616
3	-80	42.157	14.864
4	-79	42.720	15.160
5	-77	43.849	14.076

6	-82	41.027	15.11
7	-70	47.803	14.891
8	-98	31.995	12.469
9	-79	42.721	14.741
10	-70	47.803	16.148

Середнє енергоспоживання для NB-IoT з'єднання під час експериментального періоду було в 3 рази меншим, ніж у випадку з LTE (рис. 2.15). Це підтверджує заявлену енергоефективність NB-IoT [87]. Ця характеристика робить її більш підходящою для вбудованих систем, що працюють від автономних джерел живлення та вимагають низького енергоспоживання.

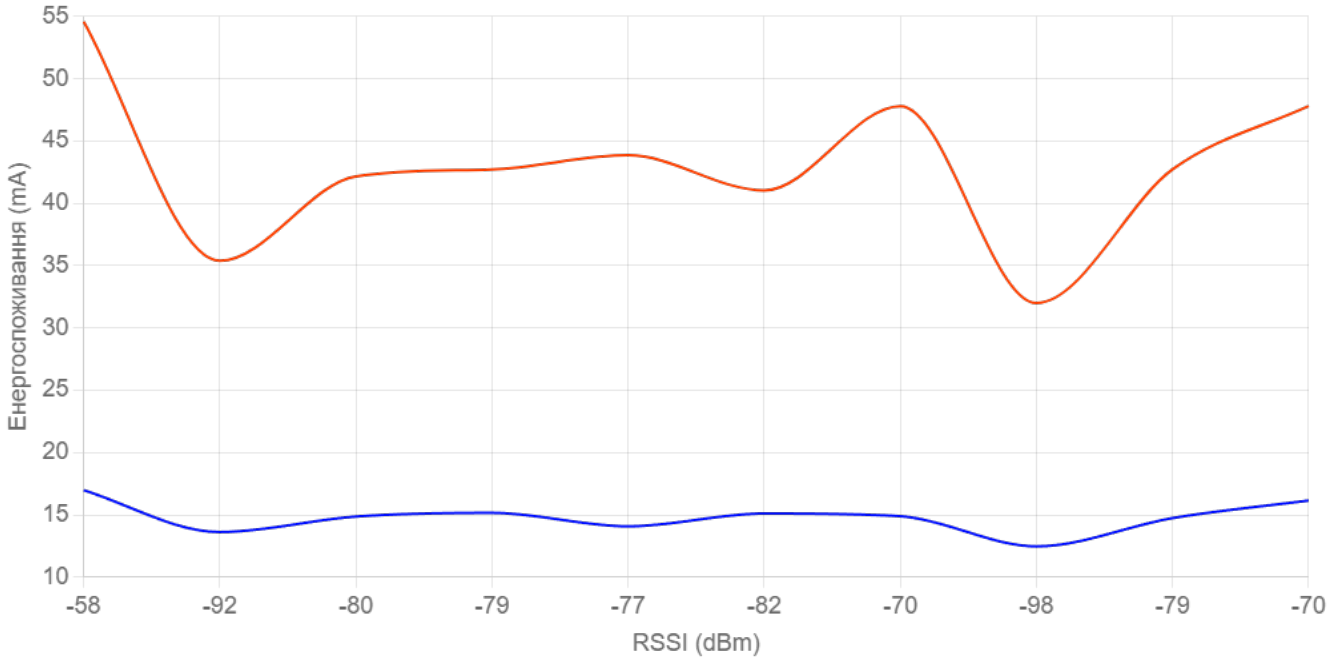


Рис. 2.15. Залежність енергоспоживання від рівня сигналу

Також, в ході експерименту визначено залежність швидкості передачі даних та коефіцієнту втрачених пакетів від рівня сигналу. Результати наведено у Таблиці. 2.6.

Таблиця 2.6. Залежність параметрів передачі даних від рівня сигналу

Номер вимірювання	RSSI (dBm)	LTE		NB-IoT	
		Швидкість передачі (Кбіт/с)	Коефіцієнт втрачених пакетів, %	Швидкість передачі (Кбіт/с)	Коефіцієнт втрачених пакетів, %
1	-58	33457.468	0.594	70.760	0.182
2	-92	14969.945	7.330	54.717	2.252
3	-80	21495.953	4.952	60.673	1.521
4	-79	22037.704	4.754	62.086	1.460
5	-77	23125.205	4.358	56.913	1.339
6	-82	20407.452	5.348	61.847	1.643
7	-70	26933.460	2.971	60.804	0.913
8	-98	11708.441	8.518	49.239	2.617
9	-79	22038.704	4.754	61.191	1.374
10	-70	29843.451	2.214	62.804	0.913

Отримані результати показали, що LTE з’єднання є більш чутливим до коливань сигналу, ніж NB-IoT (рис. 2.16) при тих самих умовах.

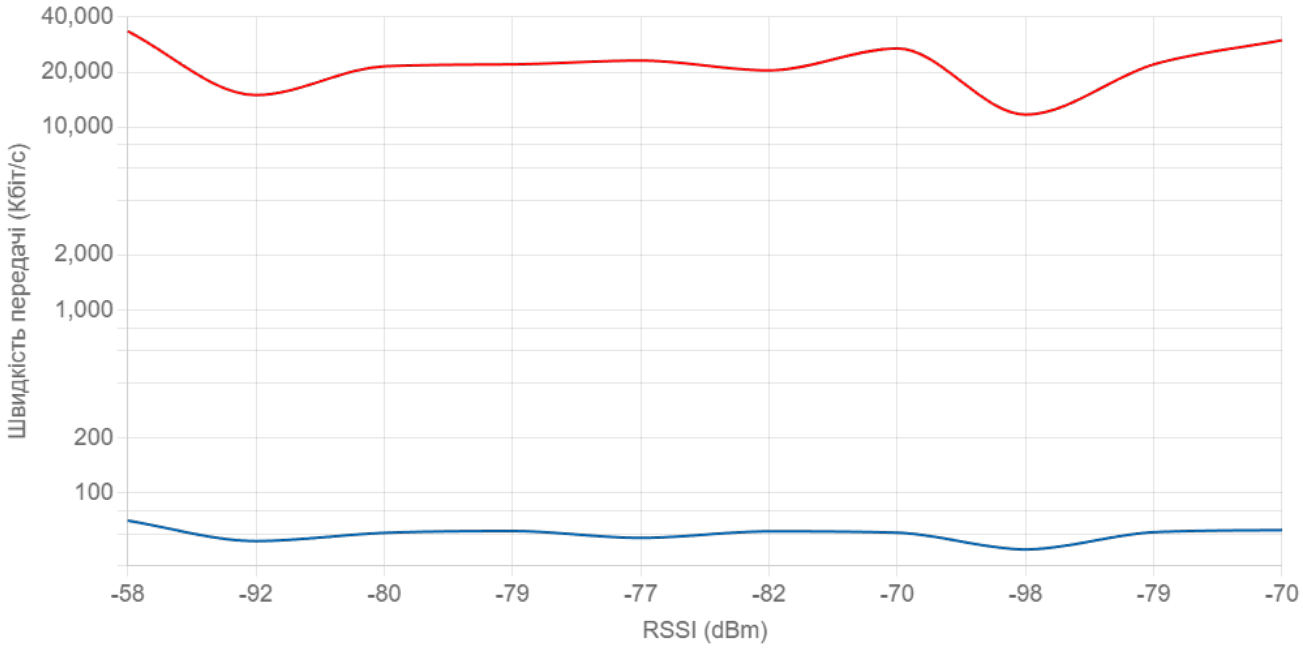


Рис. 2.16. Залежність швидкості передачі даних від рівня сигналу

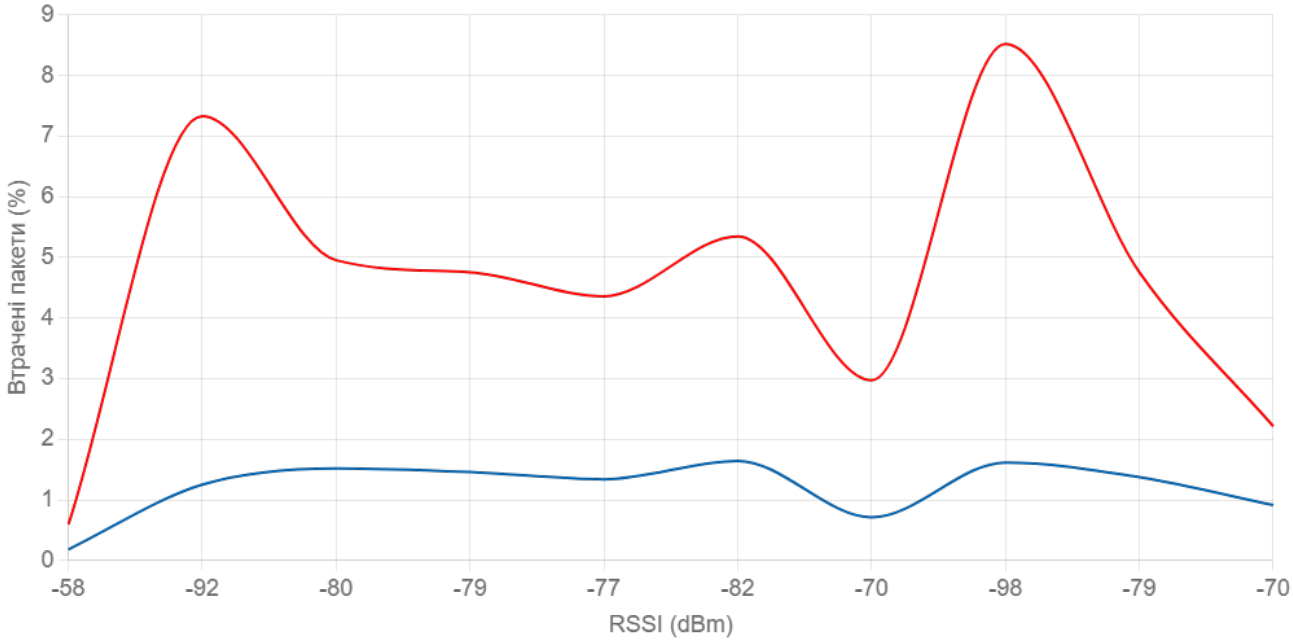


Рис. 2.17. Залежність коефіцієнту втрачених пакетів від рівня сигналу

Незважаючи на меншу швидкість передачі даних у порівнянні з LTE, NB-IoT забезпечує енергоефективність та стійкость до зниження рівня сигналу. Тому для досліджуваної системи NB-IoT є оптимальною технологією передачі даних між IoT пристроєм та сервером.

2.3 Аналіз та обґрунтування вибору протоколів програмного рівня. Проведення практичних експериментів.

Комунікацію IoT системи умовно можна розділити на дві частини: зв’язок між пристроями та сервером та зв’язок між програмними сервісами обробки та аналітики даних (програмний рівень).

У цьому розділі буде проведено аналіз, а також розроблено середовище тестування для оцінки продуктивності протоколів програмного рівня. Запропоноване середовище дозволяє емулювати IoT систему з низькою затримкою мережі, що дозволяє ефективно оцінити та порівняти протоколи, а також доцільність їх використання у тих чи інших ситуаціях. В даному випадку,

тестування буде проводитись на основі автомобільного IoT модуля, який використовувався у розділі 2.3.2.

2.3.1. Протоколи для передачі даних між IoT пристроєм та сервером.

У сфері M2M/IoT існує декілька основних протоколів, які призначені для передані даних між IoT пристроєм та сервером [91]. Основними є та протоколи, побудовані на основі HTTP/HTTP2, такі як HTTP та COAP.

У даному дослідженні проводиться оцінка продуктивності для різних типів та структур даних, та різних розмірів пакету. Оцінка проводиться за такими показниками:

- Пропускна здатність (Mbps): Визначає, яка кількість даних може бути передана через мережу за одиницю часу, вимірювана в мегабітах на секунду (Мб/с).
- Пропускна здатність - повідомлень (10 Кб) в секунду: Відображає, скільки повідомлень об'ємом 10 кілобайт можуть бути передані через мережу за одну секунду. Цей показник визначається для оцінки ефективності передачі коротких повідомлень, які часто використовуються у IoT мережах.
- Час на відправку/доставку 1М повідомлень (10 Кб): Вказує на час, необхідний для відправлення та доставки одного мільйона повідомлень об'ємом 10 кілобайт.
- Тест на відправку файлів різних розмірів (1-100 Мб): Цей тест дозволяє визначити, наскільки ефективно мережа обробляє великі файли різних розмірів, від 1 до 100 мегабайт.

Результати експерименту

Кількість переданих повідомлень Отримані результати (рис. 2.18) демонструють значну перевагу протоколу MQTT над HTTP та COAP для передачі великої кількості повідомлень протягом короткого періоду часу. В системі на базі MQTT протоколу було відправлено/доставлено 10038 повідомлень за 1 секунду, а в системах на базі MQTT та COAP 137 і 341 відповідно. Дана різниця

обумовлена моделлю роботи даних протоколів. З'єднання MQTT протоколу встановлюється один раз, та є активним протягом усього сеансу передачі. Натомість HTTP та COAP для кожної передачі встановлюють нове з'єднання, що збільшує загальний час, що необхідний для відправки даних.

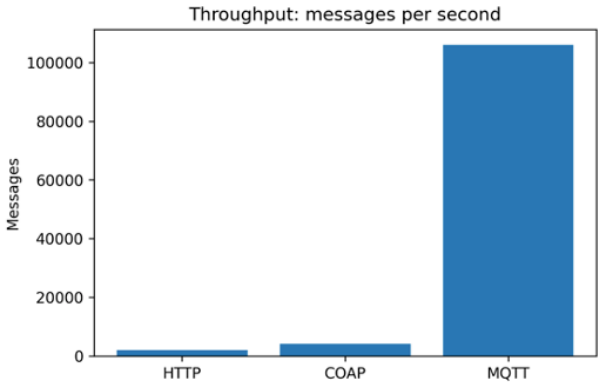


Рис. 2.18. Кількість переданих повідомлень в секунду

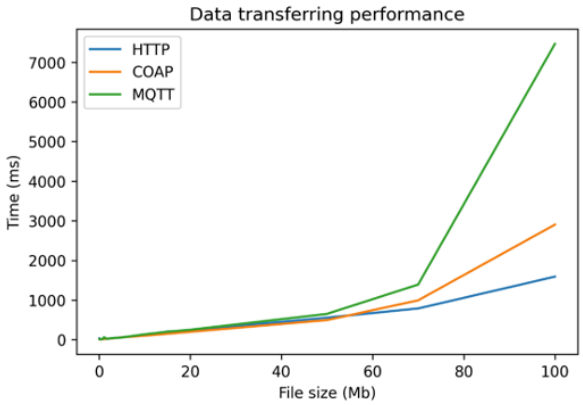


Рис. 2.19. Залежність пропускної здатності від розміру файлу

Залежність пропускної здатності від розміру файлу. На Рисунку 2.19 наведено залежність пропускної здатності протоколу від розміру даних, що передаються. Отримані результати показали, що при збільшенні розміру файлу, продуктивність передачі (об'єм даних переданий за одиницю часу) погіршується. Для протоколів HTTP та COAP час, необхідний для відправки 1 Мб. протягом періоду експерименту змінювався в діапазоні 400 мс. – 3 с., що є задовільним результатом. Для MQTT протоколу, вищенаведений показник змінювався від 300 мс до 7.4 с.. Проаналізувавши графік (рис. 2.19) можна зробити висновок, що MQTT протокол, є ефективним для передачі даних до ~ 65 Мб.

Час на відправку 1М повідомлень. Результати даного експерименту (2.20) корелюють з результатами (2.18), та підтверджують значну перевагу MQTT протоколу для відправки повідомлень невеликого об'єму (10 Кб.) над HTTP та COAP. Для відправки 1 мільйона повідомлень, системі на основі MQTT протоколу необхідно 10.77 секунди, натомість для системи на основі HTTP та COAP – 9 та 7.5 хвилин відповідно.

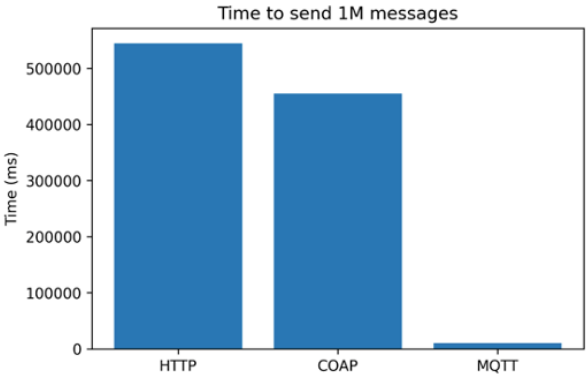


Рис. 2.20. Час на відправку 1M повідомлень

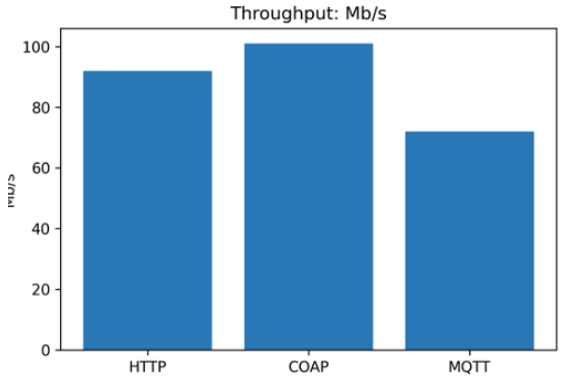


Рис. 2.21. Пропускна здатність (Мб/с)

Пропускна здатність. Для відправки повідомлень об’ємом до 50 Мб. пропускна здатність досліджуваних протоколів є ідентичною. Незначна перевага протоколів на основі HTTP з’єднання, обумовлена можливістю передачі повідомлення єдиним пакетом.

Оскільки, в телекомунікаційних мережах систем Інтернету речей, для обміну інформацією використовуються короткі повідомлення (до 1 Мб), то на основі одержаних результатів, можна зробити висновок, що MQTT є оптимальним протоколом програмного рівня для передачі даних в даних мережах

2.3.2. Протоколи мікросервісної комунікації

Кожен компонент системи в мікросервісній архітектурі може бути розподілений між кількома веб-серверами або доменами. Залежно від типу і структури даних повинні бути реалізовані методи зв'язку, які повинні базуватися на існуючих протоколах передачі даних.

На основі моделі роботи протоколів, їх можна розділити на синхронні та асинхронні (таб. 2.7) [92].

Проведено тести для найпопулярніших протоколів програмного рівня: HTTP, MQTT, AMQP і GRPC. Оцінювання продуктивності проводилося на основі таких показників як: пропускна здатність, паралельність, масштабованість, та ініціалізація з’єднання.

Таблиця 2.7. Протоколи програмного рівня

Протокол	**Модель**	**Архітектура**	**Тип даних**	**Механізми захисту**	**Доставка повідомлень**
HTTP/HTTPS	Синхронний	Клієнт-Сервер	Текстовий	SSL/TLS	Не гарантовано
HTTP2	Асинхронний	Клієнт-Сервер	Двійковий	SSL/TLS	Не гарантовано
AMQP	Асинхронний	Видавець-Підписник	Двійковий	SSL/TLS та SASL	Гарантовано
MQTT	Асинхронний	Видавець-Підписник	Двійковий	SSL/TLS	Гарантовано
gRPC	Асинхронний	Клієнт-Сервер	Двійковий	SSL/TLS	Гарантовано

Сценарій експерименту. Для тестування продуктивності вищенаведених протоколів було розроблене програмне забезпечення (алгоритм роботи - додаток), яке відтворює алгоритм роботи IoT модуля (Рис. 8) та виконує передачу даних на сервер з використанням наступних протоколів: HTTP, AMQP, MQTT і GRPC. Характеристики сервера, який використовувався під-час експерименту наведені в Таблиці 2.8.

Таблиця 2.8. Характеристики сервера

Параметр	**Значення**
Instance type	t2.large
CPU	3.3 GHz Intel Xeon** Processor
vCPU	2
Mem (GiB)	8
Storage	EBS-Only
Network Performance	Moderate

Протоколи HTTP та GRPC використовують клієнт-серверну архітектуру (рис. 2.22), де клієнт ініціює запит і чекає відповіді від сервера. Для проведення експерименту було створено два екземпляри (Таб. 7): один для сервера та один для клієнта.

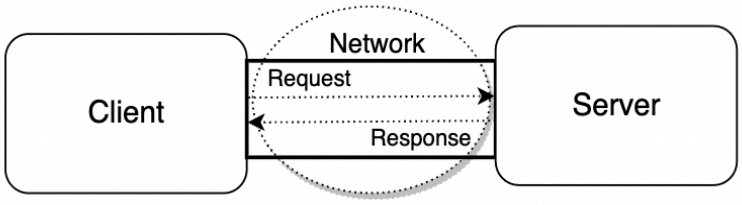


Рис. 2.22. Клієнт – Серверна архітектура

Протоколи AMQP та MQTT використовують архітектуру Видавець - Підписник (рис. 2.23), згідно з якою обмін повідомленнями складається з трьох частин - видавець надсилає повідомлення брокеру, брокер створює чергу, а підписник підписується на тему та отримує повідомлення від брокера.

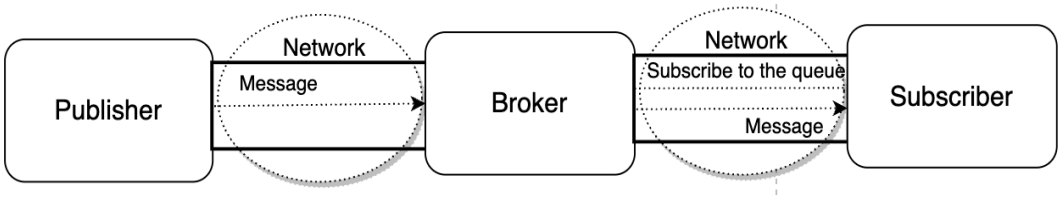


Рис. 2.23. Архітектура Видавець – Підписник

У даному експерименті використовується три екземпляри сервера: для видавця, підписника та брокера повідомлень. Щоб максимально зменшити вплив затримки мережі, під час експерименту всі сервери були запущені в одній підмережі.

Результати експерименту. Час ініціалізації з'єднання - час, необхідний для встановлення початкового підключення TCP а також ініціалізації SSL. Результати (рис. 2.24) суттєво не відрізняються для усіх розглянути протоколів. Але слід зазначити, що для протоколів Клієнт - Серверної архітектури для кожного запиту встановлюватиметься нове з'єднання (тобто при кожному запиті, загальний час буде зростати на величину часу ініціалізації), натомість для інших протоколів ця дія буде виконана лише один раз.

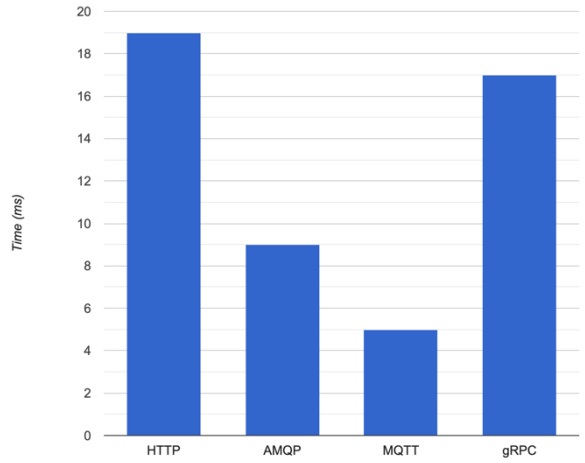


Рис. 2.24. Час ініціалізації з’єднання

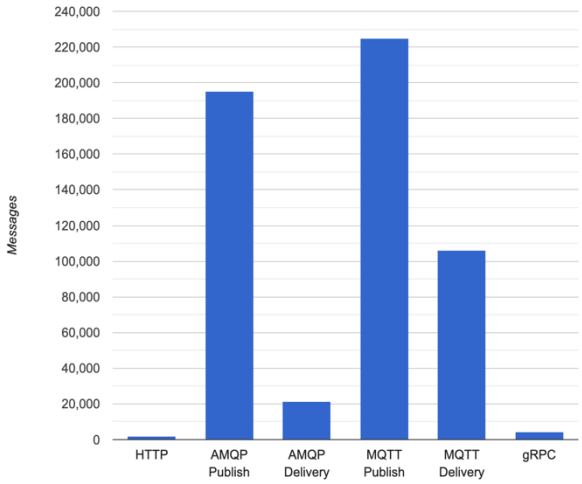


Рис. 2.25. Кількість запитів/повідомлень в секунду

Пропускна здатність — це показник того, скільки запитів сервер може обробити за секунду. Результати цього тесту (рис. 2.25) показують, що протоколи MQTT/AMQP мають значну перевагу над HTTP/gRPC. Даний результат пояснюється тим, що при використанні архітектури видавець - підписник при відправці нового повідомлення, використовується вже існуюче з’єднання, замість ініціалізації нового.

Час на відправку 1M повідомлень - результат тесту (рис. 2.26) корелює з (Рис. 2.25) і демонструє значну перевагу MQTT/AMQP над HTTP/gRPC.

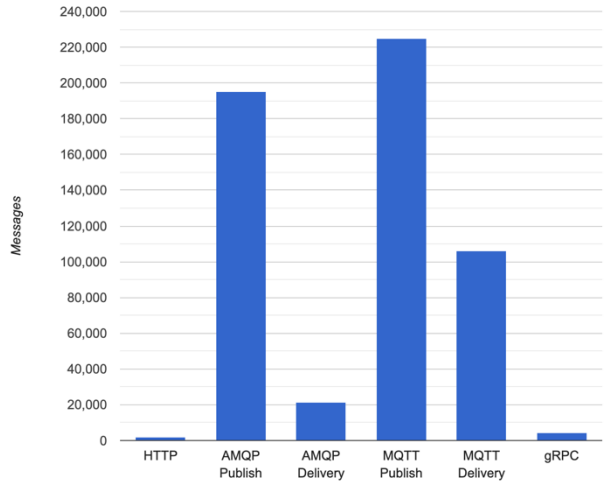


Рис. 2.26. Час на відправку 1M повідомлень

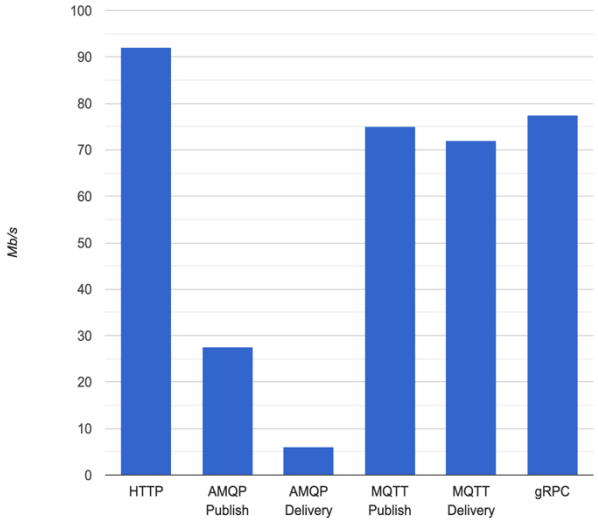


Рис. 2.27. Пропускна здатність (Мб/с)

Передача файлів різних розмірів. В даному експерименті (рис. 2.27 – 2.28) використовувалися файли розміром від 0,1 до 500 мегабайт. Оскільки, для деяких протоколів існує обмеження на максимальний розмір повідомлення, то повідомлення, які перевищували цей ліміт, передавались кількома пакетами.

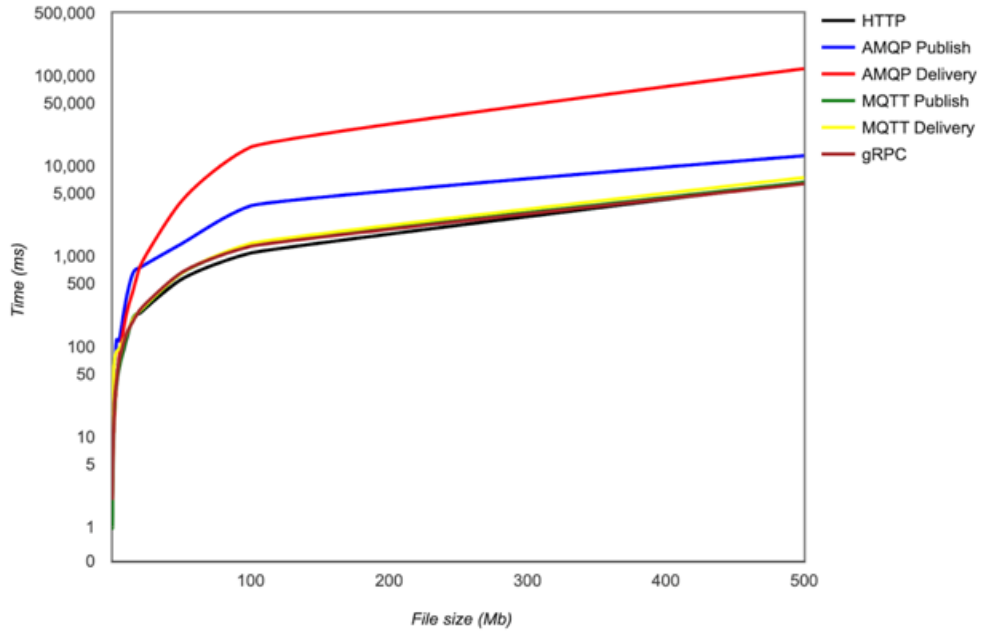


Рис. 2.28. Залежність пропускної здатності від розміру файлу

2.4. Моделювання поведінки IoT системи на основі розподіленої архітектури

Розглянемо неоднорідну систему Інтернету речей, яка складається з N підключених пристроїв, кожен з яких генерує навантаження відповідно до неоднорідного Пуассонівського процесу з функцією інтенсивності $\lambda(t)$ подій. Нехай система обробки складається з M серверів. Основною характеристикою серверу в контексті IoT системи є функція швидкості обробки даних $f_m(x)$, де x – навантаження на нього. Тоді час обробки завдання, яке генерує пристрій конкретним сервером R_m, можна отримати як згортку оберненої функції швидкості обробки:

$$R_m = \int_0^{\infty} \frac{1}{f_m(x)} * \lambda(t)dt, \qquad (2.1)$$

Загальна пропускна здатність X системи на основі розподіленої архітектури визначається шляхом оптимізації функції корисної дії системи U з врахуванням обмежень обчислювальної потужності серверів:

$$X = \arg\max_{X} U(X) \text{ за умови } \sum_{m=1}^{M} f_m^{-1}(X) \leq N, \quad (2.2)$$

Функція корисної дії системи $U(X)$ – є опуклою функцією, яка відображає залежність загальної продуктивності системи від пропускної здатності серверів - X. Дана функція дозволяє кількісно оцінити та порівняти різні конфігурації системи. В загальному вигляді, функція корисної дії може бути виражена як:

$$U(X) = \sum_{m=1}^{M} \log(1 + a_m X), \quad (2.3)$$

Де a_m – коефіцієнт серверу m, який відображає його вплив на систему в цілому.

Функція $\log(1 + a_m X)$ показує, що збільшення пропускної здатності кожного сервера, впливає на загальний коефіцієнт корисної дії системи. Логарифмічний характер цієї функції пояснюється тим, що в реальних системах при збільшенні кількості обчислювальних ресурсів, приріст коефіцієнта корисної дії є нелінійним, а величина його зростання зменшується з кожною інтеграцією збільшення ресурсів, оскільки система наближається до насичення та максимально ефективного використання.

Початкове зростання пропускної здатності системи мікросервісів X призводить до значного збільшення виразу $a_m X$, що призводить до значного приросту ефективності системи. Проте якщо $a_m X$ стає дуже великим, додавання одиниці до нього перед логарифмуванням має відносно незначний вплив.

Показники ефективності системи

Розподіл часу відповіді $(P_{R_m}(t))$. Функція розподілу ймовірностей часу обробки R_m для мікросервісу m:

$$P_{R_m}(t) = \int_0^t \frac{1}{f_m(x)} * \lambda(t-x)\,dx, \quad (2.4)$$

Область пропускної здатності ($\mathcal{X}$). Допустима область пропускної здатності системи обмежується функція швидкості обробки даних, та архітектурою мікросервісів (їх кількість, та вплив на коефіцієнт корисної дії системи в цілому):

$$\mathcal{X} = \{X \in R^M \mid \sum_{m=1}^{M} f_m^{-1}(X_m) \leq N,\ X_m \geq 0\}, \quad (2.5)$$

Ефективність системи (η):

$$\eta = \frac{U(X)}{N}, \quad (2.6)$$

Приклад моделювання системи:

Вхідні дані:

- N = 100 підключених IoT пристроїв
- M = 3 мікросервіси для обробки даних

Функція інтенсивності подій $\lambda(t)$ описує швидкість, з якою події (такі як повідомлення, пакети даних або запити) генеруються пристроями Інтернету речей у різні моменти часу. Дана функція надає спосіб описати, як часто події відбуваються з плином часу. Наприклад, визначимо функцію інтенсивності подій, як:

$$\lambda(t) = 10 + 5\sin(2\pi t), \quad (2.7)$$

В даному випадку, функція показує, що швидкість подій, створених пристроями IoT, змінюється з часом, починаючи з базової швидкості 10 подій на одиницю часу та коливаючись синусоїдально з амплітудою 5 подій.

Функції швидкості обробки даних показують, наскільки ефективно мікросервіс може обробити вхідне робоче навантаження, та вимірюється в кількості подій, оброблених за одиницю часу. Ці функції показуються, як

змінюється швидкість обробки подій мікросервісом залежно від робочого навантаження, яке він отримує. В математичному значенні, функція $f_m(x)$ відображає швидкість, з якою мікросервіс може виконати навантаження x.

Наприклад:

$f_1(x) = 20 + 5x$ – швидкість обробки, лінійно зростає з ростом навантаження

$f_2(x) = 15 + 3x^2$ - швидкість обробки, лінійно зростає квадратично з ростом навантаження

$f_3(x) = 25 + 2\sqrt{x}$ - швидкість обробки пропорційна квадратному кореню робочого навантаження

Коефіцієнти мікросервісів: $a_1 = 0.5, a_2 = 0.8, a_3 = 0.7$

Середній час відповіді мікросервісу 1:

$$R_1 = \int_0^{\infty} \frac{1}{20 + 5x} * (10 + 5\sin(2\pi t))\, dt, \tag{2.8}$$

Аналогічно, можна знайти R_3 та R_3 в залежності від f_2, f_3.

Для знаходження оптимального значення пропускної здатності X, яке максимізує функцію корисної дії системи, необхідно вирішити оптимізаційну проблему:

$$X = \arg\max_X U\,(X) \quad \text{за умови:} \quad \sum_{m=1}^{M} f_m^{-1}(X) \le N, \tag{2.9}$$

Для заданої функції $U(X) = \sum_{m=1}^{M} \log(1 + a_m X)$, можемо сформулювати Лагранжіан наступним чином:

$$\mathcal{L}(X, \lambda) = \sum_{m=1}^{M} \log(1 + a_m X) - \lambda\left(\sum_{m=1}^{M} f_m^{-1}(X) - N\right), \tag{2.10}$$

Взявши часткові похідні по X та λ та прирівнявши їх до нуля, можемо визначити оптимальні значення, які задовольняють умови.

Програмні реалізації для виконання обчислень, які використовуються у цьому методі наведені в додатках.

2.5 Висновки

1. На основі визначених вимог до системи розроблено структуру телекомунікаційної мережі та виділено інформаційні потоки системи. Вибір оптимальної технології бездротової передачі даних проведено на основі аналізу існуючих технологій та результатів експериментальних досліджень за розробленим алгоритмом.
2. Досліджено існуючі протоколи передачі даних програмного рівня в контексті Інтелектуальних систем громадського транспорту. З метою вибору протоколів розроблено методику та алгоритм оцінки продуктивності протоколу, програмно реалізовано тестове середовище. Проведено ряд практичних експериментів для обґрунтування вибору оптимальної технології передачі даних між IoT пристроєм та сервером, а також для мікросервісної комунікації.
3. Виконано програмну реалізацію компонентів телекомунікаційної мережі на основі концепції розподіленої мікросервісної архітектури. Для оцінки впливу архітектури, а також конфігурації окремих компонентів на пропускну здатність системи та її загальну продуктивність, розроблено математичну модель для моделювання поведінки IoT системи на основі розподіленої архітектури.

РОЗДІЛ 3. ЗБЕРЕЖЕННЯ ТА ОБРОБКА ДАНИХ ТЕЛЕКОМУНІКАЦІЙНОЇ МЕРЕЖІ

У даному розділі розглядається проблематика збереження та обробки даних телекомунікаційної мережі. Враховуючи зростаючу кількість даних, які генеруються та передаються в мережах, стає очевидним, що ефективне управління цими даними стає надзвичайно важливим завданням. Методи побудови системи збереження та обробки даних розглядаються з точки зору оптимізації використання обчислювальних ресурсів.

3.1 Забезпечення масштабованості та доступності системи

Навантаження, яке створюють IoT пристрої, є динамічним і недетермінованим і може бути високим (у час пік) або низьким (у час простою). В обох випадках серверна інфраструктура повинна адаптуватись до навантаження, яке створюють IoT клієнти. Система має стабільно працювати при великій кількості одночасно підключених пристроїв, а також не повинна використовувати надлишкові ресурси при малі кількості підключених клієнтів.

Система збереження та обробки даних може бути реалізована на основі монолітної або мікросервісної архітектури [93] (рис. 3.1). У монолітній архітектурі весь функціонал системи об'єднаний в одному цілісному блоку програмного коду. Всі компоненти та функції системи взаємодіють безпосередньо. Цей підхід може бути простим для початкової розробки та тестування, але має свої обмеження, оскільки при збільшенні навантаження вимагає вертикального масштабування, тобто збільшення обчислювальних ресурсів одного сервера чи вузла. У мікросервісній архітектурі функціонал системи розбивається на невеликі, незалежні сервіси, кожен з яких відповідає за конкретну функцію. Ці сервіси можуть взаємодіяти через API, і кожен з них може бути розгорнутий та масштабований окремо. Мікросервісна архітектура надає системі більшу гнучкість, масштабованість і доступність. При збільшенні обсягу

даних або навантаження на систему можна просто масштабувати лише ті сервіси, які цього потребують, не впливаючи на решту системи.

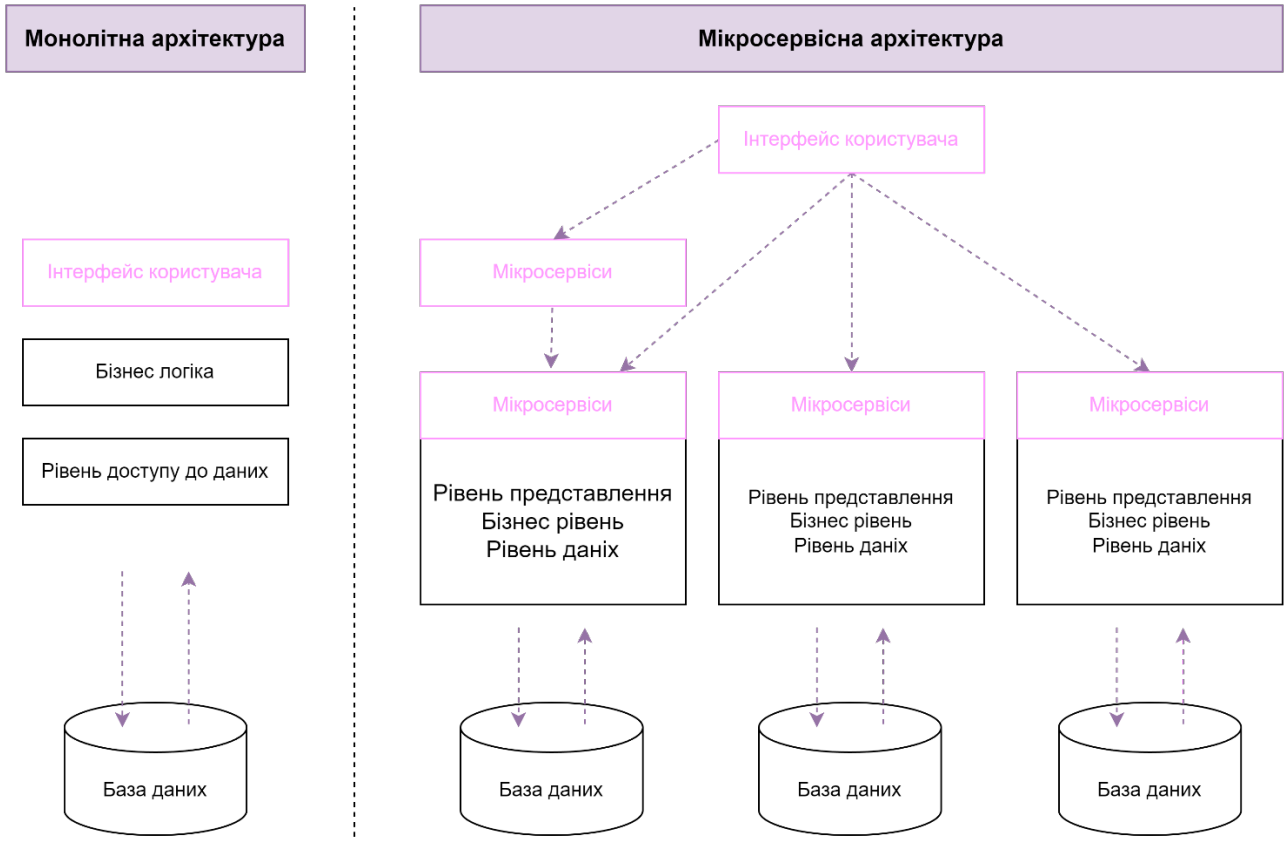


Рис 3.1. Типова схема використання ізольованих контейнерів

Розширення системи на основі мікросервісної архітектури відбувається на основі горизонтального масштабування, коли модулі розгортаються на різних серверах або в обчислювальних контейнерах, а їх кількість динамічно змінюється відповідно до навантаження.

3.1.1 Метод визначення оптимальної кількості обчислювальних контейнерів

Визначення оптимальної кількості обчислювальних контейнерів для мікросервісної архітектури в IoT системах має вирішальне значення для забезпечення економічної ефективності та якості функціонування таких систем. Значення кількості контейнерів може змінюватися відповідно до часових піків та змін в навантаженні. Оптимальне масштабування допомагає забезпечити швидку

та безперебійну реакцію на зростання або зменшення пасажиропотоку. Адаптація кількості контейнерів до поточних потреб допомагає уникнути зайвого споживання ресурсів, таких як енергія та обчислювальна потужність.

Обчислювальні контейнери працюють як абстракції вищого рівня на прикладному рівні, та дозволяють об'єднати програмний код та пов'язані з ним залежності. Завдяки можливості виконувати декілька контейнерів на одній машині при спільному використанні ядра та операційної системи (ОС), кожен контейнер функціонує автономно, як ізольований процес у просторі користувача. На відміну від віртуальних машин, контейнери потребують менших ресурсів, та часу на розгортання. Така компактність дозволяє розміщувати більшу кількість систем, потребуючи меншої кількості віртуальних машин і операційних систем, тим самим оптимізуючи використання ресурсів, що є особливо корисним в системах Інтернету речей. На Рис. 3.2 наведено типову схему контейнерних програм.

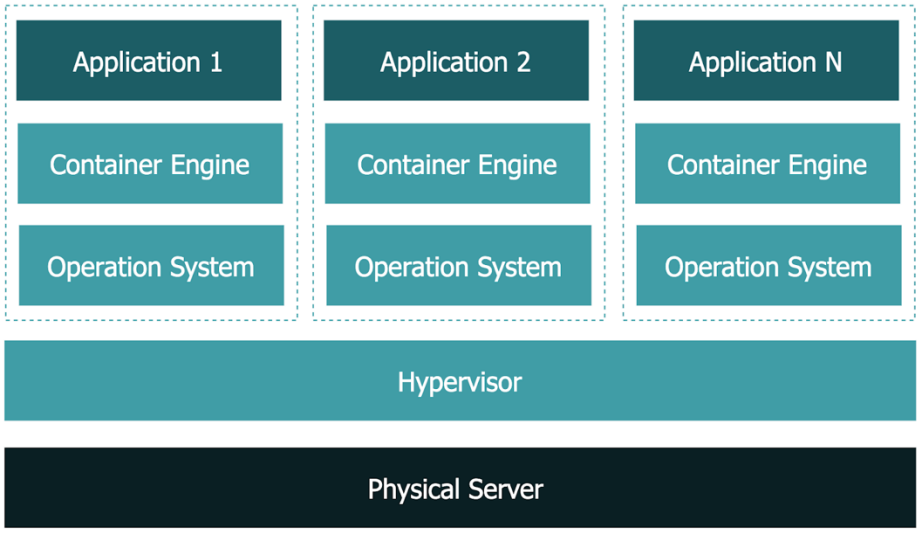


Рис 3.2. Типова схема використання ізольованих контейнерів

Для простоти ця схема представлена для одно серверної системи, але також контейнери можуть бути розподілені між кількома фізичними машинами для горизонтального масштабування. На Рисунку 3.3 представлена загальна схема розподіленої IoT системи на основі схеми з ізольованими контейнерами.

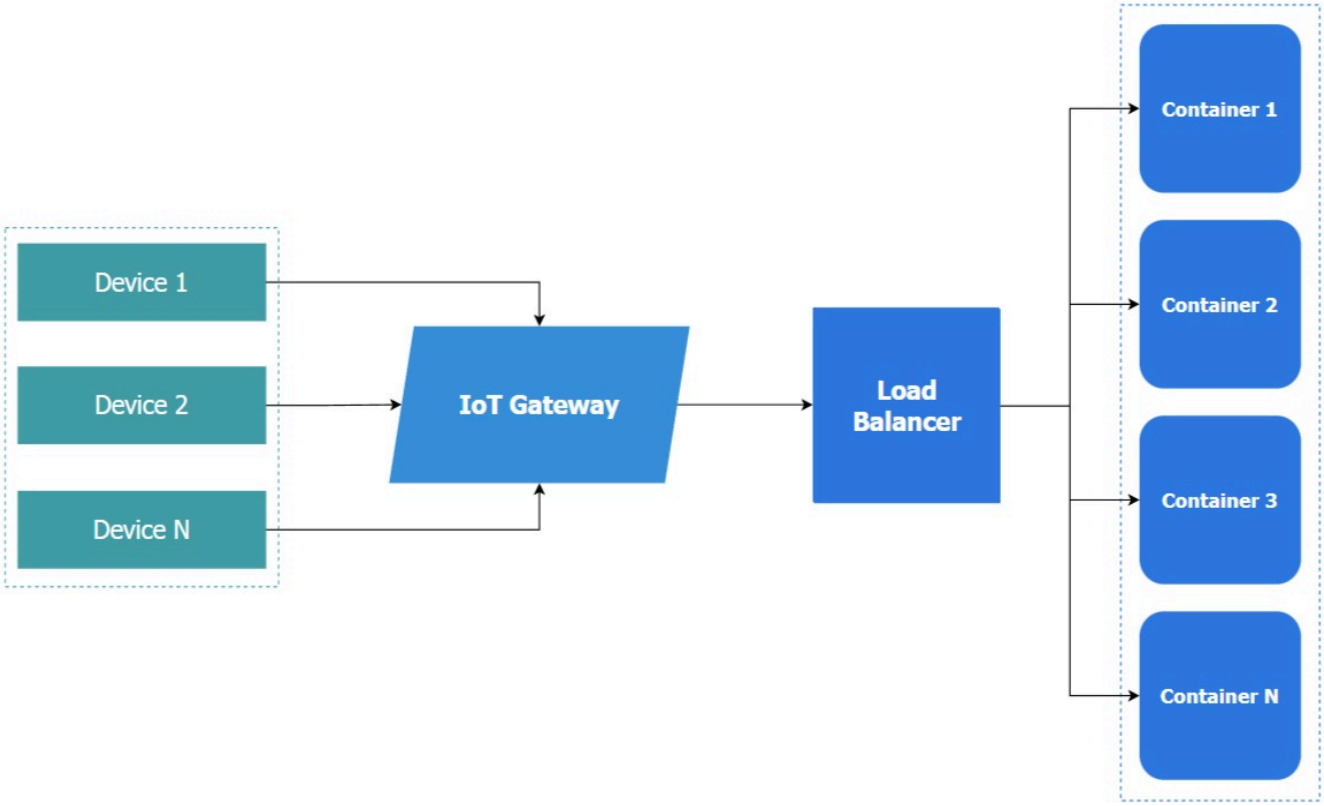


Рис 3.3. Схема розподіленої IoT системи

У цьому розділі будуть розглянуті існуючі методи для визначення оптимальної кількості обчислювальних контейнерів на основі теорії масового обслуговування та стохастичної моделі, а також буде запропоновано оптимізований метод для IoT систем на основі MILP (англ. Mixed-Integer Linear Programming) програмування.

Метод на основі теорії масового обслуговування. Дана модель дозволяє оцінити вплив кількості обчислювальних контейнерів на загальну продуктивність системи, за такими характеристиками, як: час відповіді та об’єм використаних ресурсів. Нижче, наведено узагальнену формулу визначення ймовірності і-кількості запитів в системі.

$$P_i = \left(1 + \frac{\lambda}{\mu}\right)^{-N} \times \frac{\sum_{i=0}^{N-i} \left(\frac{\lambda}{\mu}\right)^i}{i!} + \left(\frac{\lambda}{\mu}\right)^i \times \frac{N!}{N(\mu - \lambda)^N}, \quad (3.1)$$

Тоді, ймовірність відсутності запитів в системі, P_0, визначається підстановкою $i = 0$. Цей показник є ключовим параметром для оцінки продуктивності системи масового обслуговування, та дозволяє оцінити середню кількість запитів, а також середній час очікування у системі та в черзі.

В алгоритмі на базі даної моделі, швидкість надходження запитів (λ) є вхідним параметром. Він використовується в поєднанні зі швидкістю обслуговування (μ) і кількістю обчислювальних контейнерів (N) для розрахунку показників продуктивності, таких як середня кількість запитів у системі (L), середня кількість запитів у черзі (L_q), середній час очікування в системі (W), та середній час очікування в черзі (W_q).

Аналізуючи вплив швидкості надходження запитів на продуктивність системи, ми можемо визначити оптимальну кількість обчислювальних контейнерів, необхідних для ефективної обробки вхідних запитів відповідно до необхідної швидкості обслуговування:

1. Знайдемо швидкість обслуговування обчислювального контейнера:

$$\mu \leftarrow \frac{1}{S}, \tag{3.2}$$

де S – середня швидкість обробки запиту

2. Обчислимо швидкість надходження запитів в систему:

$$\lambda \leftarrow \frac{L}{W}, \tag{3.3}$$

де L – середня кількість запитів в системі, W – середній час очікування в системі

3. Обчислимо ймовірність відсутності запитів в системі P_0 на основі (1).
4. Обчислимо середню кількість запитів в системі:

$$L \leftarrow \frac{\lambda}{\mu - \lambda} \times P_0, \tag{3.4}$$

5. Обчислимо середню кількість запитів в черзі:

$$\mathrm{L_q} \leftarrow \frac{\lambda^2}{\mu(\mu - \lambda)} \times \mathrm{P_0}, \tag{3.5}$$

6. Обчислимо середній час очікування запиту в черзі:

$$\mathrm{W_q} \leftarrow \frac{\mathrm{L_q}}{\lambda}, \tag{3.6}$$

У цьому алгоритмі μ означає швидкість, з якою один обчислювальний контейнер може обробляти запити в поточному стані IoT системи. Це важливий параметр у теорії масового обслуговування.. Змінюючи кількість контейнерів (N) і спостерігаючи за відповідними змінами в показниках продуктивності, ми можемо проаналізувати вплив масштабування на здатність системи обробляти вхідні запити. Цей аналіз може надати уявлення про оптимальну кількість контейнерів, необхідних для досягнення бажаних цільових показників продуктивності, балансуючи між використанням ресурсів і часом відповіді.

Метод на основі стохастичної моделі. Нехай $X(t)$ стан системи в момент часу t, що визначає кількість активних обчислювальних контейнерів. Система може мати кінцеву кількість станів у межах від 0 до N, де N — максимальна кількість контейнерів.

Переходи станів відбуваються на основі швидкості надходження та обробки запитів та можуть бути змодельовані за допомогою ланцюгів Маркова. Швидкості переходу $a_{i,j}$, представляють ймовірність переходу зі стану i до стану j. Ці швидкості можна промоделювати за допомогою відповідних стохастичних моделей, наприклад Пуассонівських процесів. Побудувавши матрицю швидкості переходу $\mathbf{A}$ з елементами $a_{i,j}$, можна проаналізувати поведінку системи. Стаціонарні ймовірності ланцюга Маркова $\pi = [\pi_0, \pi_1, \ldots, \pi_N]$, визначають ймовірність перебування в кожному стані. Ці ймовірності задовольняють рівняння: $\pi \cdot \mathbf{A} = 0$, за умови $\sum_{i=0}^{N} \pi_i = 1$. Розв'язання цієї системи рівнянь дає змогу зрозуміти поведінку системи, наприклад використання контейнера,

пропускну здатність системи та розподіл ресурсів. Використовуючи ймовірності стаціонарного стану, можна вивести показники продуктивності для оцінки поведінки системи. Такі показники, як середня кількість контейнерів, середній час очікування та пропускна здатність системи, можна розрахувати на основі ймовірностей стаціонарного стану та швидкості переходу (рис. 3.4).

Algorithm 1 Containers Scaling Model

Input: N, λ , μ
Output: Steady-state probabilities π_i for $0 \leq i \leq N$
Construct the transition rate matrix $\mathbf{A}$ with size $(N+1) \times (N+1)$
for $i = 0$ to N **do**
  for $j = 0$ to N **do**
    if $i = j$ **then**
      $a_{i,j} \leftarrow -(\lambda + i\mu)$
    else if $j = i + 1$ **then**
      $a_{i,j} \leftarrow \lambda$
    end if
  end for
end for
Solve $\pi \cdot \mathbf{A} = \mathbf{0}$ subject to $\sum_{i=0}^{N} \pi_i = 1$
Return Steady-state probabilities π_i for $0 \leq i \leq N$

Рис. 3.4 – Алгоритм масштабування обчислювальних контейнерів на основі стохастичної моделі

Метод на основі MILP моделі. В даному випадку, завдання пошуку оптимальної кількості обчислювальних контейнерів буде сформульовано, як проблему математичної оптимізаційної моделі (рис. 3.5), зокрема задачі змішаного цілочисельного лінійного програмування (MILP). Цільова функція та обмеження визначені для вираження проблеми оптимізації, де метою є мінімізація кількості обчислювальних контейнерів, задовольняючи вимоги до швидкості обробки запитів.

Вхідні дані:

N – кількість активних IoT пристроїв.

P_i – вимоги обчислювальної ємності, які генеруються пристроєм i, $\forall_i \in [1, N]$.

T – коефіцієнт швидкості зв'язку.

C – ємність обчислювального контейнера.

CPU – кількість ядер процесора в обчислювальному контейнері.

M – об’єм оперативної пам’яті в обчислювальному контейнері

Вихідні дані:

S – мінімальна кількість обчислювальних контейнерів

Algorithm 2 Optimal Number of Computing Containers

Input: N, P_i, T, C, R, CPU, M
Output: S
Set $S \leftarrow 1$
Calculate total workload demand: $D \leftarrow \frac{1}{C \cdot S} \sum_{i=1}^{N} P_i$
Assign IoT devices to containers
Calculate maximum workload demand: $M \leftarrow \max\left\{\frac{1}{S} \sum_{i=1}^{N} P_i\right\}$
while $M > D$ **do**
    Increase number of containers: $S \leftarrow S + 1$
    Update workload demand: $D \leftarrow \frac{1}{C \cdot S} \sum_{i=1}^{N} P_i$
    Calculate maximum workload demand: $M \leftarrow \max\left\{\frac{1}{S} \sum_{i=1}^{N} P_i\right\}$
end while
return S

Рис. 3.5 – Алгоритм пошуку оптимальної кількості обчислювальних контейнерів на основі MILP моделі

3.2 Балансування навантаження підключених пристроїв Інтернету речей

Балансування навантаження (англ. load balancing) є критичним аспектом для ефективної та надійної роботи систем Інтернету речей. IoT системи включають в себе велику кількість підключених пристроїв та сенсорів, які взаємодіють між собою та з серверами через мережу. Тому виникає потреба в ефективному розподілі навантаження для забезпечення кращої продуктивності та доступності системи.

Перш за все, балансування навантаження допомагає розподілити завдання та обчислювальне навантаження рівномірно між різними серверами або вузлами.

Це дозволяє уникнути ситуацій, коли одні ресурси перевантажені, а інші - не використовуються повністю, забезпечуючи більш стабільну та швидку роботу системи.

По-друге, балансування навантаження впливає на швидкість та доступність системи. Завдяки цьому підходу можна досягти більш швидких відповідей на запити та знизити час недоступності. Рівномірне розподілення ресурсів допомагає запобігти ситуаціям, коли певні частини системи перевантажені, що може призвести до зниження продуктивності та якості обслуговування.

Третій аспект стосується ефективного використання ресурсів. Завдяки балансуванню навантаження досягається оптимальне розподілення обчислювальних та мережевих ресурсів. Це має позитивний вплив на економію коштів на інфраструктурі та сприяє енергозбереженню, оскільки недоексплуатовані ресурси не витрачають зайву енергію.

В контексті мікросервісної архітектури, необхідно забезпечити ефективний розподіл навантаження, між доступними обчислювальними контейнерами. У розділі 2 було проведено аналіз а також обґрунтовано доцільність застосування MQTT протоколу, як основного протоколу програмного рівня, для передачі даних між підключеними IoT пристроями та серверами, на яких буде відбуватись збереження та обробка даних. Тому завдання балансування навантаження, буде розглядатись в контексті архітектури запропонованої в розділах 2, та 3.1. А саме, розподіл навантаження, яке надходить від активних MQTT підписників, між доступними обчислювальними контейнерами.

3.2.1 Застосування існуючих методів балансування

В стандартному режимі роботи (рис. 3.6) MQTT протоколу кожен клієнт, який підписався на певну тему, отримує копію кожного повідомлення, що в ній публікується [94].

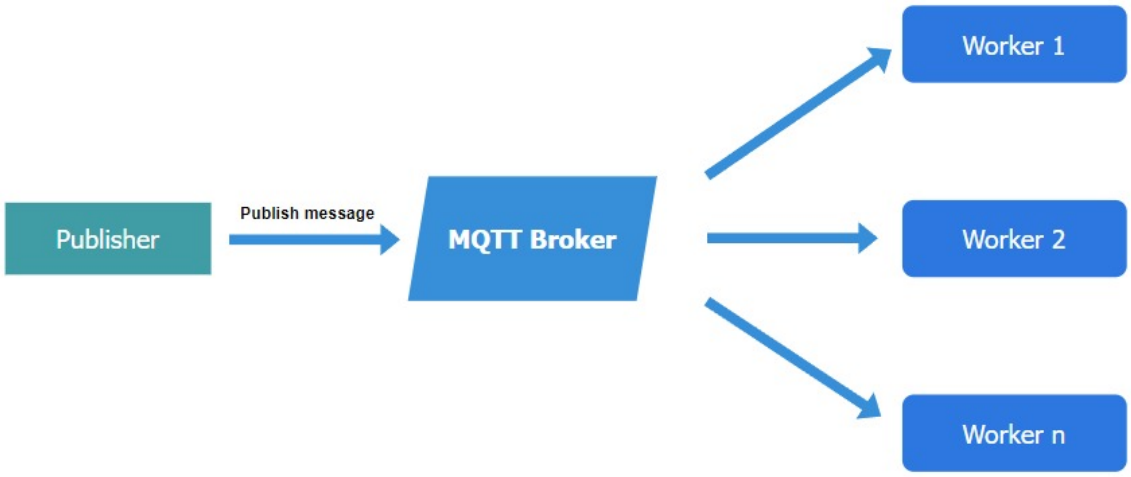


Рис 3.6. Механізм розподілу повідомлень MQTT протоколу в звичайному режимі

В такому випадку, для балансування навантаження, необхідно розподілити його по темах, та виконати підписку кожного обчислювального контейнера на конкретну тему. Але в такого підходу є певні недоліки, оскільки:

- Навантаження в IoT системах є недетермінованим та важкопрогнозованим, тому завчасний розподіл навантаження між обчислювальними контейнерами є неможливим в реальних системах, зокрема в системі, що розглядається.
- Завантаженість контейнерів буде нерівномірною.
- Розподіл навантаження за допомогою розділів, є неправильним в рамках концепції MQTT протоколу.

Особливо вищенаведені недоліки стосуються систем, де необхідно виконувати обробку великих об’ємів даних в режимі реального часу, зокрема в інтелектуальних транспортних системах.

Можливим варіантом вирішення є використання методу спільних підписок (англ. shared subscriptions), який з’явився в релізі MQTT 5 [95]. За допомогою методу спільних підписок (рис. 3.7) підписники, які мають підписку на спільну тему в одній групі, отримують повідомлення по черзі. Цей процес називають балансуванням навантаження клієнта.

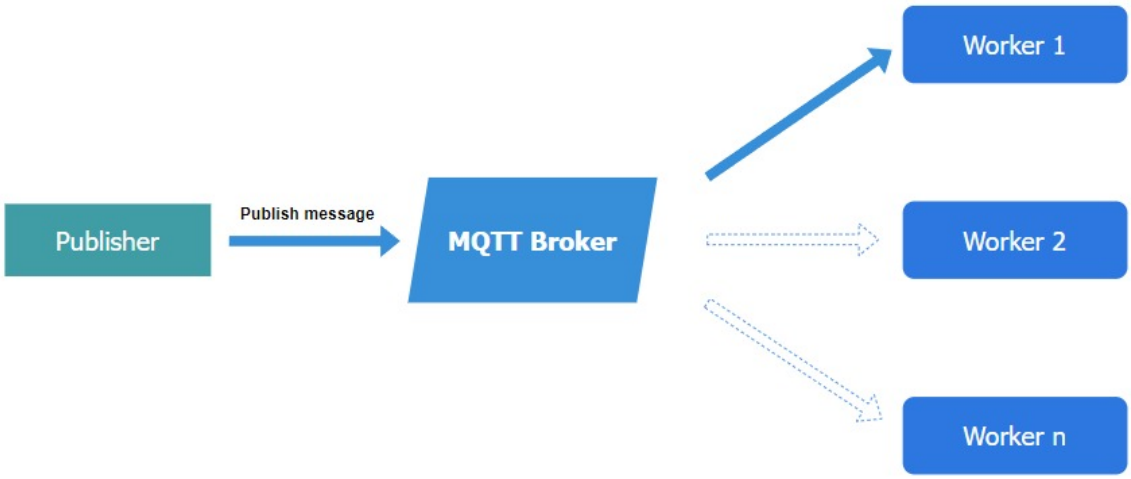


Рис 3.7. Механізм розподілу повідомлень MQTT протоколу в режимі спільних підписок

Але в даного підходу також є недоліки пов'язані з нерівномірним розподілом навантаження. Повідомлення отримані в різні моменти часу, в залежності від їх типу можуть потребувати різного типу програмної обробки, а отже, генерувати різне навантаження. Таким чином, будуть виникати ситуації, що частина обчислювальних контейнерів будуть повністю завантаженими, і не зможуть виконати частини завдань (або виконають з затримкою), а інша частина в той же час будуть не задіяними, або низько завантаженими.

Тому є необхідність в розробці ефективного алгоритму балансування навантаження, в залежності від поточного рівня завантаженості. Алгоритм має забезпечити рівномірний розподіл повідомлень між доступними контейнерами, та максимально ефективне використання серверних ресурсів.

3.2.2 Метод балансування навантаження в розподілених IoT системах на основі багатопараметричного моніторингу

Покращена архітектура MQTT брокера. Для оптимального розподілу навантаження між доступними серверами (в даному випадку обчислювальними контейнерами), MQTT брокер повинен отримувати інформацію про поточний стан завантаженості кожного доступного сервера, і виконувати маршрутизацію повідомлення до найменш завантаженого на даний момент.

На Рисунку 3.8 наведено покращену модель MQTT протоколу. В даній моделі, в MQTT брокері вводяться два додаткові модулі - *Utilization monitoring* та *Load Balancer*. В запропонованому варіанті, для системи моніторингу вводиться додатковий канал - Monitoring channel (TCP, 8123 порт). По вищенаведеному каналу MQTT брокер отримує інформацію про завантаженість кожного сервера по основних характеристиках, таких як: використання CPU, RAM, Disk, Network, і зберігає їх в спеціальний структуру - Utilization state, яку в свою чергу використовує Load balancer.

Load balancer це асинхронний модуль, який маршрутизує вхідні повідомлення від Publishers до Subcribers (обчислювальних контейнерів) по стандартному MQTT каналу (TCP, 8124 порт).

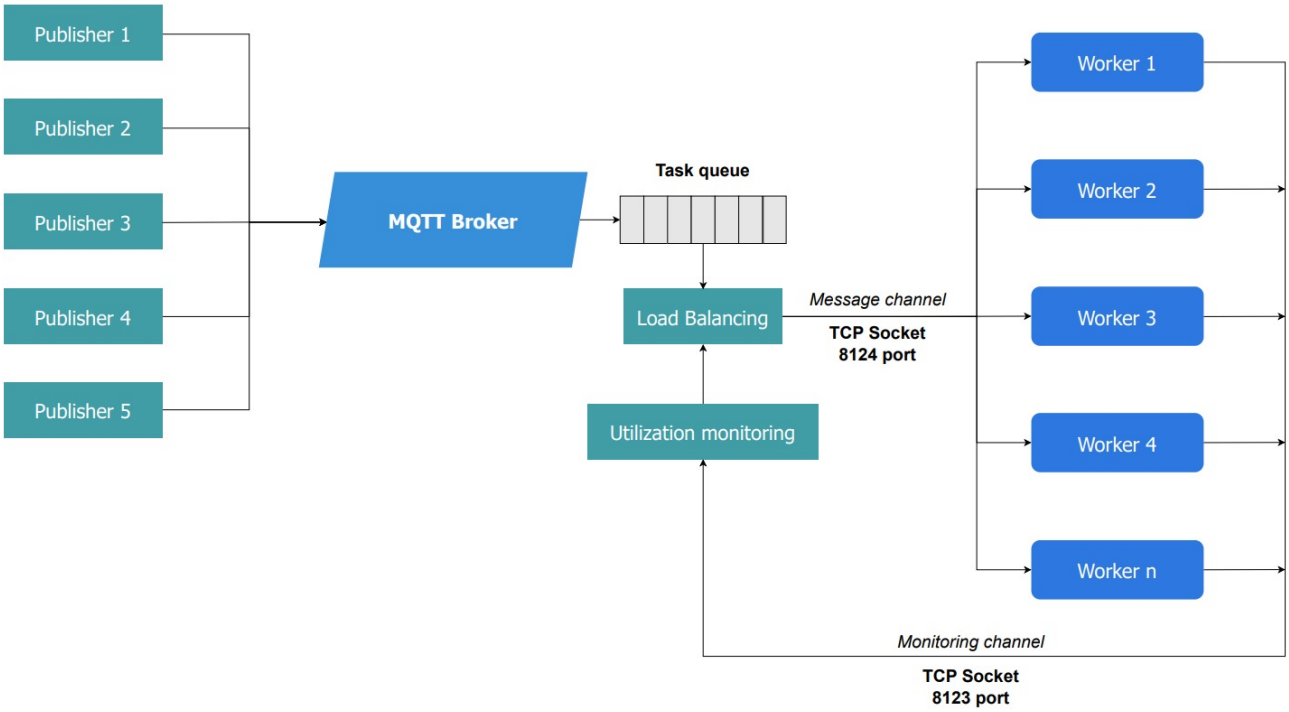


Рис 3.8. Покращена модель MQTT протоколу

Модуль моніторингу. Модуль моніторингу визначає та аналізує миттєві значення завантаженості по кожному показнику. Вищенаведений модуль складається з трьох основних компонентів:

- **Сервіс моніторингу сервера** - визначає поточне значення завантаженості сервера, та передає по моніторинговому каналу передачі. В запропонованій реалізації, є частиною клієнтської бібліотеки MQTT.
- **Канал передачі** - TCP сокет, по якому передаються моніторингові дані. В даному випадку передається серіалізований JSON (табл. 3.1).

Таблиця 3.1. Структура об’єкту з моніторинговими даними

Поле	**Тип**
utilization	Number
cpu_utilization	Number
ram_utilization	Number
disk_utilization	Number
network_utilization	Number

- **Сервіс обробки та аналізу моніторингової інформації** - компонент MQTT брокера, який отримує, зберігає та аналізує значення завантаженості по кожному підключеному серверу.

Визначення коефіцієнта завантаженості. Для балансування навантаження, введемо загальний показник (3.7) завантаженості сервера - L$oadScore$, який визначаємо на основі отриманих моніторингових даних.

$$\text{L}oadScore(w) = \int_0^T [w_{CPU} * CPU(w,t) + w_{RAM} * RAM(w,t) + w_{DISK} * DISK(w,t) + w_{NET} * NET(w,t)] , \quad (3.7)$$

де: w_{CPU}, w_{RAM}, w_{DISK}, w_{NET} – вагові коефіцієнти використання ресурсів процесора, оперативної пам’яті, диска, мережі відповідно, а: $CPU(w,t)$, $RAM(w,t)$, $DISK(w,t)$, $NET(w,t)$- функції використання ресурсів сервера w в момент часу t.

В даному випадку, вводяться додаткові коефіцієнти для кожної характеристики, що дозволяє динамічно змінювати вплив того, чи іншого параметра на розподіл навантаження в системі. Використання динамічних коефіцієнтів, дозволяє уникнути перевантаження конкретних ресурсів.

Наприклад, якщо деякий сервер має великий об’єм оперативної пам'яті (RAM) і в той же час є обмеженим по CPU ресурсам, то можливо зменшити коефіцієнт : w_{RAM} та збільшити : w_{CPU}, що дозволить підвищити загальну продуктивність сервера.

Алгоритм балансування навантаження. В процесі балансування навантаження для активних серверів вводиться додаткова змінна - порогове значення (англ. threshold). Порогове значення встановлюється на основі прийнятного навантаження, яке сервер може впоратися без надмірного навантаження.

Коли LoadScore сервера перевищує це порогове значення, це вказує на те, що використання ресурсів сервера (завантаження ЦП, використання оперативної пам’яті, використання диска та використання мережі) досягло точки, коли воно вважається занадто високим для оптимальної продуктивності (рис. 3.9).

Algorithm 1 Load Balancing Algorithm

Require: Set of workers: $W = \{w_1, w_2, ..., w_n\}$
Require: Metrics for each worker w:
$CPU(w)$ {CPU utilization}
$RAM(w)$ {RAM usage}
$Disk(w)$ {Disk utilization}
Require: Load weights: CPU_{weight}, RAM_{weight}, $Disk_{weight}$
for each worker w in W **do**
    CollectData(w, Δt) {Collect CPU, RAM, Disk data}
    Calculate $LoadScore(w)$ {Based on load weights}
end for
Overload threshold: $Threshold$
Set of available servers below threshold:
$AvailableServers = \{w \in W | LoadScore(w) < Threshold\}$
Ranked workers list: $R = [w_1, w_2, ..., w_n]$ sorted by $LoadScore(w)$ in ascending order
Received new message with load preferences: $Message = (CPU_{pref}, RAM_{pref}, Disk_{pref})$
for each worker s in ranked list R **do**
    if $(CPU(w) \leq CPU_{pref})$ and $(RAM(w) \leq RAM_{pref})$ and $(Disk(w) \leq Disk_{pref})$ **then**
        Send $Message$ to worker w and break
    end if
end for
Continuously update load scores based on real-time data

Рис 3.9. Алгоритм балансування навантаження в покращеній моделі MQTT протоколу

Сервери з LoadScores нижче порогового значення вважаються такими, що мають кероване навантаження та можуть отримувати нові повідомлення (завдання). Натомість, сервери з LoadScores вище порогового значення не повинні отримувати повідомлення, щоб запобігти подальшому навантаженню своїх ресурсів.

Визначення коефіцієнта рівномірності розподілу навантаження. Для оцінки ефективності балансування введемо спеціальну характеристику - коефіцієнт рівномірності розподілу навантаження. Для цього застосуємо комбінацію статистичних метрик для кожного параметра, який впливає на

завантаженість сервера, а також агреговану метрику для всієї системи. Один з можливих підходів - використання "відстаней" між серверами на основі значень параметрів завантаженості. Для розрахунку "відстаней" - застосуємо евклідової різниці між векторами параметрів. В даному випадку, будуть розглядатись вектори миттєвих значень.

Позначимо вектори параметрів (3.8) миттєвого завантаженості сервера як, x_1, x_2, x_i, де x_i - вектор вимірів для і-го сервера:

$$x_i = (CPU_i, RAM_i, DISK_i, NET_i), \tag{3.8}$$

Тоді евклідова відстань між двома векторами x_i та x_j може бути обчислена за формулою:

$$Distance(x_i, x_j) = \sqrt{\sum_{k=1}^{4} (x_{ik} - x_{jk})^2}, \tag{3.9}$$

де: x_{ik} – k-та координата вектору x_i, x_{jk} – k-та координата вектору x_j.

Таким чином, можемо обчислити евклідову відстань між кожною парою серверів. На наступному етапі можна використовувати ці відстані для обчислення агрегованої метрики середньої відстані:

$$Average\ Distance = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} Distance(x_i, x_j), \tag{3.10}$$

Вищенаведена метрика і буде показувати рівномірність розподілу навантаження між серверами. Якщо значення цієї метрики велике, це може вказувати на нерівномірний розподіл навантаження, а якщо маленьке - на рівномірний розподіл.

Проведення експерименту для визначення ефективності запропонованого методу. Метою даного експерименту є порівняння ефективності існуючого та запропонованого методів балансування навантаження в системах на основі MQTT протоколу. Для проведення експерименту

використовувалося два MQTT брокери з різними методами балансування. А саме HiveMQ з методом Shared Subscriptions (далі - метод 1) а також брокер розроблений, на основі запропонованого методу (далі - метод 2).

Сценарій експерименту. Для емуляції нерівномірного навантаження, яке генерується в транспортній системі, було згенеровано 50 завдань з різними типами та інтенсивністю. В Таблиці 3.2 наведено приклад експериментальних завдань. На основі цих даних, MQTT клієнти формують повідомлення та відправляють на MQTT брокер, який в свою чергу маршрутизує його до відповідного сервера (контейнера) за певним алгоритмом.

Таблиця 3.2. Приклад завдань експерименту

Ідентифікатор повідомлення	**Навантаження**			
	CPU	RAM	Disk I/O	Network
1	50	46	21	32
2	12	78	12	24
3	1	32	55	21
4	16	76	6	75
5	9	11	22	87
6	44	7	37	14
7	1	9	61	53
8	12	43	29	32
9	43	30	18	98
10	4	12	14	16

Після того, як контейнер отримує повідомлення, виконується програмний код, який генерує навантаження пропорційне до отриманих коефіцієнтів: CPU/RAM/Disk/Network. В даному експерименті, було використано 5 контейнерів, характеристики яких наведені в Таблиці 3.3.

Таблиця 3.3. Характеристики обчислювальних контейнерів

Параметр	Значення
Назва сервера	Basic
Тип процесора	Regular Intel
Кількість ядер процесора	1
Об’єм оперативної пам’яті	512Mb
Тип дискового накопичувача	SSD
Об’єм дискового простору	8Gb
Пропускна здатність мережі	1Gb/s
Операційна система	Debian

Основною характеристикою при порівнянні методів балансування є коефіцієнт рівномірності розподілу навантаження.

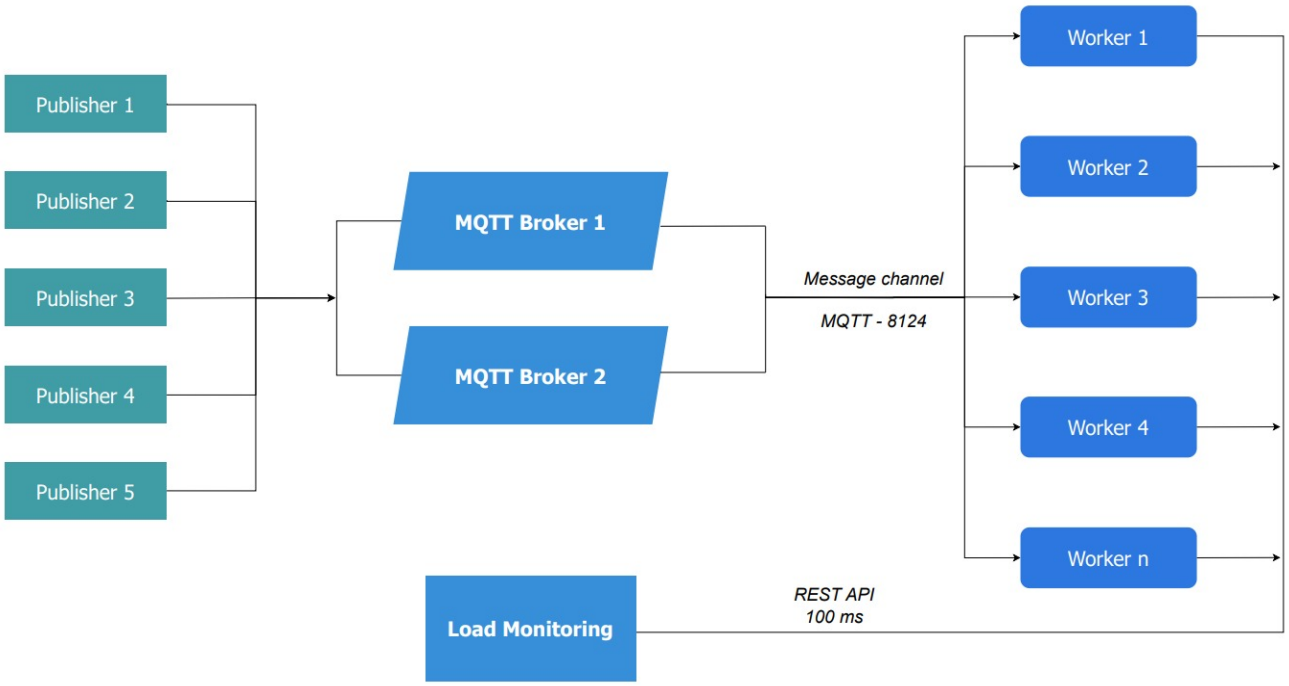


Рис 3.10. Загальна схема експерименту

Для обчислення даного коефіцієнта необхідно зберігати миттєві значення навантаження кожного контейнера з певним інтервалом. У випадку з

запропонованою архітектурою, це можна зробити на рівні брокера, оскільки він отримує показники навантаження від контейнерів в режимі реального часу. Але брокер HiveMQ не отримує цієї інформації, тому для проведення експерименту необхідно винести моніторинг в окремий модуль - Load Monitoring. В даному експерименті, сервер моніторингу отримує показники навантаження від контейнерів з інтервалом 100 мс, та обчислює коефіцієнт рівномірності розподілу навантаження для кожного набору отриманих значень. На Рисунку 3.10 наведено загальну схему експерименту.

Результати експерименту. У цьому розділі продемонстровані основні результати експерименту. Для оцінки ефективності досліджуваних методів, було проведено моніторинг та порівняння основних характеристик завантаженості сервера. Діаграми (рис. 3.11, 3.12) дозволяють візуально оцінити відмінність розподілу завдань між серверами двома досліджуваними методами (метод 1 та метод 2) у різні моменти часу. На основі цих даних можна виявити ключові піки навантаження або перевантаження серверів

Таблиця 3.4. Розподіл завдань між обчислювальними контейнерами. Метод 1.

Task ID	**Container ID**	**Execution Time**		
		Start	End	Total
1	3	1691166950859	1691167000860	50001
2	1	1691166951847	1691166963848	12001
3	5	1691166952849	1691166953850	1001
4	2	1691166969850	1691166969850	16000
5	4	1691166954851	1691166963852	9001
6	3	1691166955853	1691166999853	44000
7	1	1691166956856	1691166957856	1000
8	5	1691166957857	1691166969857	12000
9	2	1691166958855	1691167001856	43001
10	4	1691166959856	1691166963857	4001

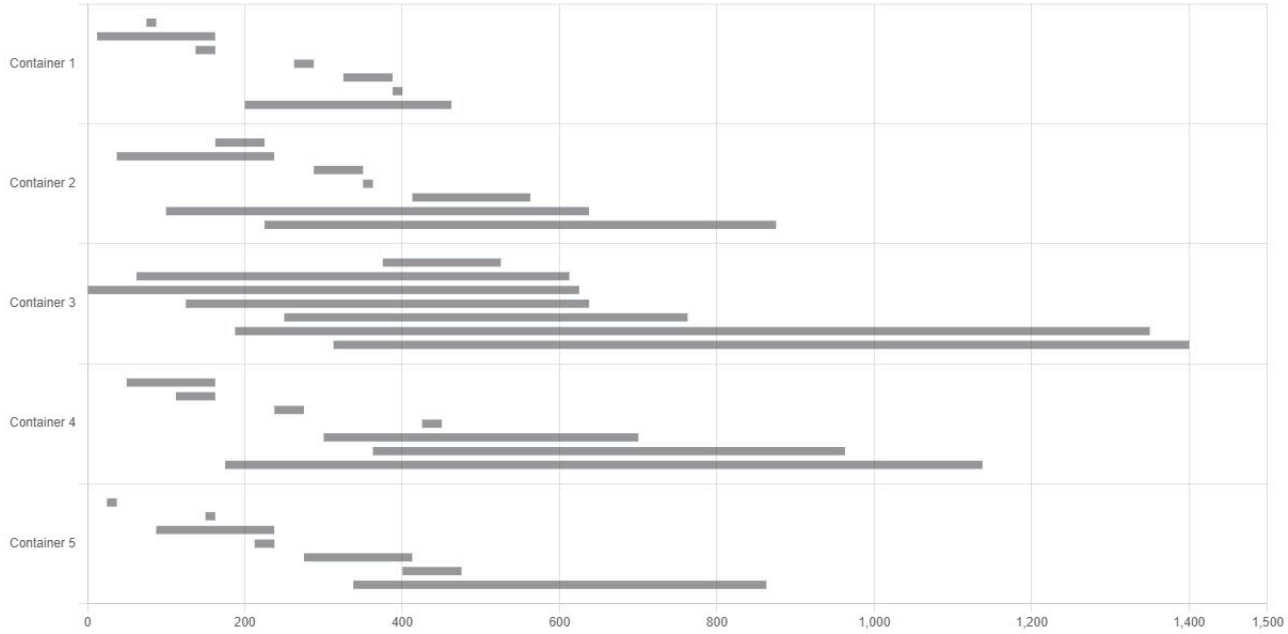


Рис 3.11. Діаграма виконання завдань. Метод 1

Таблиця 3.5. Розподіл завдань між обчислювальними контейнерами. Метод 2.

Task ID	**Container ID**	**Execution Time**		
		Start	End	Total
1	5	1691167138683	1691167188684	50001
2	4	1691167139682	1691167151683	12001
3	2	1691167140684	1691167141684	1001
4	3	1691167141687	1691167157688	16000
5	2	1691167142686	1691167151686	9001
6	1	1691167143691	1691167187692	44000
7	5	1691167144688	1691167145689	1000
8	4	1691167145695	1691167157695	12000
9	5	1691167146693	1691167189693	43001
10	3	1691167147694	1691167151695	4001

На Рисунку 3.14 та 3.19 наведено динаміку зміни основних характеристик завантаженості сервера для кожного методу.

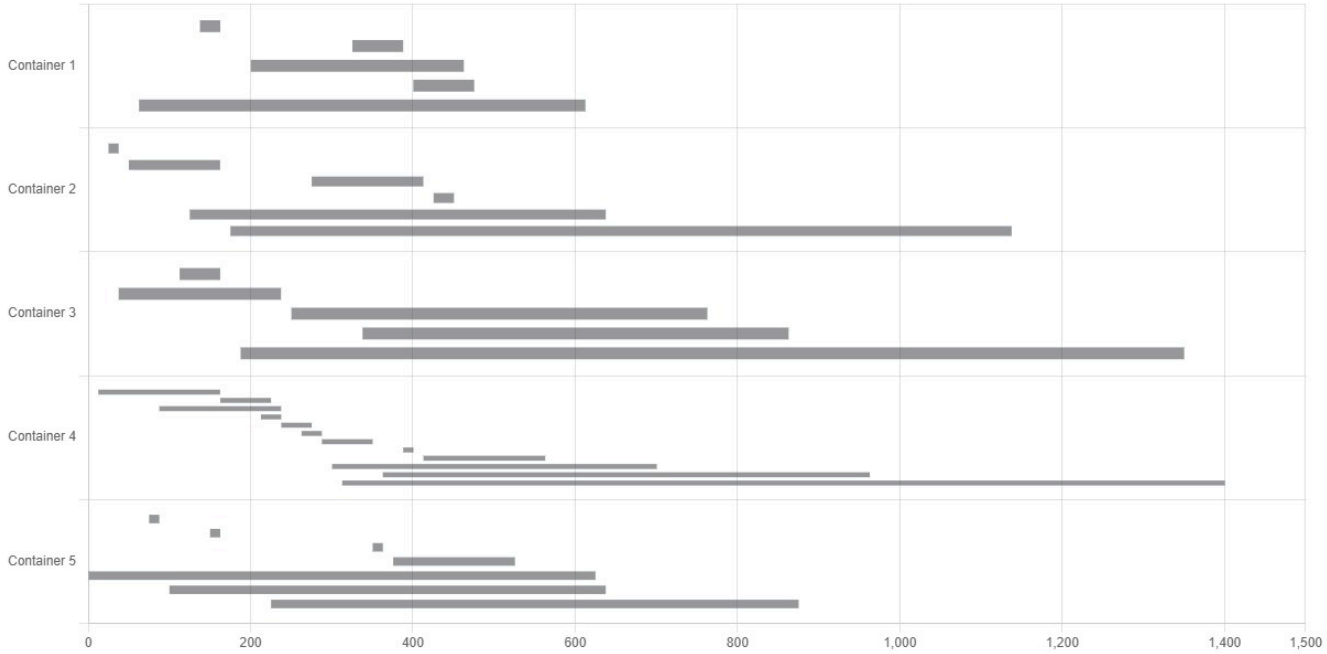
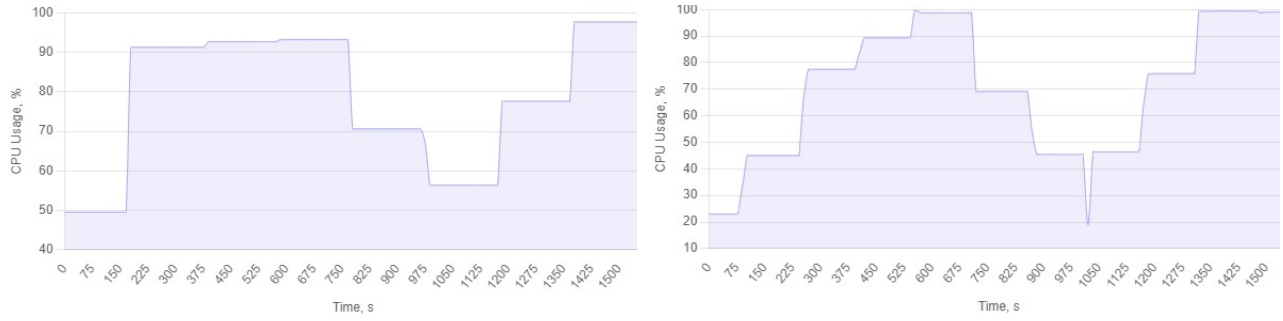


Рис 3.12. Діаграма виконання завдань. Метод 2

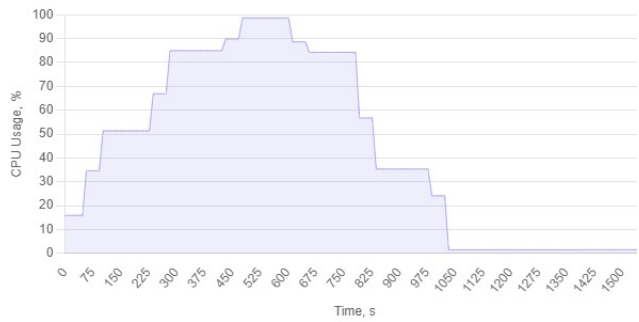
Детальний аналіз цих характеристик є важливою частиною процесу оцінки ефективності та надійності досліджуваних методів балансування.

Оцінка проводилась по наступних характеристиках: CPU Usage, Disk I/O та Network Utilization.



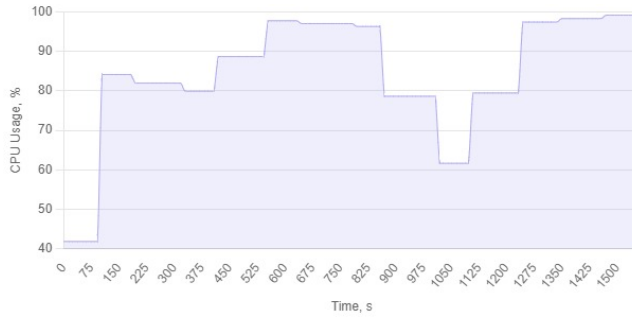
а)

б)



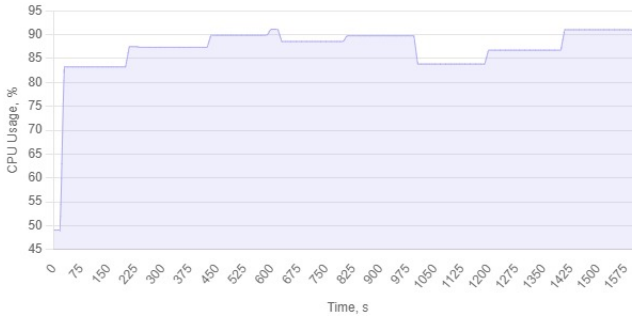
в)

г)

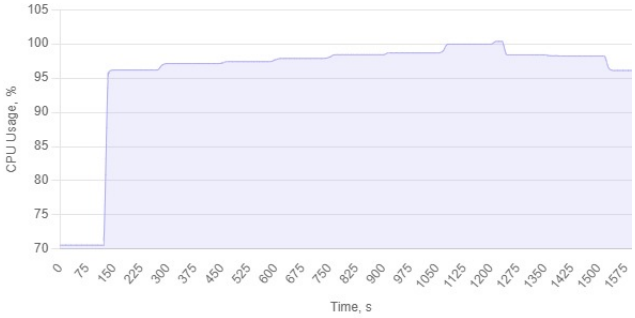


д)

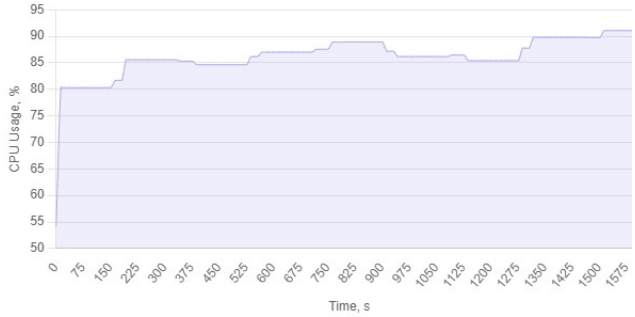
Рис. 3.14. Динаміка зміни завантаженості процесора. Метод 1: а-д) контейнер 1-5



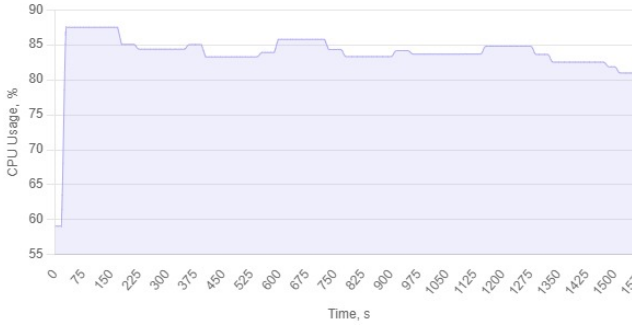
а)



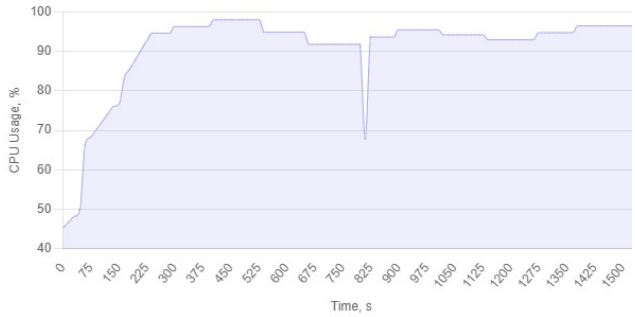
б)



в)

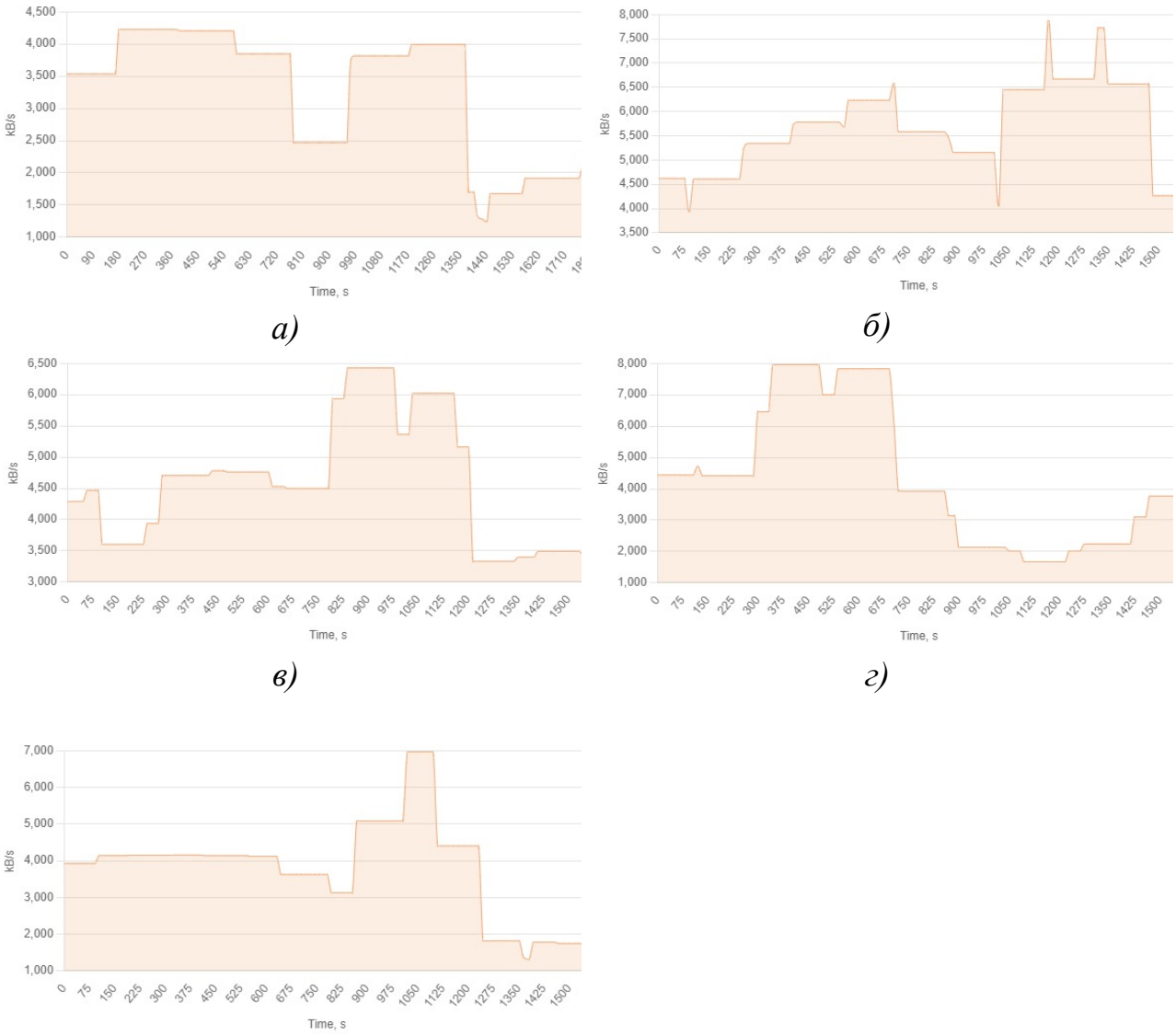


г)



д)

Рис. 3.15. Динаміка зміни завантаженості процесора. Метод 2: а-д) контейнер 1-5

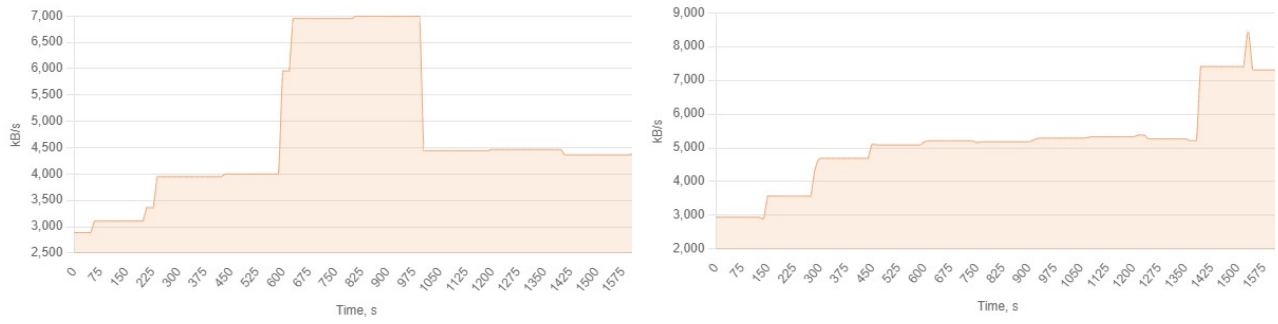


а) *б)*

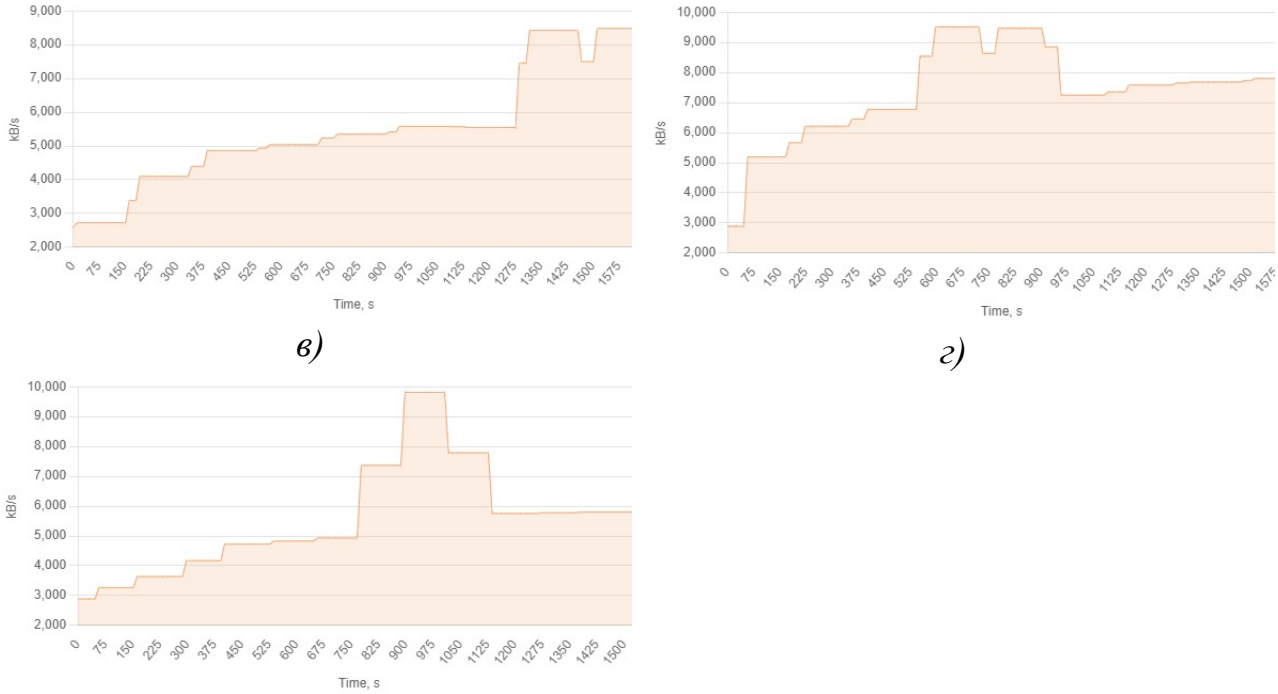
в) *г)*

д)

Рис. 3.16. Динаміка зміни використання серверних ресурсів. Метод 1: а-д) контейнер 1-5



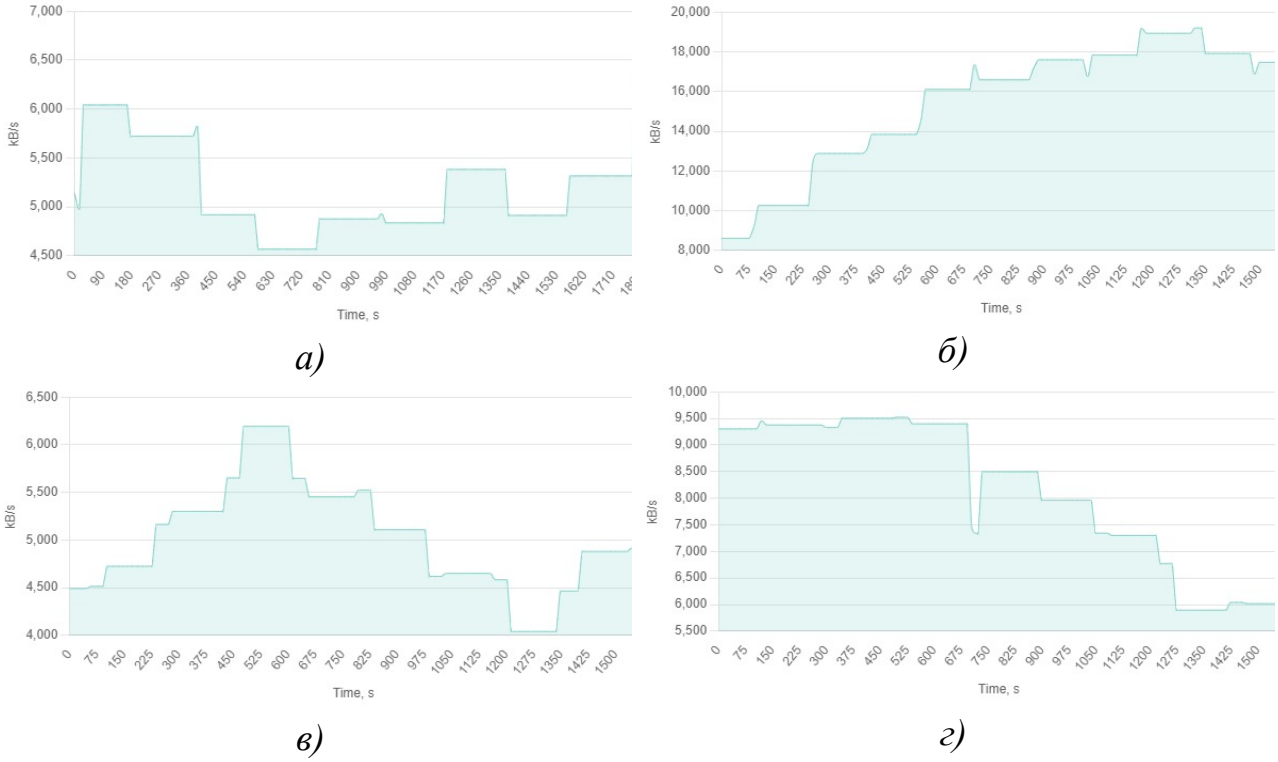
а) *б)*



в) *г)*

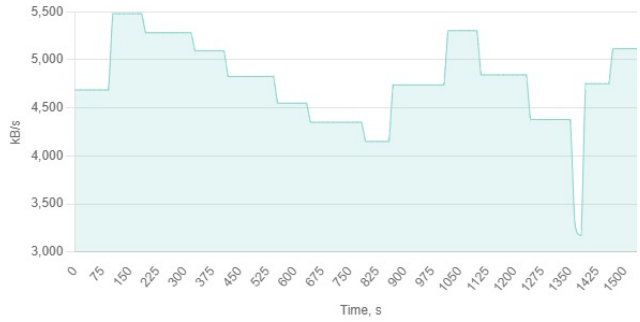
д)

Рис. 3.17. Динаміка зміни використання серверних ресурсів. Метод 2: а-д) контейнер 1-5



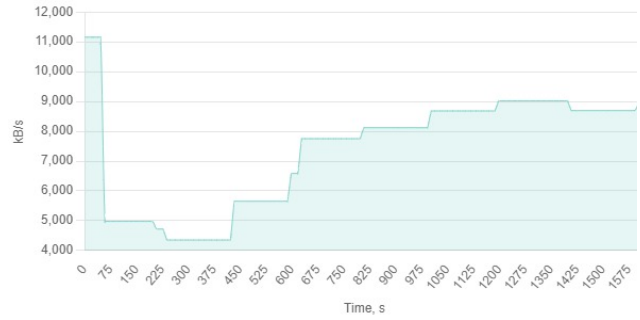
а) *б)*

в) *г)*

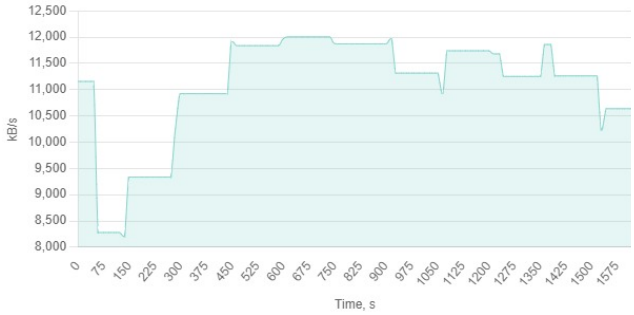


д)

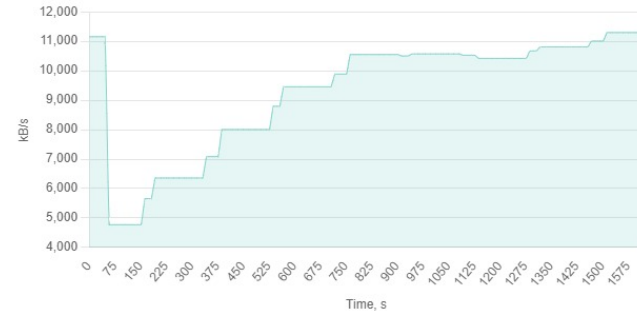
Рис. 3.18. Динаміка зміни використання дискових ресурсів. Метод 1: а-д) контейнер 1-5



а)



б)



в)

г)

д)

Рис. 3.19. Динаміка зміни використання дискових ресурсів. Метод 2: а-д) контейнер 1-5

В даному випадку, ідеальний стан відображає рівномірне розподілення завантаження та ресурсів між всіма серверами, що дозволяє уникнути перевантаження та підвищити швидкість виконання обчислень. Крім того, оцінка рівномірності використання ресурсів в межах одного сервера є також важливою.

Оскільки, дана характеристика допомагає виявити можливий нерівномірний розподіл завдань на рівні окремого сервера та підвищити його продуктивність, оптимізуючи використання CPU, пам'яті, дискового простору та мережі.

Коефіцієнт рівномірності розподілу навантаження. Визначення коефіцієнта рівномірності розподілу навантаження проводилось згідно (3.10) для миттєвих значень завантаженості серверів з інтервалом 100 мс.

Розглянемо приклад розрахунку коефіцієнта для 5-ти серверів с наступними значеннями навантаження:

$$Server\ 1\colon x_1 = \{0.8,0.7,0.6,0.5\}$$
$$Server\ 2\colon x_2 = \{0.9,0.65,0.55,0.45\}$$
$$Server\ 3\colon x_3 = \{0.75,0.6,0.7,0.55\}$$
$$Server\ 4\colon x_4 = \{0.85,0.75,0.65,0.6\}$$
$$Server\ 5\colon x_5 = \{0.88,0.68,0.58,0.48\}, \quad (3.11)$$

Обчислимо евклідову відстань між кожною парою серверів за допомогою формули (3). Для прикладу, обчислимо відстані між серверами 1 і 2:

$$Distance(x_1, x_2) = \sqrt{(0.8-0.9)^2 + (0.7-0.65)^2 + (0.6-0.55)^2 + (0.5-0.45)^2} = 0.169, \quad (3.12)$$

Аналогічно, обчислюються відстані між усіма парами серверів. Після цього можна обчислити середню відстань як агреговану метрику:

$$Average\ Distance = \frac{1}{5*4}\big(Distance(x_1,x_2) + Distance(x_1,x_3) + Distance(x_1,x_4) + Distance(x_1,x_5) + Distance(x_2,x_3) + Distance(x_2,x_4) + Distance(x_2,x_5) + Distance(x_3,x_4) + Distance(x_3,x_5) + Distance(x_4,x_5)\big) = 0.173, \quad (3.13)$$

На графіку (рис. 3.20) продемонстровані значення обчисленого коефіцієнту розподілу навантаження в системі для двох методів балансування. В даному випадку, чим значення обчисленої відстані менше, тим рівномірними є розподіл, і відповідно такий метод є ефективнішим.

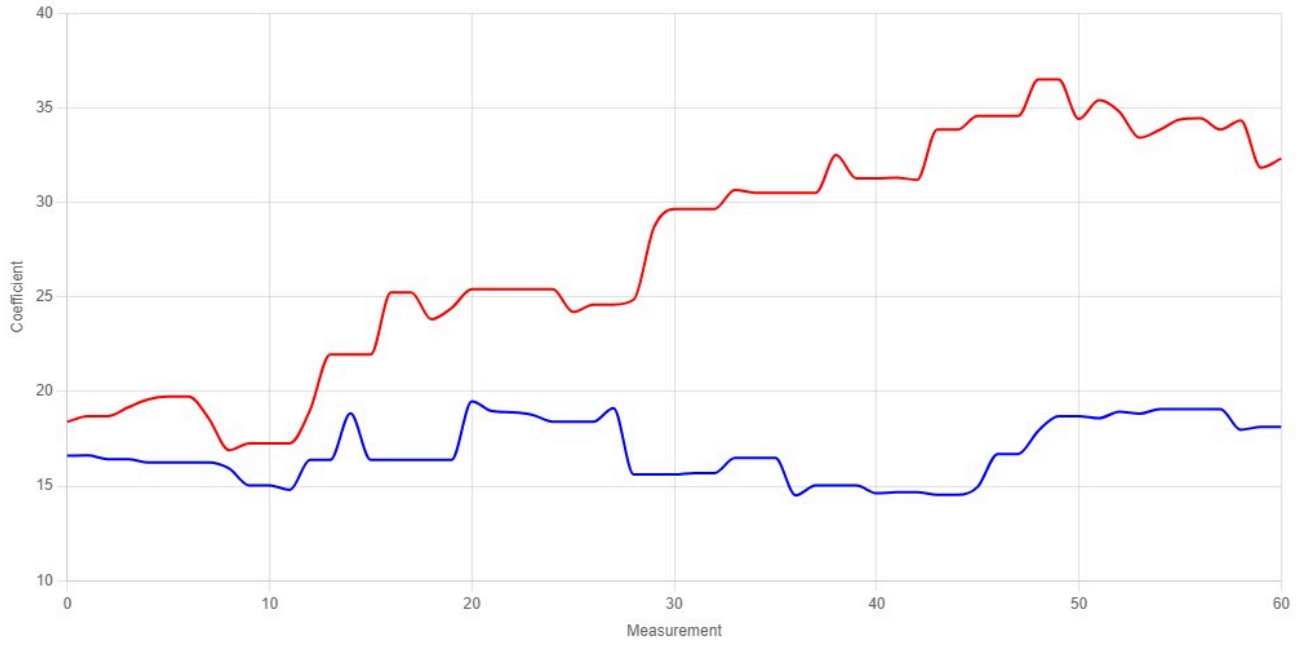


Рис. 3.20. Коефіцієнт рівномірності розподілу навантаження. Червоний графік – метод 1; синій – метод 2

Вищенаведений коефіцієнт є основним показником для аналізу та оцінки ефективності роботи методів балансування навантаження, оскільки він відображає здатність системи до адаптації та оптимізації розподілу завдань та ресурсів. Даний показник дозволяє оцінити, наскільки добре система впоралася зі змінною навантаження, та наскільки ефективно використовуються доступні обчислювальні ресурси.

Обговорення результатів. Отримані результати демонструють переваги запропонованого методу балансування навантаження порівняно з існуючим методом - MQTT Shared Subscriptions. Проведений експеримент з симуляцією реальної системи, яка генерує динамічне навантаження, показав основні відмінності в роботі цих двох методів.

З аналізу часових діаграм (рис. 3.11) видно, що використання стандартного алгоритму балансування Round Robin (використовується в Методі 1) призводить до перевантаження деяких серверів та неефективного використання інших. Особливо показовою є ситуація, коли на сервері Container 3 виконувалось одночасно 7 завдань, тоді як на сервері Container 1 - лише одне. Ця нерівномірність у навантаженні виникає через те, що вищенаведений алгоритм не враховує поточний стан завантаженості серверів та ресурси, необхідні для обробки конкретних завдань, які можуть відрізнятися за складністю чи об'ємом даних.

Порівнюючи це з графіком (рис. 3.12), де представлений розподіл на основі запропонованого алгоритму, можна помітити, забезпечується більш рівномірне балансування завдань між активними серверами. Максимальна різниця в кількості активних завдань протягом усього періоду експерименту не перевищує одиниці. Це свідчить про те, що запропонований метод демонструє більш стійкий та адаптивний підхід до балансування навантаження, забезпечуючи оптимальне використання обчислювальних ресурсів.

Аналіз графіків (рис. 3.14 - рис. 3.19) підтверджує вплив запропонованих методів у забезпеченні ефективного використання основних ресурсів системи і демонструє значну перевагу запропонованого методу. Даний висновок підтверджується графіком (рис. 3.20), де показана зміна коефіцієнта рівномірності розподілу для розглянутих методів. Для запропонованого методу характерно, що значення цього коефіцієнта в середньому вище на 50% у порівнянні з Методом 1, і суттєво не змінюється протягом періоду експерименту. Цей коефіцієнт відображає загальну ефективність та прогнозовану

продуктивність системи, а його підвищення сприяє зниженню затрат на ресурси, оскільки система працює більш ефективно та стабільно.

До недоліків запропонованого підходу можна віднести ускладнену архітектуру MQTT брокера, а також необхідність забезпечення додаткового каналу та модуля моніторингу підключених серверів. Можливим варіантом оптимізації, є об'єднання каналів передачі даних, та каналу моніторингу. Також в даній реалізації, для передачі даних моніторингу використовуються серіалізовані JSON пакети. Більш оптимальним рішенням, може бути використання бінарних форматів передачі даних, наприклад - Protocol Buffers.

3.3. Застосування методів нейронних мереж для обробки даних телекомунікаційної мережі

В системах Інтернету речей, зокрема в досліджуваній системі моніторингу громадського транспорту генеруються великі об`єми структурованих даних. Основним джерелом інформації є підключені IoT пристрої, які передають інформацію з сенсорів та датчиків в режимі реального часу. IoT контролер, через мережу передає на сервер пакети даних з геолокацією, рівнем пального, зарядом акумулятора, кількістю пасажирів, та показниками технічного стану ТЗ. Крім того, дана інформація доповнюється інформацією про погоду, технічний стан доріг, дорожні інциденти та ДТП з відкритих джерел. Після процесів нормалізації та збереження даних, стає можливим їх подальший аналіз та побудова прогнозів. Для вирішення даного типу завдань широко застосовуються методи на основі апарату нейронних мереж. Перевагою застосування методів нейронних мереж, є можливість виявлення прихованих аномалій та трендів, а також визначення кореляцій між параметрами системи та подіями, що в ній виникають. У даному дослідженні розглядаються дві основні задачі: прогнозування пасажиропотоків та визначення ймовірності виникнення ДТП. Розв'язання цих завдань розподілене на три основних етапи: збір та структурування навчальних даних, навчання нейромережі (НМ), розміщення моделі на сервері та її використання в модулі аналізу даних. Процес тренування

НМ включає в себе пошук оптимальної архітектури та конфігурації мережі і вимагає проведення практичних експериментів. Запропонований процес вибору оптимальної архітектури та конфігурації нейронної мережі наведено на Рисунку 3.21

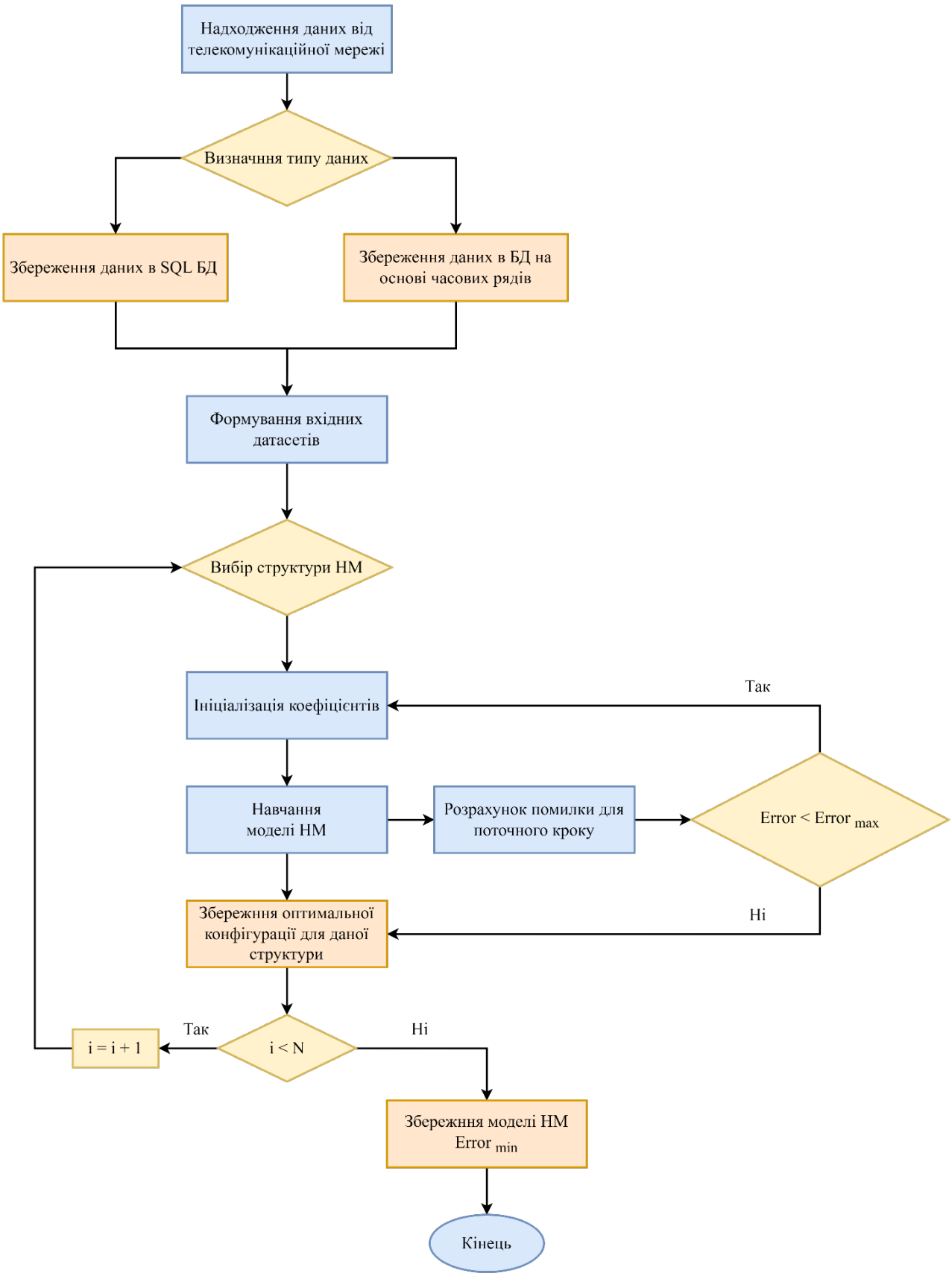


Рис. 3.21. Процес вибору оптимальної моделі нейронної мережі

3.3.1. Вирішення завдання прогнозування пасажиропотоку

Точне прогнозування пасажиропотоку міського транспорту має велике значення для планування транспортних ресурсів, громадської безпеки та оцінки ризиків транспортної системи. Традиційні статистичні підходи до прогнозування часових рядів є не ефективними на практиці. Вони часто вимагають або суворої, або слабкої стаціонарності даних, яку майже неможливо отримати в реальних системах.

Альтернативним методом є прогнозування часових рядів за допомогою нейронних мереж. За своєю природою нейронні мережі є нелінійними і навчаються на основі вхідних і вихідних даних. При такому підході підвищення ефективності мережі зводиться до збільшення обсягу даних вхідної вибірки. Сьогодні для прогнозування часових рядів в основному використовується клас рекурентних нейронних мереж.

В цьому розділі розглядається побудова нейронних мереж для прогнозування пасажиропотоку наземного міського транспорту на основі LSTM та GRU архітектур, а також порівнюється їх ефективність.

Модель призначена для прогнозування кількості пасажирів на основі історичних даних.

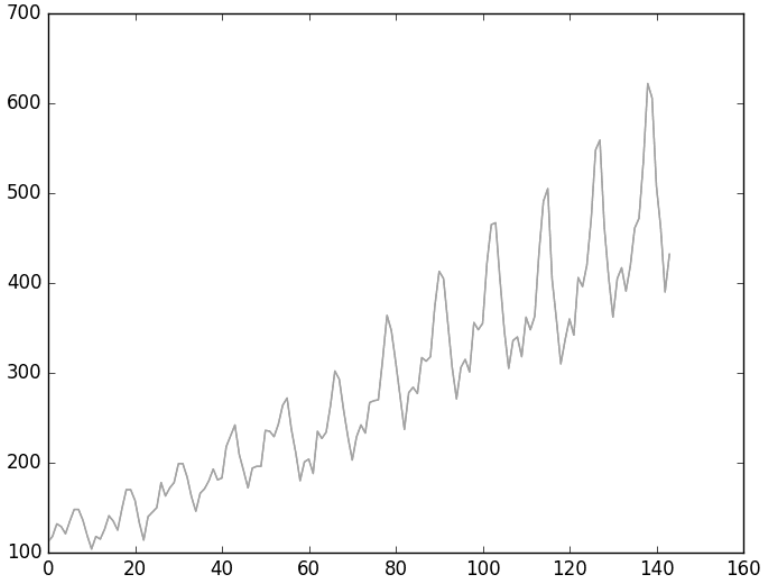


Рис. 3.22. Візуалізація навчального набору даних

Для навчання використовувався датасет, який містить кількість перевезених пасажирів за місяць, і складається з 144 спостережень (рис. 3.22). Існує певна періодичність, яка пояснюється сезонністю.

На першому етапі, необхідно розділити вхідні дані на навчальну та контрольні вибірки. Вхідні дані будуть розділені наступним чином – 67% навчальна вибірка, 33% контрольна вибірка. Для навчання мережі було створено новий набір даних, де X — кількість пасажирів у певний час (t), а Y — кількість пасажирів у наступний момент (t + 1).

Мережа LSTM очікує, що вхідні дані (X) надаються деякою структурою масиву у формі [зразків, часових кроків, особливостей].

Нейронна мережа складається з: 1 вхідного шару, 2-х прихованих шарів із 64 блоками нейронів LSTM та вихідного шару, який робить передбачення єдиного значення (кількості пасажирів). В якості активаційної функції використовується – ReLu [96] (3.14).

$$f(u) = max(0, u), \tag{3.14}$$

Модель тренується з різною кількістю епох навчання, та використовує метод Адам в якості оптимізаційної моделі. Вищенаведена модель є методом стохастичного градієнтного спуску, який базується на адаптивній оцінці моментів першого та другого порядку.

Якість мережі визначається за допомогою функції втрат (в даному випадку середньо квадратична помилка) (3.15), яка визначає різницю між початковими (прогнозованими значеннями) і раніше відомими.

$$cost = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2, \tag{3.15}$$

Основою навчання нейронної мережі є мінімізація вищенаведенної функції. На Рисунку 3.23 наведено алгоритм навчання мережі.

В Таблиці 3.6 наведено результати навчання нейронної мережі в залежності від архітектури та кількості епох.

Таблиця 3.6. Точність LSTM на GRU моделей

Модель	**Точність**	**Відсоток помилок**
LSTM (10 Epochs)	71.18%	28.82%
GRU (10 Epochs)	67.11%	32.89%
LSTM (30 Epochs)	73.64%	26.36%
GRU (30 Epochs)	61.02%	38.98%
LSTM (50 Epochs)	86.83%	13.17%
GRU (50 Epochs)	67.12%	32.88%
LSTM (100 Epochs)	86.1%	13.9%
GRU (100 Epochs)	67.18%	32.82%
LSTM (1000 Epochs)	71.18%	28.82%
GRU (1000 Epochs)	67.11%	32.89%
LSTM (10k Epochs)	81.39%	18.61%
GRU (10k Epochs)	67.05%	32.95%
LSTM (50k Epochs)	76.82%	23.18%
GRU (50k Epochs)	66.22%	33.78%
LSTM (100k Epochs)	53.68%	46.32%
GRU (100k Epochs)	66.16%	33.84%

Відповідно до наведених вище даних найкращий результат (точність 86,83%) отримано з архітектурою LSTM і 50 епохами навчання. Також можна помітити точку зниження продуктивності моделі (LSTM > 50k епох). Це момент насичення, або момент перенавчання мережі.

На графіках результатів (рис. 3.24 – 3.27) наведено найкращі результати для обох мереж (LSTM – 100 епох, GRU – 10 епох).

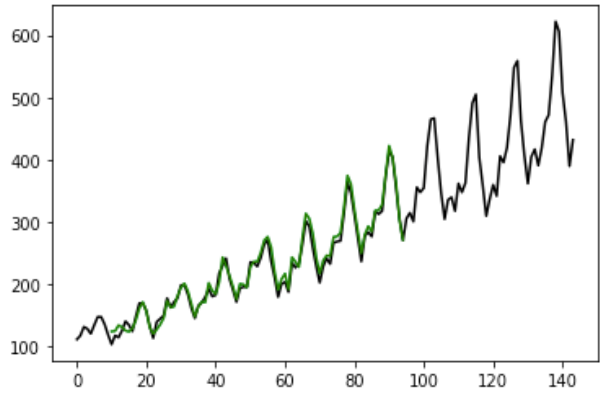


Рис. 3.23. Точність LSTM мережі на тренувальних даних

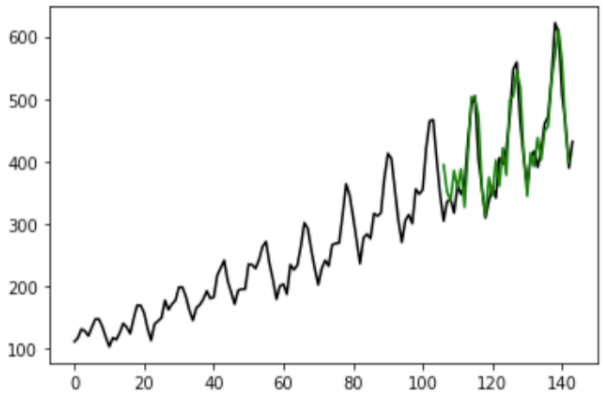


Рис. 3.24. Точність LSTM мережі на контрольних даних

На вищенаведених графіках вісь X – номер місяця, вісь Y – кількість пасажирів. Основний рядок показує вхідні дані, допоміжний – результат модельного прогнозу.

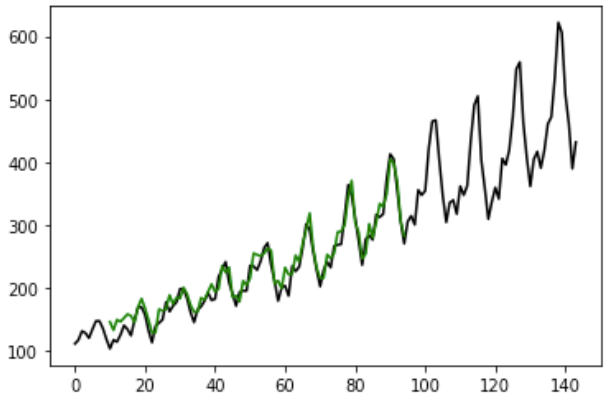


Рис. 3.25. Точність GRU мережі на тренувальних даних

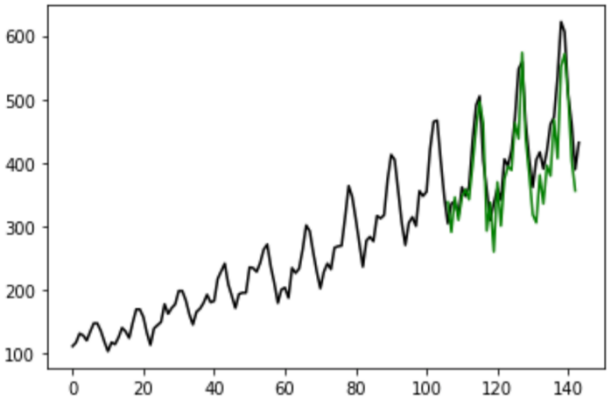


Рис. 3.26. Точність GRU мережі на контрольних даних

На Рисунках 2.23 та 3.25 показано точність мережі з тренувальними даними, а на Рисунках 2.24 та 2.26 – з контрольними.

Отримані результати показують, що рекурентні нейронні мережі можуть бути застосовані для прогнозування даних часових рядів (зокрема прогнозування пасажиропотоку). Завдання підвищення точності мережі, зводиться до розширення навчальної вибірки даних.

3.3.2 Визначення ймовірності виникнення ДТП

Будь-яка дорожньо-транспортна пригода є випадковою подією, яка може статися в будь-якому місці та в будь-який час, тому лінійні методи прогнозування є малоефективними для цієї задачі. Протягом останніх років такі системи будувались на основі статистичних регресійних методів, таких як логістична регресія. В цьому розділі буде розглянуто модель на основі нейронних мереж, яка отримує вхідний набір даних з часопросторовими характеристиками транспортного вузла та визначає ймовірність виникнення ДТП у ньому.

Система виявляє істотні елементи (фактори), що при аварії, або створює зв'язок між аваріями та різними факторами виникнення. Методом проведення дослідження є навчання та тестування двох нейронних мереж з різними архітектурами в однакових апаратних умовах, з використанням того самого набору даних. Для побудови нейронної мережі використовувались дві популярні архітектури, а саме: рекурентна нейронна мережа на основі LSTM елементів та нейронна мережа на основі багатошарового персептрона.

Для обох моделей були досліджені оптимальні конфігурації навчання, щоб отримати максимальну точність прогнозу. Навчання мереж проводилось на основі набору відкритих даних (таб. 3.7) про ДТП у м. Барселона, Іспанія за 2010-2020 роки [97]. Також даний набір був доповнений значеннями з відкритих джерел, такими як: температура повітря, вологість, дорожнє покриття, та ін..

Таблиця 3.7. Структура вхідного набору даних

Поле	Тип	Опис
Severity	Number	Складність ДТП (0-5)
Location	Geo	Географічні координати виникнення ДТП
Temperature (F)	Number	Температура повітря
Humidity (%)	Number	Відносна вологість повітря
Pressure (in)	Number	Атмосферний тиск
Wind Direction	String	Напрям вітру
Wind Speed	Number	Швидкість вітру

Road Surface	String	Тип дорожнього покриття
Day	Number	Номер дня місяця (0-30)
Hour	Number	Година (0-23)
Month	Number	Номер місяця (0-11)
Weekday	Number	Номер дня тижня (0-6)
Year	Number	Рік
Crossing	Boolean	Ознака перехрестя

Завданням моделі є передбачити ймовірність виникнення ДТП для конкретного транспортного вузла в той чи інший час. Вхідний набір мережі містить дані про ДТП у м. Барселона за період із січня 2010 р. по грудень 2021 р.. А саме 737712 спостереження (предиктори), що в свою чергу складається з 29 характеристик, які частково наведенні в Таблиці 3.7. Спостереження були сегментовані та прив'язані до конкретного вузла. Загалом в вхідному наборі даних виділено 5092 унікальних транспортних вузла в яких відбувались ДТП за вищенаведений період.

Також вхідний набір даних був розширений, емпіричними спостереженнями, в ті моменти часу (дані дискретизовані погодинно), коли в транспортному вузлі була нульова аварійність, тобто Severity=0.

Після формування та нормалізації вхідних даних, відбувається створення навчальної та контрольної вибірки. В даному випадку використовується класичний розподіл: 70% навчальна та 30% контрольна. Формування вибірки відбувається на основі розподілу спостережень сегментованих по транспортному вузлу. Це дозволяє оцінити можливість застосування моделі для нового транспортного вузла (дані якого не застосовувались при навчанні моделі).

Після виконання тренування моделі, необхідно оцінити її якість. В даному випадку для оцінки якості мережі використовується метод середньої відносної похибки [98] (3.16).

$$MAPE = \frac{1}{n}\sum_{1}^{n}\frac{|Y_i - \hat{Y}_i|}{max(|Y_i|, \varepsilon)}100\%, \qquad (3.16)$$

де n – кількість ітерацій, Y – вектор відомих значень, $\hat{Y}$ – вектор прогнозованих значень. Процес навчання (рис. 3.27) відбувається до того етапу, коли мережа може передбачити правильне вихідне значення на основі вхідного (тренувальна вибірка), або максимально наблизиться до нього.

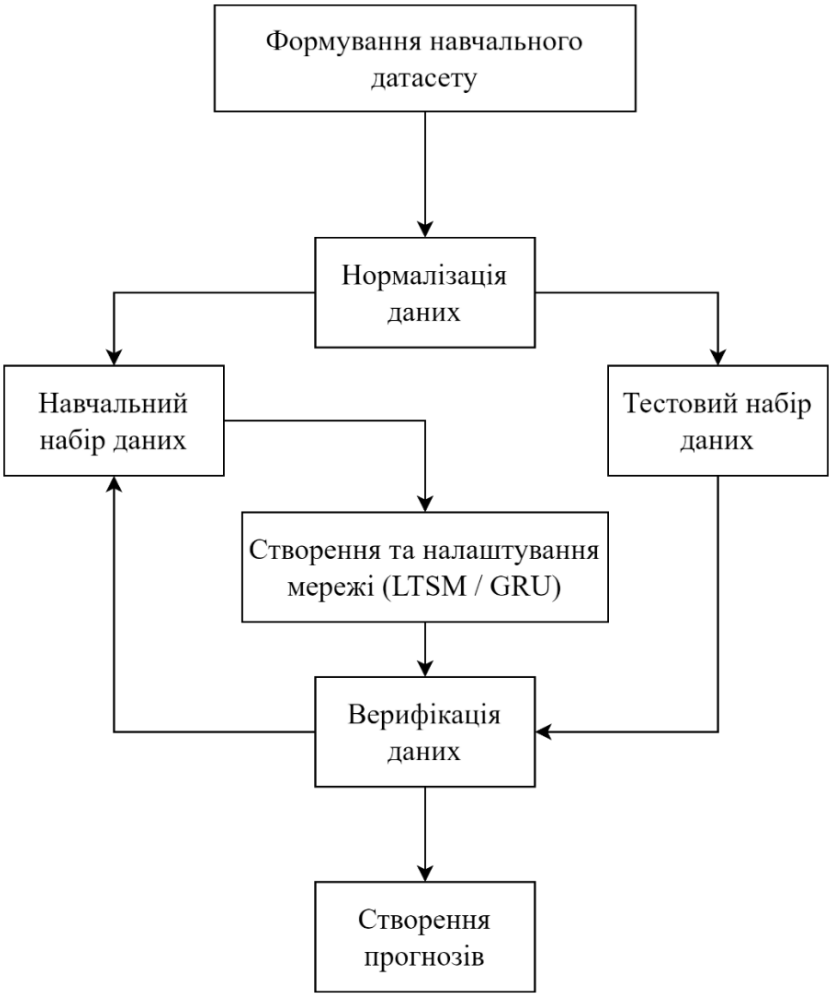


Рис. 3.27. Алгоритм навчання моделі

Таблиця 3.8. Конфігурація мережі на основі багатошарового персептрона

Шар мережі	**Опис**	**К-сть нейронів**	**Активаційна функція**
Вхідний	Входи нейронної	29 (входи)	-
1	Внутрішній шар 1	9	ReLU
2	Внутрішній шар 2	10	ReLU
3	Вихідний шар	1	-
Метод оптимізації		Adam / SGD with Nesterov	
Кількість епох (оптимальна)		100 (27)	
Функція втрат		Mean squared error	
Метод вимірювання точності		Mean absolute percentage error	

В процесі навчання відбувається визначення експериментальним чином найкращих параметрів мережі – кількість внутрішніх шарів, кількість нейронів, кількість епох навчання та методи оптимізації. Оптимальні параметри та архітектура мереж наведені у Таблиці. 3.8 та Таблиці. 3.9.

Таблиця 3.9. Конфігурація мережі на основі LSTM

Шар мережі	**Опис**	**К-сть нейронів**	**Активаційна функція**
Вхідний	Входи нейронної	29 (входи)	-
1	Внутрішній шар 1	16	ReLU
2	Внутрішній шар 2	8	ReLU
3	Вихідний шар	1	-
Метод оптимізації		Adam / SGD with Nesterov	
Кількість епох (оптимальна)		50000 (30000)	
Функція втрат		Mean squared error	
Метод вимірювання точності		Mean absolute percentage error	

Рисунок 3.28 демонструє результати навчання моделі багатошарового персептрона. На тренувальній множині даних вдалось досягти точності прогнозу - 69%. На тестовій множині – 64 - 66%, що є недостатнім для повноцінного практичного використання, однак результати можуть бути покращені за допомогою розширення вхідних характеристик (таб. 3.7), а також кількості

спостережень на основі яких проводиться навчання та тренування мережі. Для отримання оптимальних результатів, моделі достатньо 27 епох навчання, завдяки чому тренування та переконфігурація мережі не вимагає багато часу.

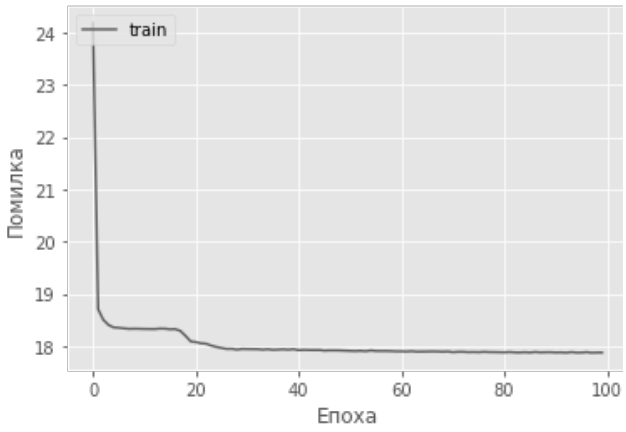


Рис. 3.28. Результати навчання - багатошаровий персептрон

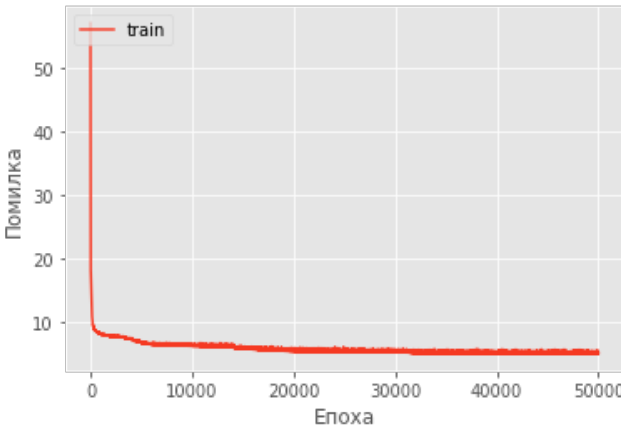


Рис. 3.29. Результати навчання - LSTM

Мережа на основі LSTM нейронів досягла точності 92% на тренувальній множині (рис. 3.29), та 79% на тестовій. Такий результат є цілком прийнятним, а покращення точності мережі зводиться до розширення кількості навчальних даних. До мінусів даної реалізації можна віднести велику кількість ітерацій навчання (30000), що при великому об’ємі вхідних даних, вимагає багато часу та обчислювальних ресурсів.

Отримані результати можуть бути покращені за допомогою розширення вхідних характеристик, та кількості спостережень, на основі яких відбувається навчання. Розроблені моделі нейронних мереж є універсальними, та можуть бути легко застосовані до будь-яких транспортних вузлів, з різними характеристиками.

3.4. Висновки

1. Запропоновано архітектуру системи збереження та обробки даних, сформульовано основні завдання даної системи, а також побудовано її функціональну схему.

2. На основі запропонованої математичної моделі розроблено метод для визначення оптимальної кількості обчислювальних контейнерів в розподілених системах Інтернету речей. Виконано програмну реалізацію даного методу. Даний метод на відміну від існуючих методів враховує динамічні вимоги до обчислювальних систем, які генеруються IoT пристроєм, що дозволяє знаходити оптимальну кількість обчислювальних контейнерів. Застосування даного методу дозволяє підвищити ефективність використання обчислювальних ресурсів на 30-65%, що підтверджують результати експериментів проведених у Розділі 4. Даний метод може бути застосований не лише при побудові Інтелектуальних систем громадського транспорту, а і при побудові систем Інтернету речей в цілому, оскільки є незалежним до типів та структур даних, що передаються в системі.

3. Розроблено математичну модель для оцінки завантаженості обчислювальних контейнерів, а також модель для оцінки рівномірності розподілу навантаження в системі. Запропонована модель, на відміну від існуючих виконує багатопараметричний моніторинг обчислювальних контейнерів (серверів), що дозволяє підвищити точність та ефективність балансування в системах з динамічним навантаженням. На основі запропонованих моделей розроблено алгоритм балансування навантаження. Розроблено покращену архітектуру MQTT брокера, який виконує балансування навантаження, згідно з розробленим алгоритмом. Виконано програмну реалізацію MQTT брокера, а також бібліотеки для підключення клієнтів (підписників). Запропоновано методику тестування, а також виконано програмну реалізацію середовища тестування. Проведено комплекс експериментів для порівняння запропонованих методів та існуючих. Аналіз результатів експериментів підтвердив, що запропоновані методи підвищують ефективність використання основних ресурсів системи і демонструють перевагу запропонованого методу на існуючими. Для запропонованого методу значення коефіцієнта рівномірності розподілу навантаження в середньому на 50% вище ніж існуючого методу. Цей коефіцієнт відображає загальну ефективність та прогнозовану продуктивність системи, а його підвищення сприяє зниженню затрат на ресурси, оскільки система працює

більш ефективно та стабільно. Запропоновані моделі та алгоритми також можуть бути застосовані при побудові систем з динамічним навантаженням та розподіленою архітектурою, де необхідно виконувати рівномірне балансування навантаження.

4. Розроблено алгоритм формування даних телекомунікаційної мережі для навчання нейронних мереж. На основі даного алгоритму виконано програмну реалізацію та отримано моделі для прогнозування пасажиропотоку та визначення ймовірності виникнення ДТП.

РОЗДІЛ 4. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ГРОМАДСЬКОГО ТРАНСПОРТУ НА ОСНОВІ ЗАПРОПОНОВАНОЇ АРХІТЕКТУРИ ТА РОЗРОБЛЕНИХ МЕТОДІВ. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У даному розділі реалізовано Інтелектуальну систему наземного громадського транспорту, яка базується на методах та висновках, отриманих у розділах 2-3 даного дослідження. Зокрема, на підставі аналізу архітектурних підходів, який був проведений у розділі 2.3, було обґрунтовано вибір та прийнято рішення про використання мікросервісної архітектури, як основного архітектурного шаблону при проектуванні системи. Обґрунтованість архітектури реалізованої системи також підкріплена результатами аналізу ефективності технологій передачі даних та протоколів комунікації між основними компонентами системи, що у даному випадку є мікросервісами. А саме, це підтверджують результати експериментів, отримані у розділах 2.2.3, 2.4.1 та 2.4.2. Ці експерименти спрямовані на визначення ефективності та обмежень досліджуваних технологій, та підтвердили їх придатність для розглянутої системи.

На етапі проектування використано математичну модель, запропоновану в розділі 2.5, для імітаційного моделювання поведінки системи Інтернету речей. Результати цього моделювання описані в додатку 8 даної роботи. Крім того, в даній реалізації системи використано методи які сприяють підвищенню робастності та оптимізованому використанню обчислювальних ресурсів системи, які були запропоновані у Розділі 3.

З метою оцінки відповідності розглянутої реалізації вимогам, визначеним у розділі 1, було проведено тестування системи в умовах підвищеного навантаження. Сценарій тестування відтворює динаміку функціонування наземного громадського транспорту міста Києва. З метою об'єктивного порівняння переваг та недоліків з існуючими рішеннями, дане тестування

проводиться і для системи, побудованої на базі вже існуючих комерційних засобів для створення систем Інтернету речей – AWS IoT Core. На основі отриманих результатів також проведено аналіз економічного ефекту в наслідок запровадження запропонованих методів.

4.1. Загальна архітектура системи

Як було зазначено вище, реалізована система складається з набору окремих модулів (мікросервісів). На Рисунку 4.1 наведено блок-схему реалізованої системи. Кожен модуль інкапсульовує в собі певну частину функціоналу системи, а також надає зовнішній інтерфейс комунікації для одержання та передачі даних з іншими модулями.

Розглянемо основні модулі реалізованої системи:

Група автомобільних модулів. Це група пристроїв Інтернету речей, які встановлюються в транспортних засобах. Кожен пристрій містить контролер для збору та обробки даних від підключених сенсорів, а також модем NB-IoT для передачі цих даних через LTE мережу. Це дозволяє системі в режимі реального часу відстежувати основні параметри транспортних засобів, такі як положення, швидкість, витрату пального, рівень заряду та інші. В рамках даного дослідження було реалізовано прототип модуля на базі мікрокомп’ютера Raspberry PI 3 та модему Teltonika TRM250. Також для проведення експериментів було розроблено програмний емулятор даного пристрою. Розроблений емулятор застосовувався при проведені експериментів для генерації динамічного навантаження. Програмна реалізація наведена у додатках.

Канал передачі даних. Для передачі даних використовується технологія NB-IoT. Інформація передається по протоколу MQTT, між пристроями та центральним MQTT шлюзом.

Центральний MQTT шлюз (брокер): Даний сервіс є маршрутизатором між автомобільними модулями та компонентами системи збереження та обробки даних. Він приймає дані від IoT пристроїв та розподіляє їх до відповідних

підсистем. В ході даного дослідження було реалізовано покращений брокер MQTT протоколу, на основі методу запропонованого у розділі 3.3. Варто зазначити, що модуль балансування навантаження є окремим компонентом, але технічно, даний модуль, є частиною MQTT брокера, і реалізовує логіку динамічного навантаження, згідно запропонованого методу. Програмна реалізація брокера наведена у додатках. Також для отримання даних підписниками, на основі вищезазначеного методу було реалізовано клієнтську бібліотеку, яка окрім стандартного функціоналу MQTT клієнта, містить в собі реалізацію модуля моніторингу, який передає дані брокеру по окремому TCP сокету. Програмна реалізація клієнтської бібліотеки наведена у додатках.

Підсистема моніторингу руху. В даному мікросервісі інкапсульована логіка моніторингу маршрутів транспортних засобів, збору даних про їх розташування, швидкість та технічний стан, а також поточну завантаженість та кількість пасажирів. Дана підсистема отримує дані в режимі реального часу від MQTT брокера, на основі алгоритму балансування навантаження. Оскільки система моніторингу отримує дані в режимі реального часу то навантаження на неї є недермінованим. Тому, в даній реалізації, мікросервіс розгортається по схемі кластера, та може динамічно масштабуватись відповідно до навантаження.

Підсистема аналітики даних. Даний компонент реалізовує функціонал для аналізу та обробки даних, отриманих від автомобільних модулів. Окрім лінійних методів, дана система використовує моделі машинного навчання. Зовнішнім інтерфейсом є REST API, через який інші мікросервіси можуть отримати агреговані вибірки історичних даних, а також результати застосування вищезазначених моделей до них.

Підсистема машинного навчання. Даний мікросервіс складається з комплексу систем, та реалізовує методи запропоновані у розділі 3.4. А саме, на основі історичних даних формуються моделі для прогнозування пасажиропотоку та ймовірності виникнення дорожньо-транспортних пригод. На наступному етапі ці моделі використовуються в сервісах моніторингу та аналітики. Алгоритми та

моделі реалізовані мовою програмування Python, на основі бібліотек Keras та TensorFlow. Приклад реалізації моделі наведений у додатках.

Платіжна система: Ця складова відповідає за обробку платежів та фінансових транзакцій в системі. Вона забезпечує безпеку операцій та точність обліку витрат пасажирів. В контексті даної реалізації, даний мікросервіс, виконує збереження платіжної інформацію в базу даних, а також містить кодові абстракції, для подальшої інтеграції з банківськими системами. Оскільки даний мікросервіс є критично важливим, то необхідно забезпечити його високий технічний рівень доступності та масштабованості. В даному випадку, платіжна підсистема, є клієнтом MQTT брокера. Як і система моніторингу, даний мікросервіс, отримує дані від MQTT брокера в режимі реального часу на основі запропонованого алгоритму балансування та розгортається по схемі кластера.

Диспетчерська панель. Даний мікросервіс надає операторам зручний інтерфейс для моніторингу місцезнаходження та технічного стану транспортних засобів. Також в даній системі візуалізується аналітика даних, яка отримується через REST API сервісу аналітики даних. Також, оператори можуть отримати прогнози пасажиропотоку, а також отримати повідомлення про виявлення несправностей та аномалій.

Зовнішні інтеграції. Мікросервіс для надання доступу до системи користувачам через мобільний додаток та веб-інтерфейс. Користувачі мають можливість доступу до розкладів руху, отримання сповіщень та отримання рекомендацій щодо оптимальних маршрутів.

Система авторизації / аутентифікації. Система для управління процесами ідентифікації користувачів та контролю рівнів доступу. Дана система гарантує, що лише авторизовані користувачі мають доступ до відповідних функцій та даних. Мікросервіс використовує, такі методи та стандарти, як система управління ідентифікацією (IAM) та JSON Web Token [99] (JWT) токени, для гарантування безпечності та ефективності управління доступом. Система

управління ідентифікацією виконує автентифікацію та авторизацію користувачів у системі. Це означає, що кожний користувач, який бажає отримати доступ до системи, повинен спочатку пройти ідентифікацію. Це може включати в себе введення логіну та пароля, аутентифікацію з використанням біометричних даних або інші способи перевірки. Після ідентифікації користувача, генерується JSON Web Token (JWT) токен. JWT токен є зашифрованим об'єктом, який містить інформацію про користувача та його права доступу. Він підписаний цифровою підписом, що гарантує його цілісність та відсутність модифікацій. Далі, при кожному запиті до системи, користувач представляє свій JWT токен для перевірки своєї авторизації та доступу до конкретних ресурсів. Система перевіряє підпис токена та розшифровує його, отримуючи інформацію про користувача та його дозволені дії.

Зважаючи на великий об’єм та тип даних, що генеруються в даній системі, застосовано підхід до збереження та обробки інформації, який включає в себе використання трьох баз даних різних моделей (таб. 4.1). А саме: реляційні (SQL), нереляційні (NOSQL) а також бази даних на основі часових рядів.

Таблиця 4.1. Застосовані моделі для збереження даних

Модель	**База даних**	**Тип даних**
SQL	PostgreSQL AWS RDS	Дані користувачів Ролі доступу
NoSQL	MongoDB DynamoDB	Конфігурації пристроїв Налаштування системи Метадані
Time-Series	AWS Timestream [100]	Дані телеметрії, що генерується IoT пристроями

Використання різних типів баз даних у рамках даної системи ускладнює процес обробки та управління. З метою досягнення єдиної точки доступу та спрощення взаємодії з різними типами баз даних, було створено спеціальний

програмний компонент – Адаптер БД. Цей компонент виконує роль проміжної інтерфейсної ланки, що забезпечує високорівневий доступ до даних різних типів та структур.

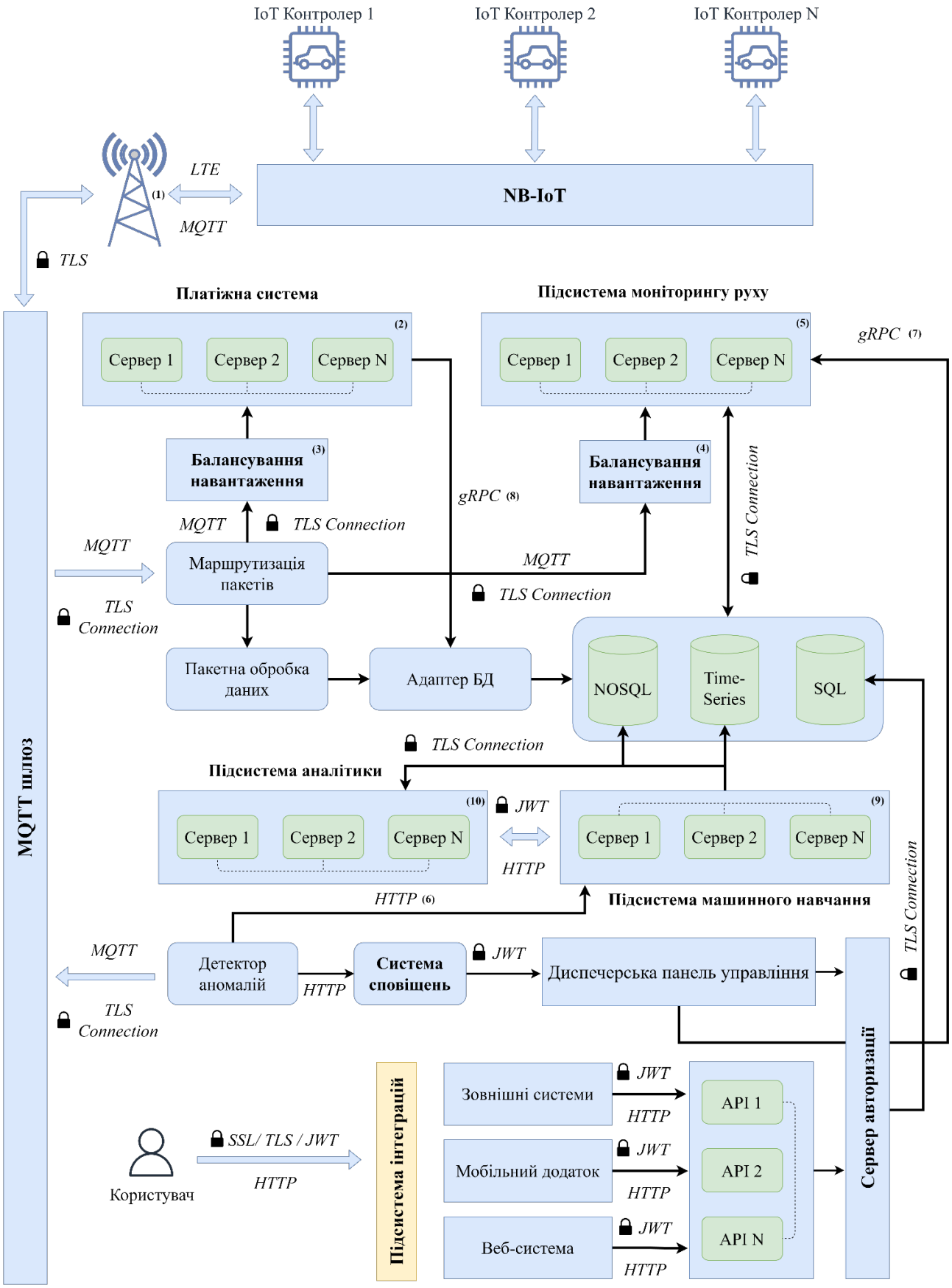


Рис. 4.1. Блок схема реалізованої системи

Таблиця 4.2. Застосування запропонованих методів в розробленій системі

№ вузла	**Застосовані методи / Результати експериментів**	**Опис**
1	*Експериментальні дослідження технологій передачі даних та протоколів програмного рівня. Розділ 2.2.*	На основі проведеного аналізу та результатів експериментів, в даній системі NB-IoT застосовано як основну технологію передачі даних між фізичними пристроями та системою обробки та збереження. Також на основі проведених експериментів, було застосовано протокол MQTT.
2	*Метод визначення оптимальної кількості обчислювальних контейнерів Розділ 3.2.1.*	Мікросервіс платіжної системи повинен забезпечувати високу робасність та відмовостійкість. Тому даний модуль реалізований по схемі кластера, який містить декілька дублюючих екземплярів контейнерів. Для визначення оптимальної кількості, застосовується запропонований у розділі 3 метод.
3	*Покращений метод балансування в розподілених IoT системах. Розділ 3.3.2.*	Забезпечення рівномірного розподілу навантаження між активними контейнерами.
4	*Покращений метод балансування в розподілених IoT системах. Розділ 3.3.2.*	Забезпечення рівномірного розподілу навантаження між активними контейнерами.
5	*Метод визначення оптимальної кількості обчислювальних контейнерів. Розділ 3.2.1.*	Мікросервіс моніторингу отримує дані в режимі реального часу, а його недоступність може привести до втрати даних. Тому для даного мікросервісу також необхідно забезпечити відмовостійкість та високу доступність. Для цього, даний модуль також використовує мульти-контейнерну архітектуру, з динамічним визначенням кількості обчислювальних контейнерів, на основі запропонованого методу.

6	*Експериментальні дослідження протоколів програмного рівня. Розділ 2.4.2.*	Для передачі даних між програмними мікросервісами системи збереження та обробки даних застосовано протоколи, на основі експериментів проведених у розділі 2. Між компонентами, де необхідно передавати дані невеликого об'єму з високою швидкістю, застосовано протокол – gRPC, а там де необхідно забезпечити модель ЗАПИТ-ВІДПОВІДЬ, та для нерегулярної передачі великих об'ємів даних – HTTP.
7	*Експериментальні дослідження протоколів програмного рівня. Розділ 2.4.2.*	
8	*Експериментальні дослідження протоколів програмного рівня. Розділ 2.4.2.*	
9	*Методи нейронних мереж для обробки даних. Розділ 3.4*	Підсистема машинного навчання отримує історичні дані телекомунікаційної мережі з бази даних на основі часових рядів, на основі яких будується вхідний датасет для тренування моделей машинного навчання.
10	*Метод визначення оптимальної кількості обчислювальних контейнерів Розділ 3.2.1.*	Навантаження на мікросервіс аналітики є нерівномірним, тому для забезпечення масштабованості реалізовано динамічне масштабування відповідно до навантаження.

*	*Математична модель для моделювання поведінки IoT системи на основі розподіленої архітектури. Розділ 2.5*	При проектуванні архітектури системи було використано математичну модель, запропоновану у розділі 2. На етапі моделювання, було описано характер поведінки окремих мікросервісів, та визначено вплив структури системи збереження та обробки даних на продуктивність системи в цілому. Згідно запропонованої моделі, розроблена конфігурація системи є оптимальною, оскільки забезпечується максимальна пропускну здатність (згідно поставлених вимог), але не є «перенасиченою» надлишковими ресурсами.

4.2. Забезпечення відповідності розробленої системи стандартам захисту інформації

У даному розділі аналізується відповідність розробленої системи ряду ключових стандартів захисту інформації. Розглянуто застосовані методи та технології, що забезпечують конфіденційність, цілісність та доступність даних, а також здійснюють контроль доступу та ідентифікацію користувачів. У Таблиці 4.3 наведені основні стандарти захисту інформації, та методи якими забезпечується їх відповідність у реалізованій системі.

Таблиця 4.3. Відповідність системи стандартам захисту інформації

Стандарт захисту інформації	**Методи забезпечення відповідності**	**Опис**
ISO/IEC 27001	1. Використання технологій TLS/SSL для всіх типів MQTT з'єднання (Клієнт-Брокер, Брокер-Підписник).	Міжнародний стандарт, який визначає вимоги до систем управління інформаційною безпекою. Він включає планування, реалізацію, встановлення, контроль,

	2. Використання SSL сертифікатів та захищеного DNS з'єднання для підсистем з DNS доступом (HTTPS). 3. Використання безпечного механізму gRPC з'єднання на основі авторизації та SSL/TLS. 4. З'єднання програмних модулів з базами даних відбувається в захищеній приватній підмережі, та використовують SSL сертифікати. 5. Використання технології OAUTH 2 для реалізації зовнішніх інтеграцій.	вдосконалення та оцінку ефективності системи захисту інформації.
NIST SP 800-53		Стандарт Національного інституту стандартів і технологій США, який надає набір рекомендацій щодо управління ризиками інформаційної безпеки.
PCI DSS	1. Використання технологій TLS/SSL для всіх типів MQTT з'єднання (Клієнт-Брокер, Брокер-Підписник). 2. Використання SSL сертифікатів та захищеного DNS з'єднання для підсистем з DNS доступом (HTTPS). 3. Платіжні дані користувача не зберігаються в системі.	Стандарт захисту платіжної інформації.
GDPR	1. Використання технологій TLS/SSL для всіх типів MQTT з'єднання (Клієнт-Брокер, Брокер-Підписник). 2. Використання SSL сертифікатів та захищеного DNS з'єднання для підсистем з DNS доступом (HTTPS). 3. Паролі користувача зберігаються в хешованому вигляді, на основі криптографічного методу SHA256.	Ця загальна регламентація про захист даних в Європейському Союзі встановлює правила щодо збору, обробки та зберігання персональних даних. Даний стандарт включає вимоги до забезпечення конфіденційності та прав користувачів.

	4. Системи баз даних використовують зашифрований дисковий простір.	
OAuth 2.0	1. Використання SSL сертифікатів та захищеного DNS з'єднання для підсистем з DNS доступом (HTTPS). 2. Використання JWT токенів, для авторизації запитів. 3. Розроблено систему управління ідентифікацією (Сервер авторизації)	Протокол авторизації, який дозволяє стороннім системам отримувати обмежений доступ до ресурсів користувача в даній системі без передачі йому його облікових даних.

4.3. Розробка програмного забезпечення

Для проведення дослідження була використана методологія тестової розробки (TDD - Test Driven Development). TDD представляє собою специфічний процес розробки програмного забезпечення, який передбачає створення вимог до програми шляхом визначення очікуваного результату за допомогою програмних тестів перед тим, як фактичний код програми буде повністю розроблений [101]. Цей підхід суттєво відрізняється від традиційного підходу, де спершу розробляється сам код, а потім вже створюються тести. На першому етапі створюються тести для нового функціоналу чи вимоги, яку потрібно реалізувати. Ці тести перевіряють очікувані результати від програмного забезпечення, яке ще не існує. На наступному етапі розробляється програмний код, який має пройти тести.

Після написання коду запускаються тести знову. Очікується успішне їх виконання, що підтвердить правильність реалізації функціоналу. Через цикл створення тестів, розробки коду та перевірки їх виконання, досягається більша стійкість та надійність системи в цілому. На Рисунку 4.2 наведено алгоритм методології тестової розробки.

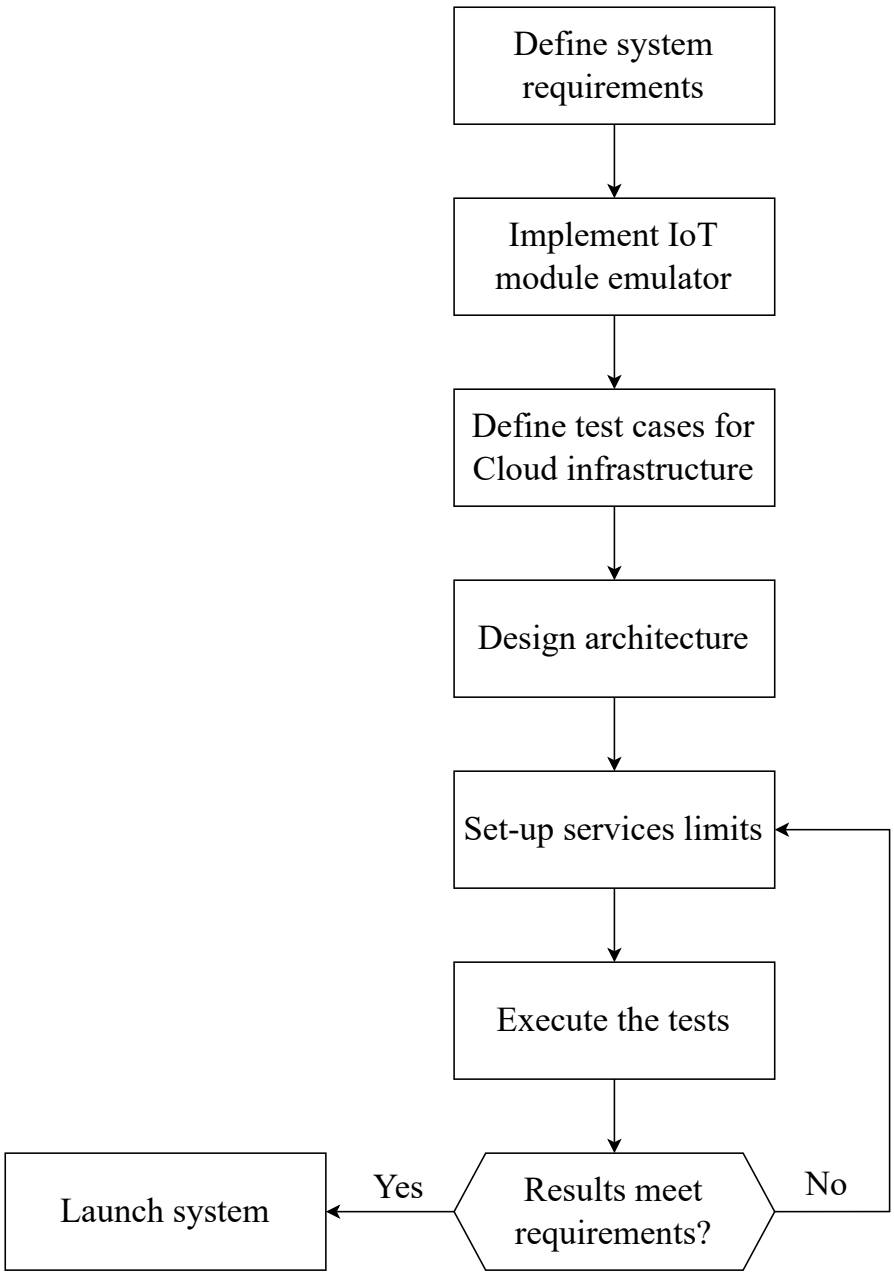


Рис 4.2. Алгоритм методології тестової розробки

В ході даного дослідження було виконано програмну реалізацію для ряду модулів системи збереження та обробки даних, а також програмне забезпечення прототипу IoT модуля. З метою перевірки функціональності системи був розроблений емулятор для прототипу IoT модуля. Розроблений емулятор дозволив відтворити різні сценарії поведінки та інтенсивності передачі даних.

Система збереження та обробки даних складається з п’яти підсистем, які реалізовані згідно концепції мікросервісної архітектури. А саме: Підсистема

моніторингу руху (далі підсистема 1), Платіжна система (далі підсистема 2), Підсистема аналітики даних (далі підсистема 3), Підсистема машинного навчання (далі підсистема 4) та Підсистема зовнішніх інтеграцій (далі підсистема 5). Структура архітектури системи та її взаємозв'язки зображені на Рисунку 4.1. Основна ідея полягає у тому, що кожен мікросервіс відповідає за специфічний обмежений функціональний блок та працює незалежно від інших компонентів. Забезпечення взаємодії між мікросервісами відбувається за допомогою протоколів HTTP(S) та gRPC, залежно від характеру даних, що передаються.

Для розробки програмного забезпечення підсистем 1-3 та 5 було використано мову програмування TypeScript та платформу для виконання серверного JavaScript-коду - Node.js. TypeScript є розширенням JavaScript, яке надає можливість явно визначити типи та підтримує використання класових компонентів, при цьому забезпечуючи зворотню сумісність з JavaScript.

В якості високорівневого фреймворка для розробки серверних систем було використано Nest.js (рис. 4.3). Обрані технології відповідають подієво-орієнтованій моделі системи, а також вимозі збереження та обробки даних в асинхронному режимі.

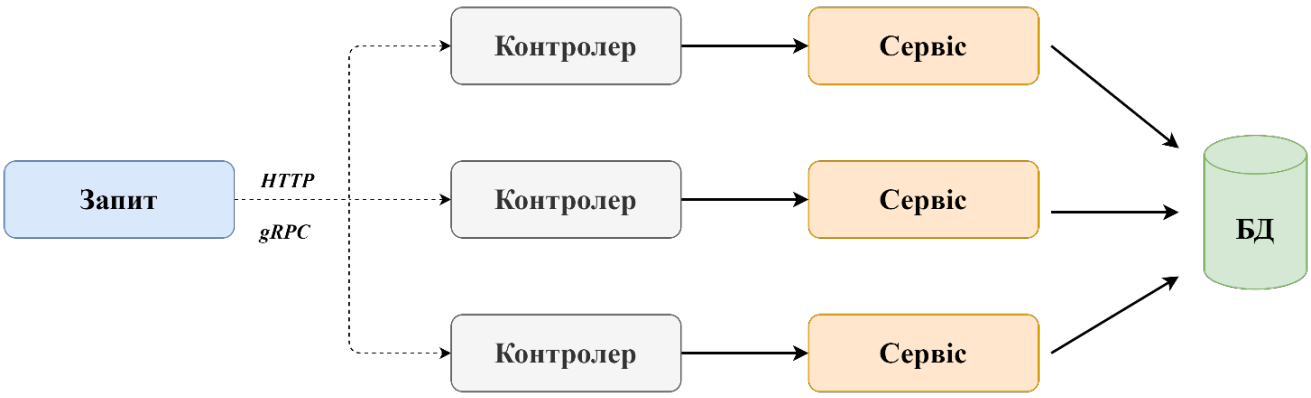


Рис 4.3. Структурна схема веб-сервісу на основі фреймворку Nest.js

При розробці підсистеми 4 було використано мову програмування Python, а також бібліотеки для машинного навчання – TensorFlow та Keras. Мова Python є популярним вибором у дослідженнях машинного навчання та аналізі даних завдяки зручним інструментам для роботи з даними. Для створення та навчання

моделей машинного навчання були використані бібліотеки TensorFlow та Keras. TensorFlow - це відкрите програмне забезпечення для розробки та навчання моделей глибокого навчання та машинного навчання [102]. Він надає широкий набір інструментів для створення різних типів моделей, зокрема для класу рекурентних нейронних мереж, LSTM та GRU архітектур, які були використанні в даному дослідження.

Keras, є високорівневим інтерфейсом над бібліотекою TensorFlow [103], спрощуючи процес розробки моделей машинного навчання. Окрім бібліотек машинного навчання, дана система використовує адаптери для роботи з базами даних. На основі історичних даних, які система отримує з бази даних на основі часових рядів, формуються навчальні вибірки та відбувається тренування моделі.

В Таблиці 4.4 наведено повний список реалізованих програмних модулів, а також технологій, на яких вони побудовані. Програмні реалізації наведені у додатках.

Таблиця 4.4. Програмні компоненти та технології реалізації

Компонент	**Мова програмування**	**Фреймворк, бібліотека**
IoT модуль	C++, Python	-
Емулятор IoT модуля	Python, TypeScript	Node.js
MQTT брокер	JavaScript	Node.js
Підсистема моніторингу руху	TypeScript	Node.js, Nest.js
Платіжна система	TypeScript	Node.js, Nest.js
Підсистема машинного навчання	Python	Keras, TensorFlow, FastAI

Підсистема аналітики даних	TypeScript	Node.js, Nest.js
Панель керування	TypeScript	Node.js, React.js

4.4. Процес розгортання програмного забезпечення системи

Процес розгортання програмного забезпечення є частиною циклу розробки та забезпечує безперервну доставку розробленого чи оновленого програмного коду на цільовий сервер для його виконання. В розробленій системі використовується мікросервісна архітектура, де кожен функціональний компонент існує як інкапсульований обчислювальний контейнер. В якості системи контейнеризації використовується платформа Docker. Docker це інструментарій для управління ізольованими Linux-контейнерами, що доповнює інструментарій LXC більш високорівневим API, та дозволяє керувати контейнерами на рівні ізоляції окремих процесів [104]. Процес контейнеризації мікросервісу починається з опису залежностей та оточення, необхідних для правильної роботи програмного забезпечення, у спеціальному файлі конфігурації. На основі цієї конфігурації створюється образ контейнера, який містить усі необхідні компоненти, бібліотеки та конфігураційні файли. Застосування Docker дозволяє забезпечити консистентність середовища від розробки до розгортання, зменшити можливі конфлікти між залежностями та спростити процес розгортання нових версій або оновлень програмного забезпечення на серверах. Це сприяє більшій надійності та простоті управління розподіленими компонентами системи. Процес контейнеризації мікросервісу наведено на Рисунку 4.4.

Важливим етапом процесу розгортання програмного забезпечення, є проведення автоматичного тестування, перед доставкою на сервер. Автоматичне тестування передбачає автоматизовану перевірку функціональності, стійкості та

надійності розробленого або оновленого коду перед його запуском на цільовому сервері.

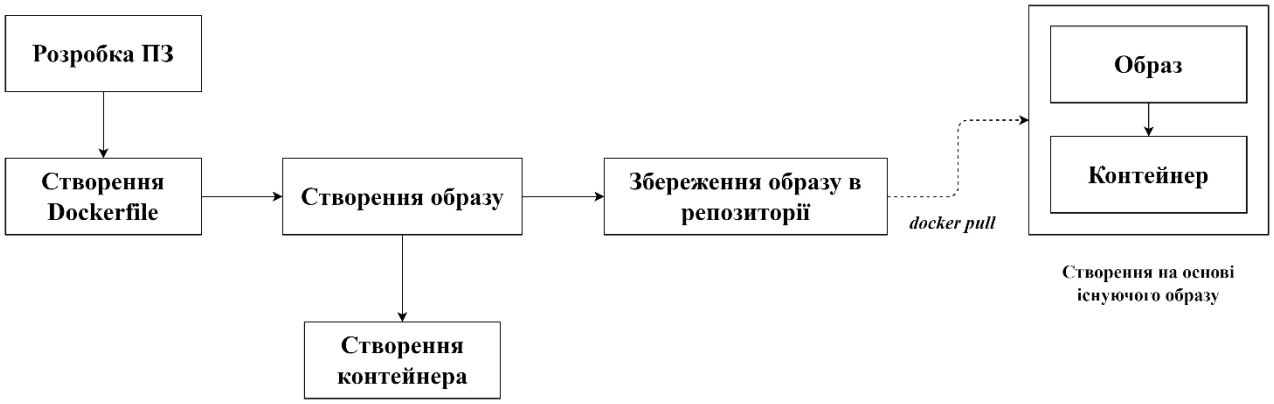


Рис 4.4. Процес Docker контейнеризації

В даній реалізації, виконується набір тестів описаних в розділі 4.3. В разі успішного виконання автоматичних тестів, на серверах оновлюється програмне забезпечення, в разі помилки процес повертається та етап розробки.

У реалізованій системі використовується низка практик та інструментів для реалізації неперервної інтеграції та неперервної доставки - CI/CD (англ. Continuous integration, Continuous delivery). Це процес автоматизації розгортання, тестування та доставки програмного коду [105]. Алгоритм даного процесу наведений на Рисунку 4.5.

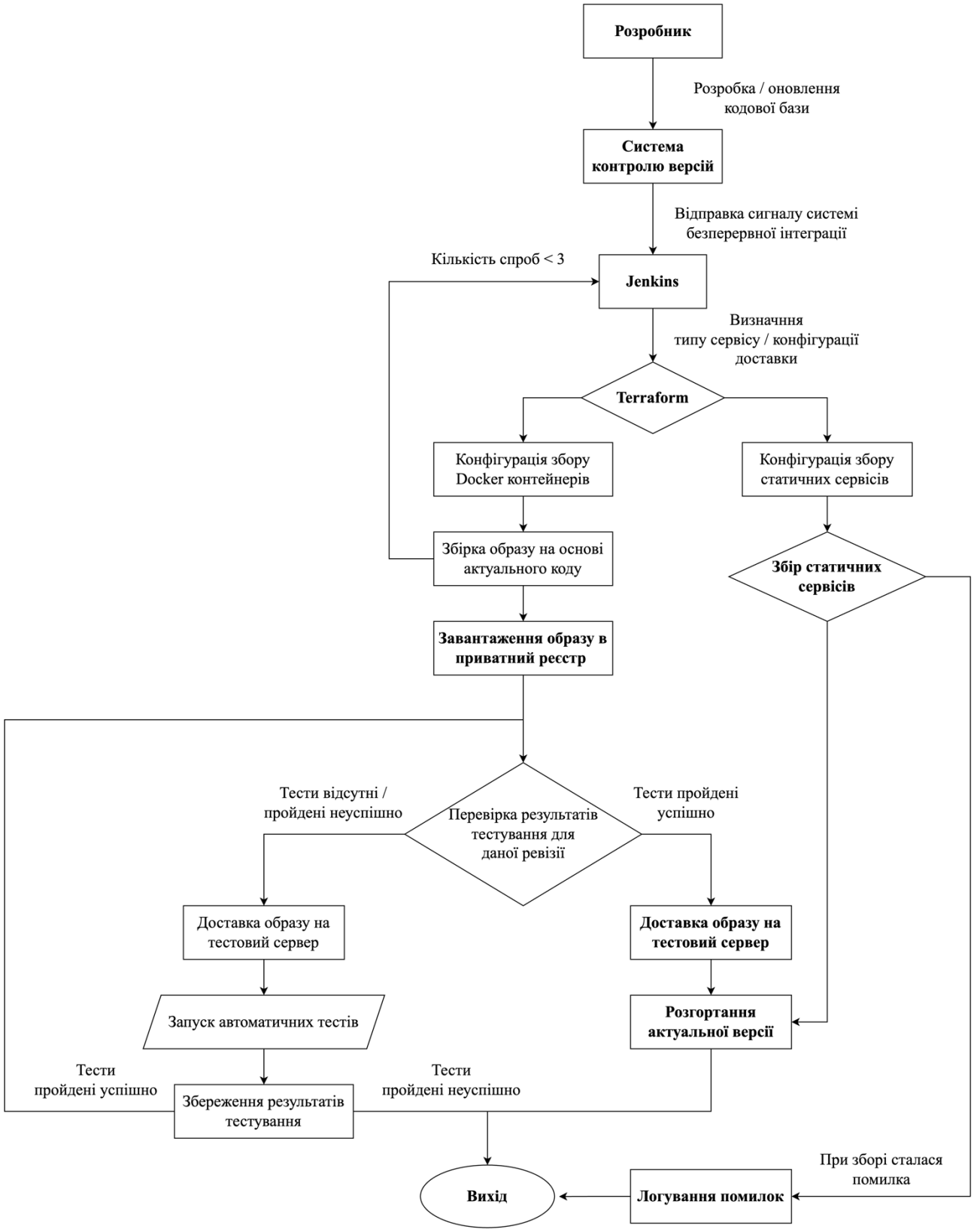


Рис 4.5. Алгоритм розгортання програмного забезпечення

4.5. Методика тестування реалізованої системи

Дослідження проводилось з використанням прототипів, та емулятора автомобільного модуля (IoT контролера), та розгорнутої серверної інфраструктури на платформі AWS IoT Core (рис. 4.7). Схема розробленого експериментального середовища наведена на Рисунку 4.6.

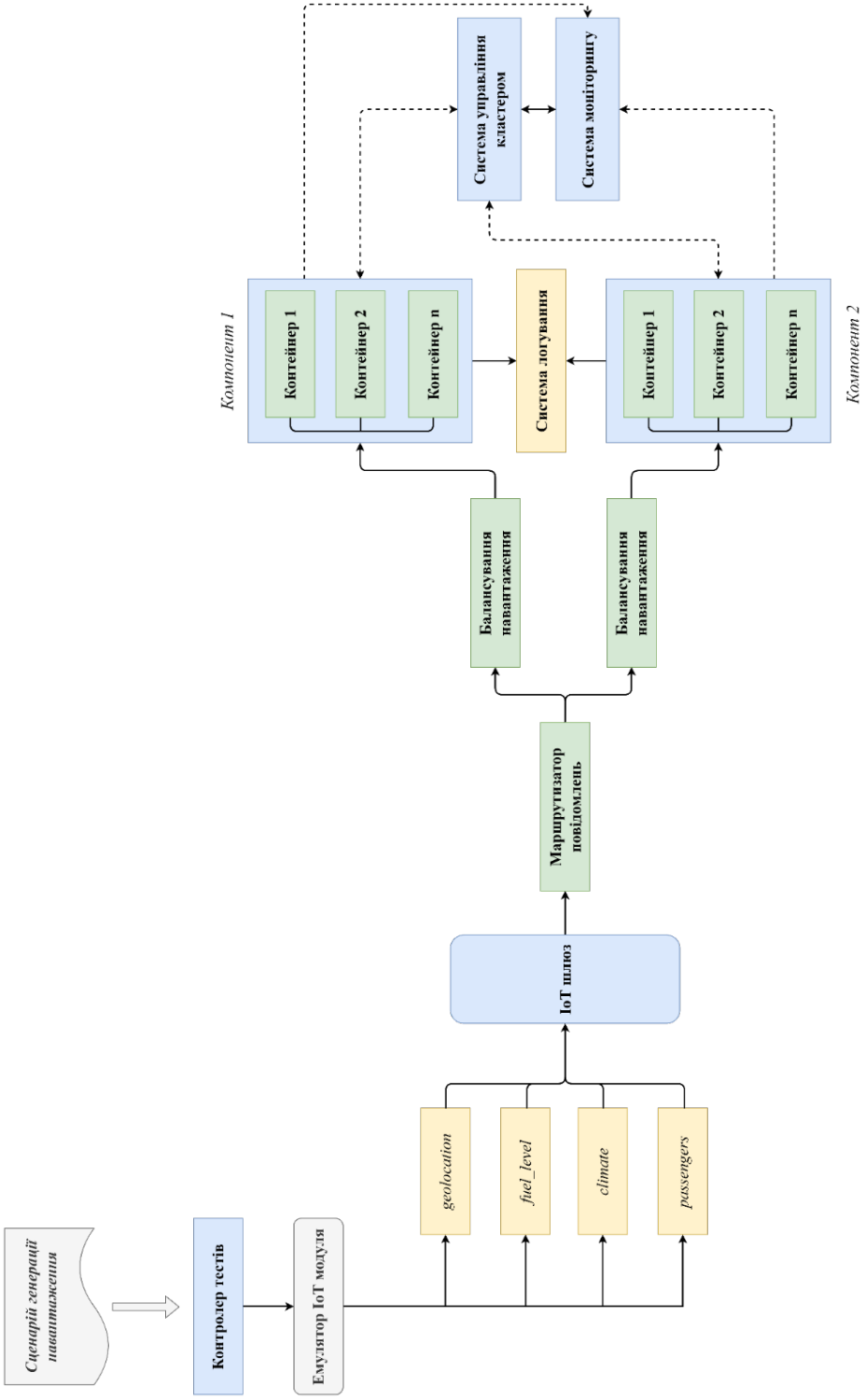


Рис. 4.6. Схема експериментального середовища

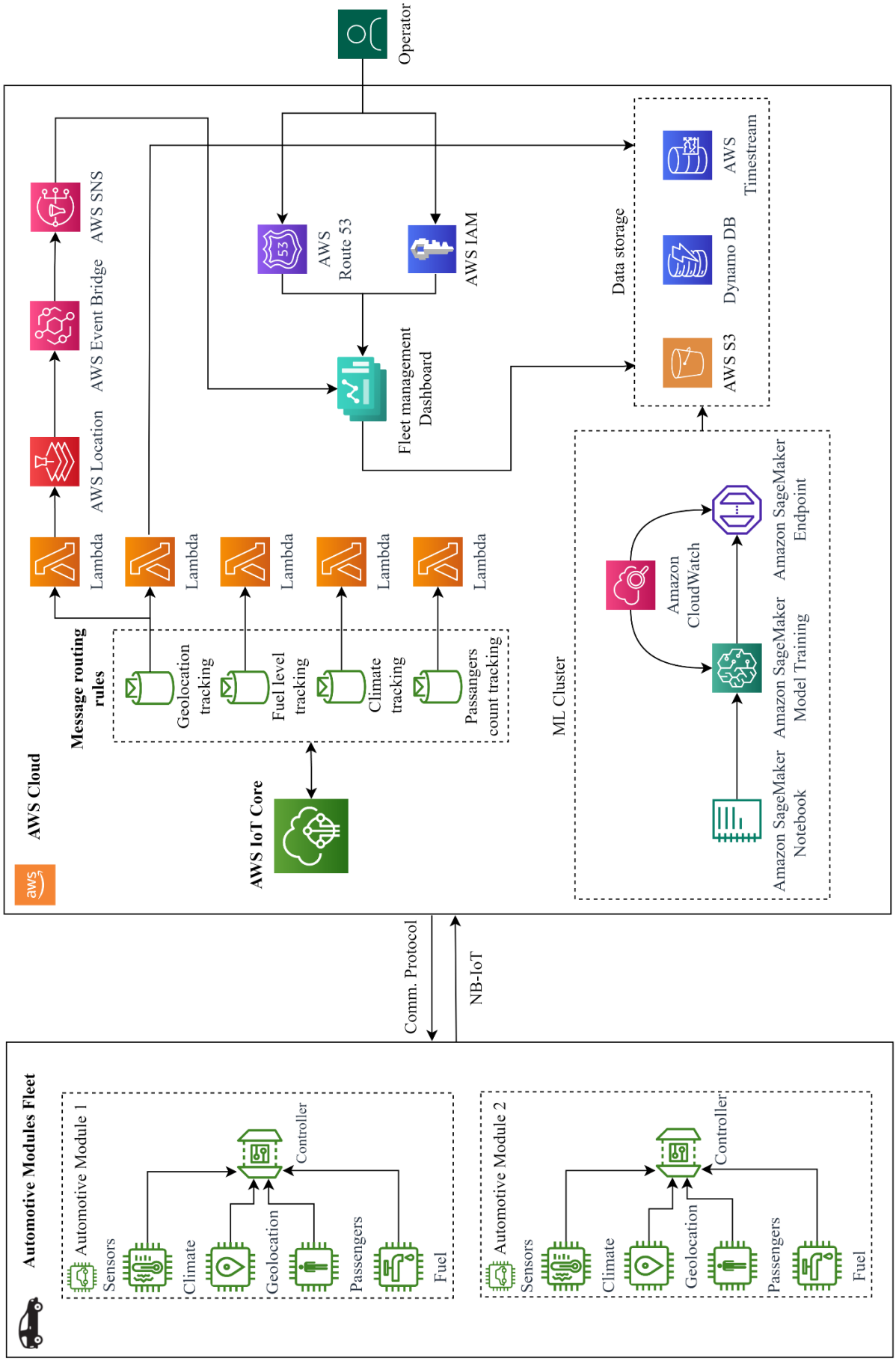


Рис. 4.7. Архітектура ІТС на базі AWS IoT Core

У цьому дослідженні було взято за основу систему наземного міського транспорту м. Київ, Україна, і проведено моделювання та прогнозування навантаження відповідно до цих даних. Громадський наземний транспорт м. Київ складається з автобусів, тролейбусів і трамваїв, якими щодня користується в середньому 1,1 млн жителів [106] (таб. 4.5).

Таблиця 4.5

Стан громадського транспорту Києва, 2019 рік

Тип транспорту	**Маршрути**	**Кількість транспортних засобів**
Автобуси	105	400
Тролейбуси	50	370
Трамваї	20	290

В процесі дослідження було прийнято припущення, що всі транспортні одиниці будуть одночасно знаходитись на маршруті, і тестове навантаження буде плануватися відповідно до цього обсягу даних і швидкості передачі.

В структурі системи IoT контролер, є основним джерелом даних, яке передає та отримує інформацію з сервера через протокол програмного рівня – MQTT. У Таблиці 4.6 наведено основні типи даних та розділи MQTT, які використовуються для обміну повідомленнями між пристроєм і сервером.

Таблиця 4.6

Розділи MQTT

Розділ MQTT	**Розмір (байт)**	**Інтервал (секунди)**	**Пропускна здатність**	**Опис даних**
geolocation	100	1	12	Поточні координати
fuel_level	30	1	6	Рівень палива/Заряд батареї
climate/temperature	30	60	1	Температура
climate/humidity	30	60	1	Вологість, %
passengers_count	30	60 секунд після події "Стоп"	Макс. 1	Кількість пасажирів

Метою тестування є відтворення навантаження, яке створюватиме система інтелектуального громадського транспорту м. Києва (таб. 4.5). Вважається, що в кожній транспортній одиниці встановлено автомобільний модуль, який передає дані на сервер через NB-IoT з'єднання, а також, що всі транспортні одиниці знаходяться на маршруті одночасно. Необхідно оцінити продуктивність і масштабованість системи відповідно до зміни навантаження: кількості підключених пристроїв, та об'єму даних, який вони генерують.

Для тестування розроблено емулятор автомобільного модуля. Для імітаційних даних емулятор використовує набір точок геолокації маршруту тролейбуса в м. Київ, а також випадково згенеровані дані про температуру, вологість, рівень палива та кількість пасажирів.

Нижче наведено основні результати та ключові показники проведених тестів. Навантаження генерувалося протягом 45 хвилин.

Таблиця 4.7. Характеристики серверів

Параметр	**Значення**
Тип сервера	t3.xlarge
vCPUs	4
Оперативна пам'ять	16
Пропускна здатність мережі	1 GB/s
Операційна система	Ubuntu

На основі моніторингових даних, що були отримані впродовж періоду тестування було визначено рівномірність розподілу навантаження на основі методу запропонованого у Розділі 3.

4.6. Результати тестування

На Рисунках 4.9 та 4.10 наведені значення розрахованого коефіцієнту рівномірності розподілу навантаження для Компоненту 1 та Компоненту 2 відповідно. На основі отриманих результатів та провівши кореляцію з графіком (рис. 4.8) можна зробити висновок, що при використанні існуючих методів балансування (які застосовані в Системі 1), при збільшенні вхідного навантаження знижується ефективність використання серверних ресурсів.

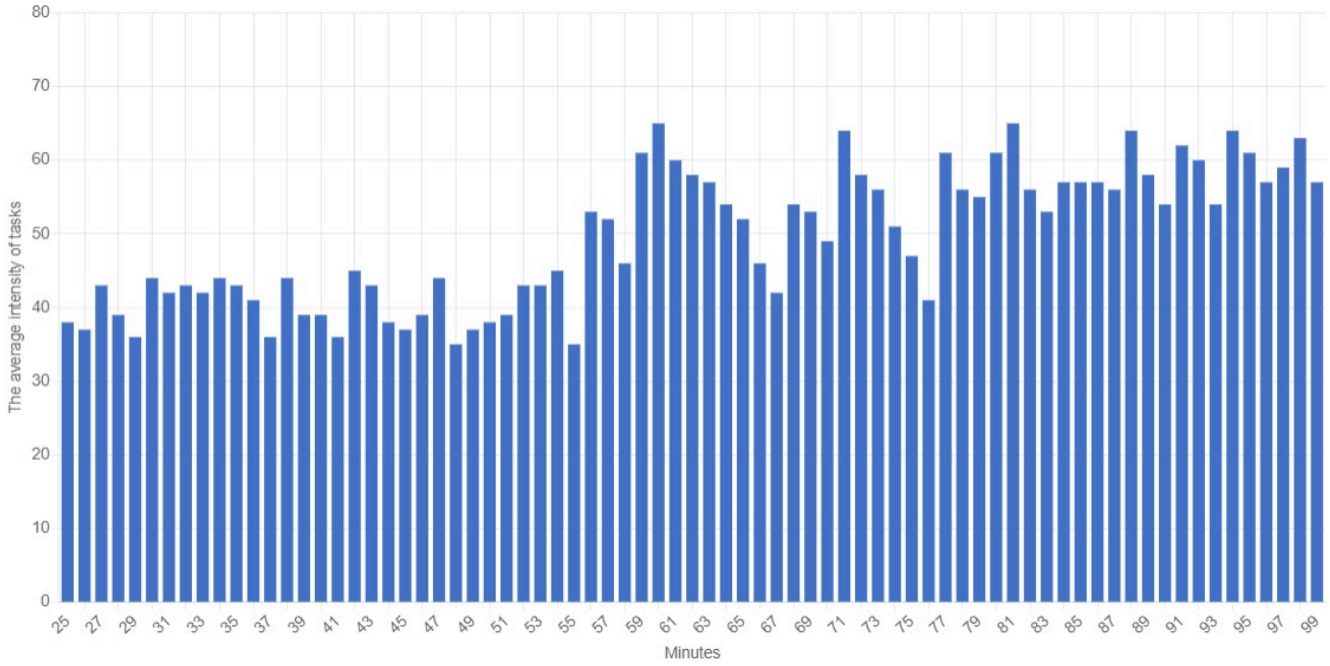


Рис. 4.8. Інтенсивність вхідного навантаження

Це підтверджують результати отримані для Компоненту 1 та Компоненту 2.

Натомість, коефіцієнт розподілу для Системи 2, яка використовує запропонований у розділі 3.2 метод балансування є нижчим протягом усього періоду тестування на 40-55% порівняно з Системою 1.

Також важливою перевагою є те, що даний показник для Системи 2 є практично незмінним протягом усього періоду тестування. Це означає, що продуктивність системи не змінюється при зміні вхідного навантаження. Це є підтвердженням того, що дана система є робастною та масштабованою.

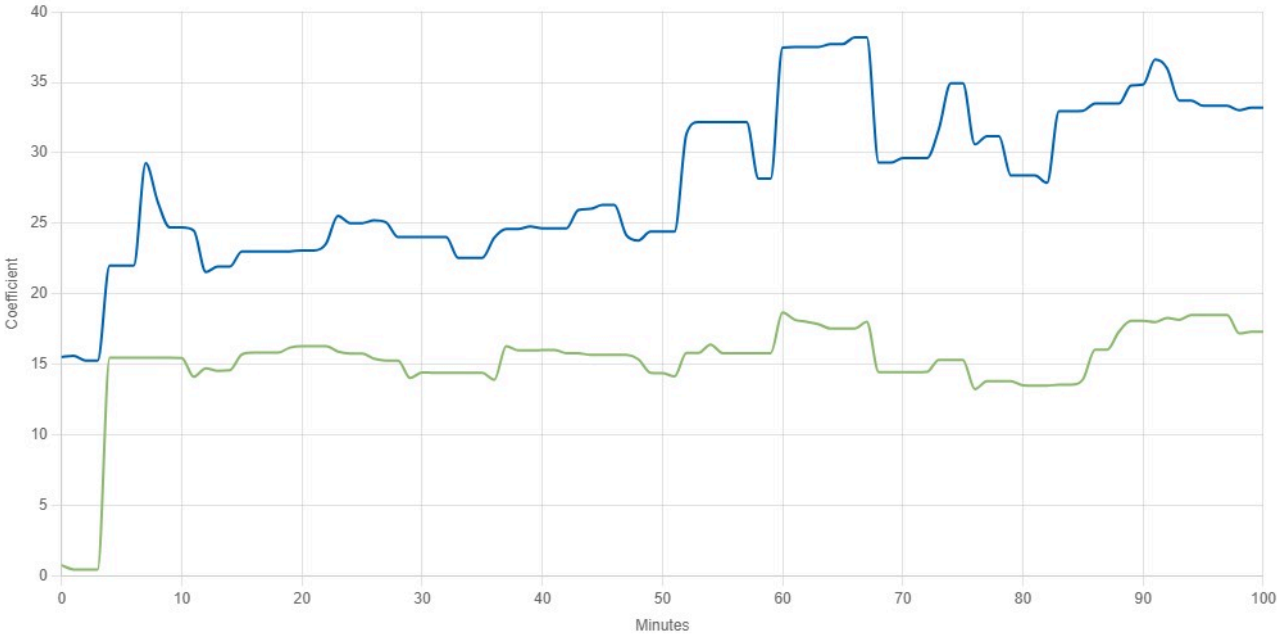


Рис. 4.9. Коефіцієнт рівномірності розподілу навантаження – Сервіс 1

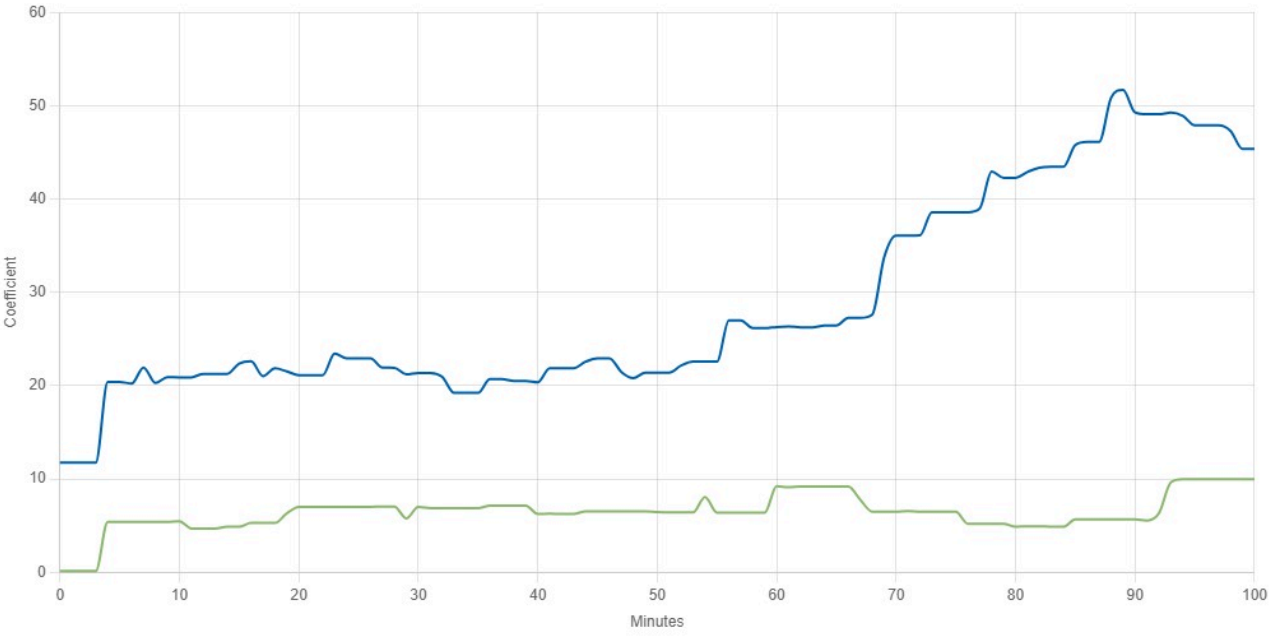


Рис. 4.10. Коефіцієнт рівномірності розподілу навантаження – Сервіс 2

Крім того, аналіз графіків показує, що на початковому етапі тестування, коли навантаження було незначним, коефіцієнт для Системи 2 дорівнював нулю. Це ілюструє той факт, що система не використовує зайві обчислювальні ресурси під час періодів простою. В той же час, для Системи 1, значення цього коефіцієнта становило 15 для Компоненту 1 та 11 для Компоненту 2, що є

результатом неоптимального використання обчислювальних ресурсів в результаті неефективного балансування навантаження.

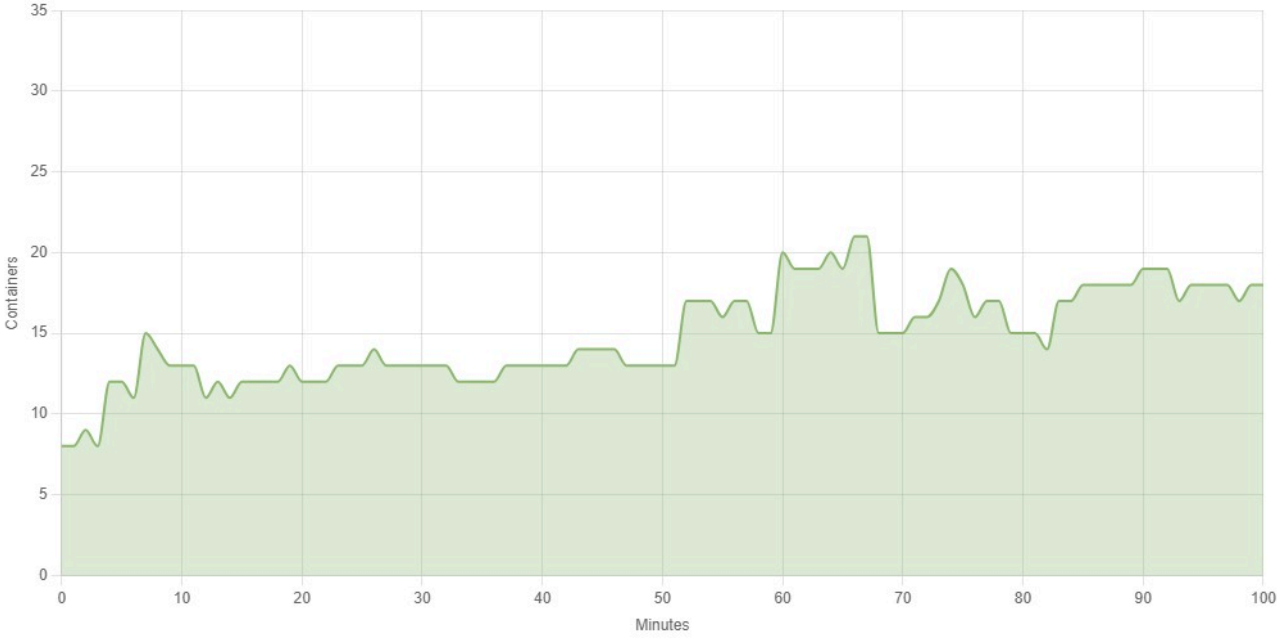


Рис. 4.11. Кількість активних обчислювальних контейнерів – Компонент 1, AWS IoT Core

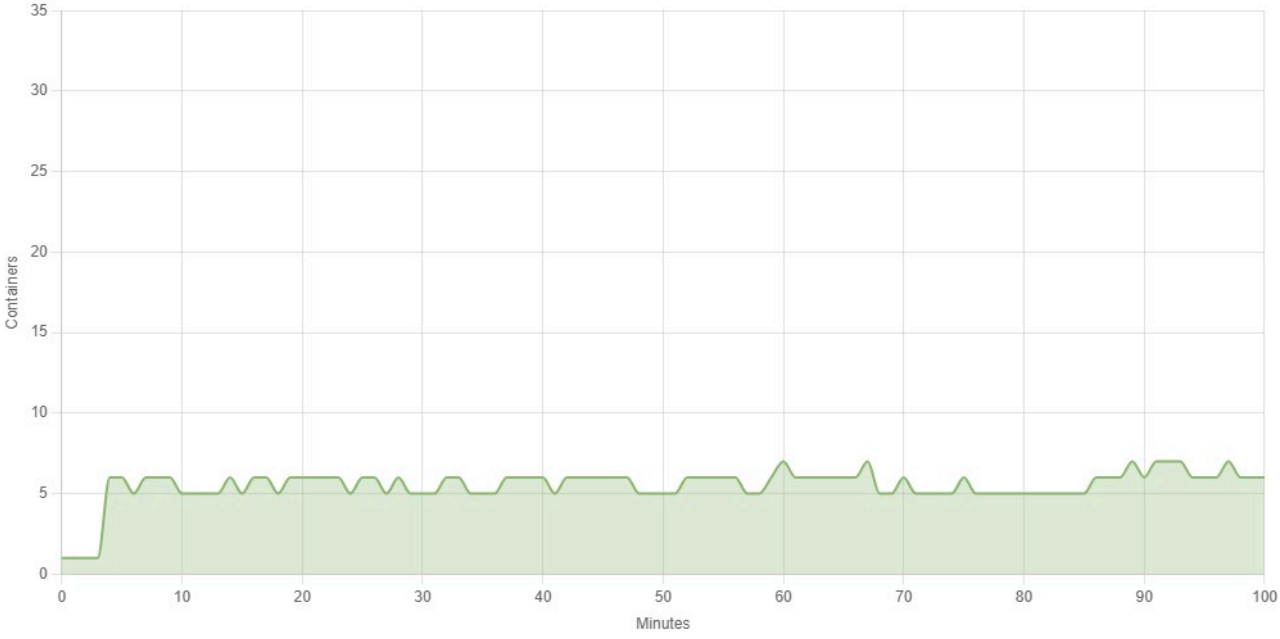


Рис. 4.12. Кількість активних обчислювальних контейнерів – Компонент 1, розроблена система

Отримані результати також корелюють з кількістю обчислювальних контейнерів (рис. 4.11-4.14). А саме з графіків видно, що для Системи 1 кількість

активних обчислювальних контейнерів є більшою на 30-65% протягом усього періоду тестування ніж для Системи 2.

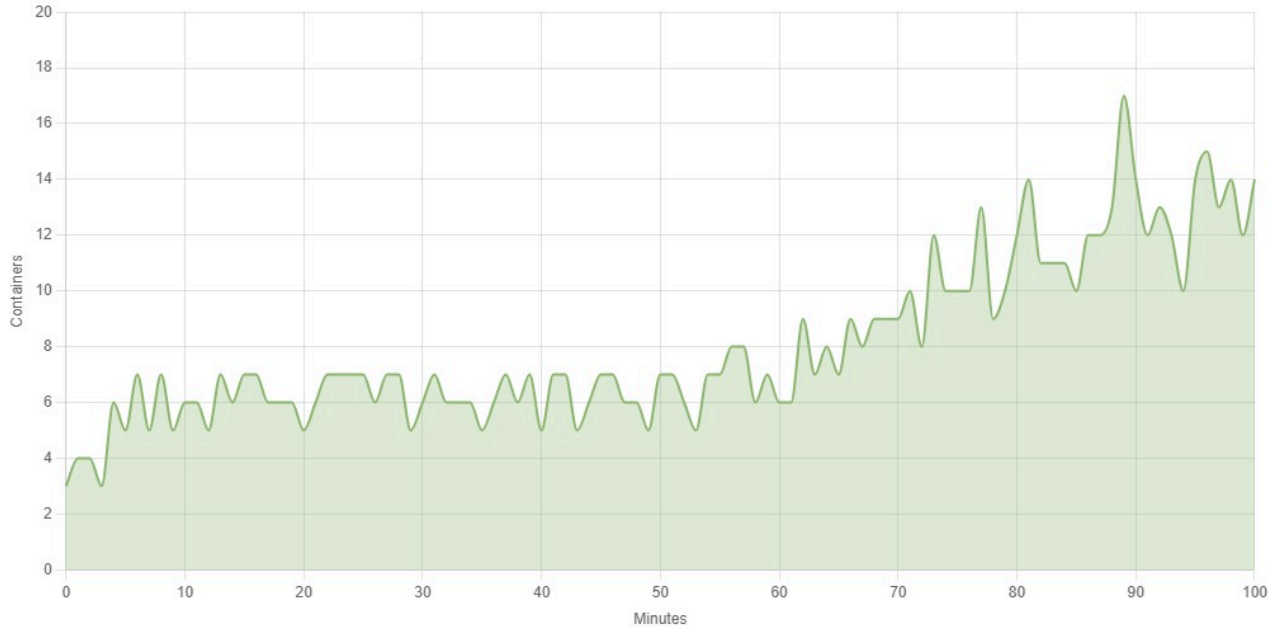


Рис. 4.13. Кількість активних обчислювальних контейнерів – Компонент 2, AWS IoT Core

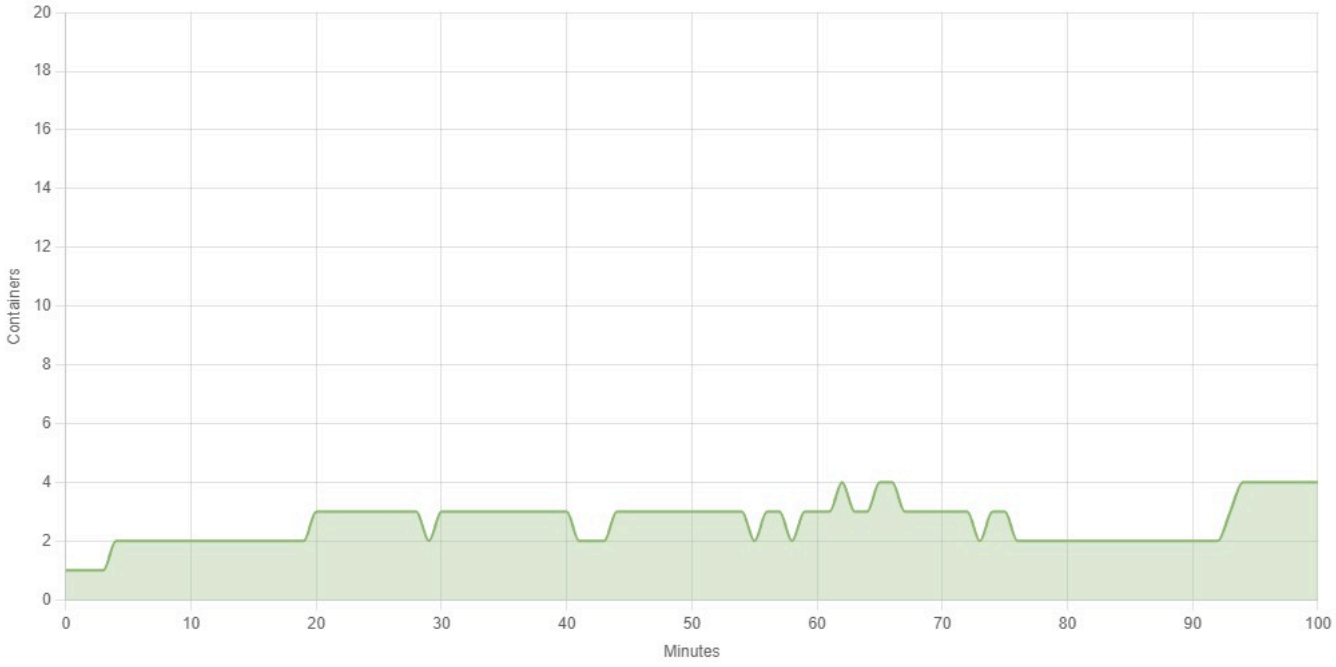


Рис. 4.14. Кількість активних обчислювальних контейнерів – Компонент 2, розроблена система

Дані результати пояснюються тим, що в Системі 2 застосовані запропоновані у розділі 3 методи. А саме багатопараметричного моніторингу навантаження, а також метод визначення оптимальної кількості обчислювальних ресурсів на основі MILP моделі.

В той же час, Система 1 використовує класичний підхід до збільшення об'єму обчислювальних ресурсів, на основі моніторингу значення завантаженості процесора, що як показали результати є неефективним.

Для оцінки впливу застосованих методів на швидкість обробки даних, було виконано збереження результатів та часу виконання функції для обробки кожного повідомлення.

На основі агрегованих значень, для зручності проведення кореляцій з графіками (рис. 4.9 – рис. 4.14) виконаємо квантування результатів:

$$T = \frac{\sum_{k=1}^{N} x_k}{N}, \quad (4.1)$$

де, x_k – час обробки повідомлення в момент часу k, N - кількість вимірювань у даній вибірці.

На Рисунках 4.15 та 4.16 наведено результати швидкості обробки даних в рамках Компоненту 1 для Системи 1 та 2.

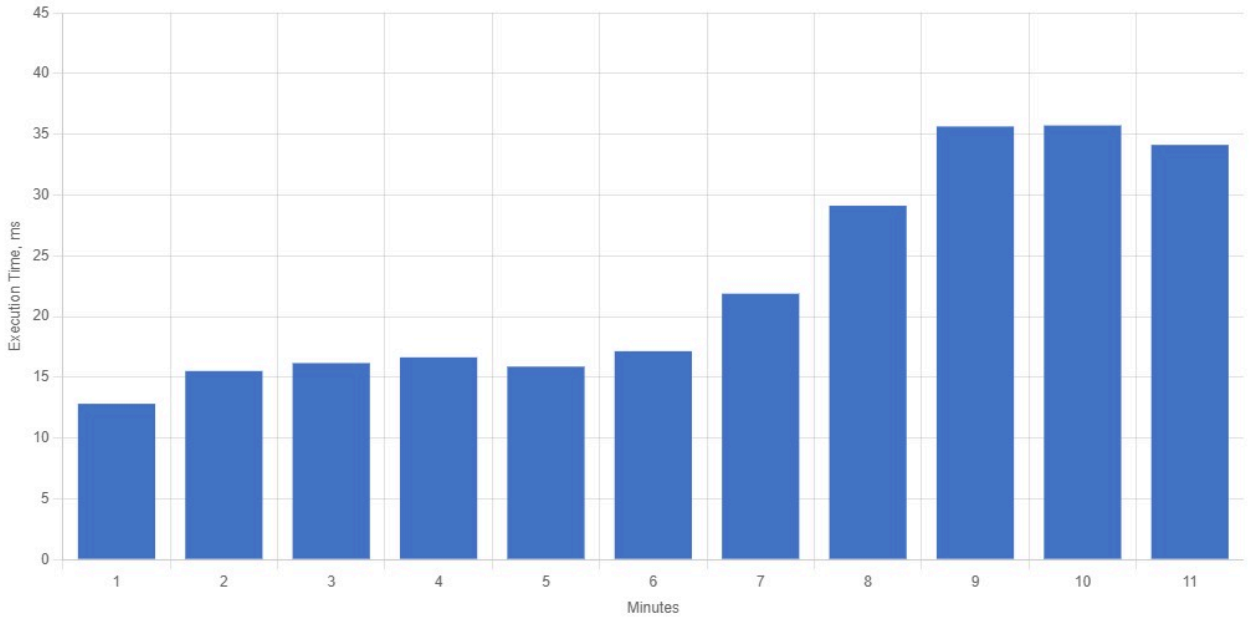


Рис. 4.15. Час обробки повідомлення – AWS IoT Core

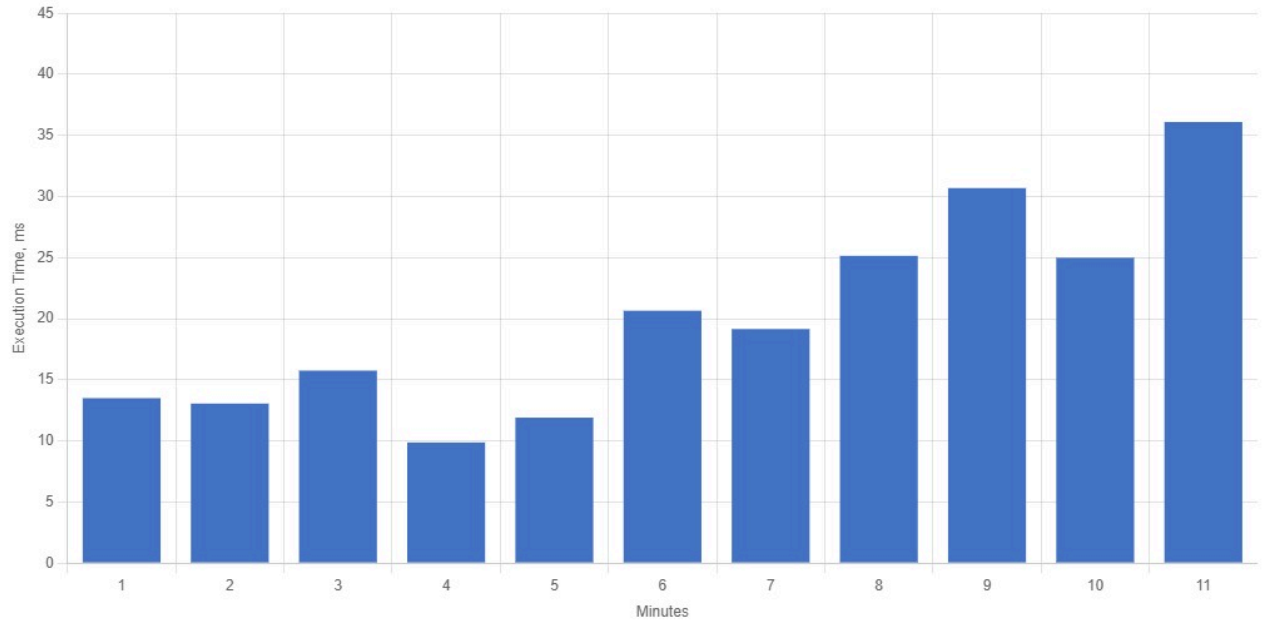


Рис. 4.16. Час обробки повідомлення – розроблена систем

Аналізуючи отримані значення, а також динаміку зміни, можна зробити наступні висновки:

- Середній час обробки повідомлення суттєво не відрізняється для обох розглянутих систем;
- Збільшення інтенсивності вхідного навантаження призводить до збільшення часу обробки даних. Протягом періоду тестування середній час обробки одного повідомлення змінювався від 13 до 36 мс;
- Проведена з графіками (рис. 4.11 – рис. 4.12) кореляція, показує, що Система 2 використовувала меншу кількість обчислювальних контейнерів, забезпечивши ту саму швидкість обробки даних. Даний результат ще раз підтверджує ефективність запропонованих методів.

4.7. Економічний ефект запропонованих методів

На основі результатів проведених експериментів, можемо визначити використаний об’єм обчислювальних ресурсів для обох систем, а отже оцінити економічний ефект запропонованих методів.

Платформа Amazon Web Services, на які були розгорнуті досліджувані системи в якості стратегії ціноутворення використовує модель Pay-As-You-Go [107]. Ця модель передбачає, що сплачуються лише ті обчислювальні ресурси, які були фактично використані, замість попередньої оплати за фіксований обсяг ресурсів. На основі цієї моделі було виконано прорахунок вартості ресурсів, необхідних для Системи 1 та Системи 2.

В ході тестування використовувались сервери типу EC2 t3.xlarge (таб. 4.7). Ціна даного типу сервера, на момент проведення тестування становила 0.0027$ за хвилину [108]. На основі даних про кількість обчислювальних контейнерів (рис. 4.11 – рис. 4.14), можемо провести розрахунок вартості використаних ресурсів, для двох досліджуваних компонентів та систем. Результати розрахунків наведені у Таблиці 4.8, та відображають вартість використаних обчислювальних ресурсів для обраної конфігурації системи та періоду використання.

$$Server\ Cost\ = \sum_{m=1}^{N} x_m * perMinutePrice, \tag{4.2}$$

де x_m – кількість активних контейнерів, N кількість вимірювань у даній вибірці.

Таблиця 4.8. Розрахунок ціни обчислювальних ресурсів

Період	**Ціна**			
	Компонент 1		Компонент 2	
	Система 1	Система 2	Система 1	Система 2
100 хвилин	3.72$	**1.6$**	3.64$	**1.59$**
Доба	53.56$	**23.04$**	52.41$	**22.89$**
Місяць	1660$	**714.24$**	1624.71$	**709.59$**

Одержані результати демонструють, що система побудована на основі запропонованих методів демонструє не лише технічну ефективність, а й значне зниження економічних витрат. Зокрема, порівняно з існуючою системою,

вартість експлуатації запропонованої системи зменшилася вдвічі, становлячи 1423$ проти 3284$ на місяць.

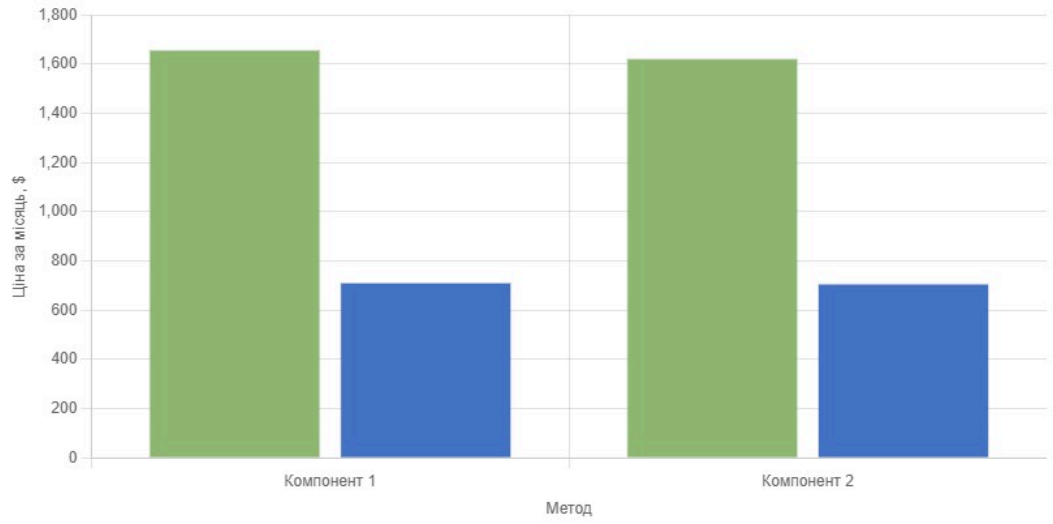


Рис. 4.17. Ціна обчислювальних ресурсів, які використовують досліджувані системи на базі платформи Amazon Web Services за місяць

Результати тестування відносяться до двох конкретних компонентів системи, але їхні характеристики та динаміка розподілу визначають загальну тенденцію для всієї системи. Таким чином, динаміка зниження вартості та

оптимізації ресурсів, знайдена для досліджуваних компонентів, відображає тенденції, які простежуються для всієї системи в цілому.

4.8. Висновки

1. Виконано реалізацію інтелектуальної системи громадського транспорту на основі розроблених методів та запропонованої архітектури.
2. Виконано програмну реалізацію компонентів системи, розроблено алгоритм процесу розгортання програмного забезпечення на серверах.
3. Проведено аналіз та обґрунтовано відповідність розробленої системи основним стандартам захисту інформації.
4. Для проведення оцінки ефективності розробленої системи, а також її порівняння з існуючими комерційними рішеннями запропоновано методику тестування, а також розроблено програмне середовище. Оцінка ефективності

системи проводилась по ключових характеристиках робастності, які були виділенні в завданні даного дослідження. Система розроблена на основі запропонованих методів, показала перевагу по вищенаведених характеристиках. А саме: рівномірність розподілу навантаження є вищою на 40-55%, а об’єм використаних обчислювальних ресурсів є меншим на 50-60%.

5. Проведено аналіз економічного ефекту від впровадження запропонованих методів.

ВИСНОВКИ

У дисертаційній роботі, на основі запропонованих методів та проведених експериментальних досліджень розроблено масштабовану та робастну систему для моніторингу та управління громадським транспортом. В ході даної роботи було виконано ряд науково-технічних задач. А саме:

1. Проведено багатокритеріальний аналіз існуючих комерційних рішень Інтелектуальних транспортних систем, визначено їх переваги, недоліки та обмеження. Також проведено аналіз існуючих наукових методів побудови систем Інтернету речей. Визначено методи, які потребують вдосконалення та проведення подальших досліджень. На основі проведеного аналізу сформоване завдання дослідження, а також розроблена методика його проведення.

2. На основі визначених вимог до системи розроблено структуру телекомунікаційної мережі та виділено інформаційні потоки системи. Вибір оптимальної технології бездротової передачі даних проведено на основі аналізу існуючих технологій та результатів експериментальних досліджень за розробленим алгоритмом.

3. Досліджено існуючі протоколи передачі даних програмного рівня в контексті Інтелектуальних систем громадського транспорту. З метою вибору протоколів розроблено методику та алгоритм оцінки продуктивності протоколу, програмно реалізовано тестове середовище. Проведено ряд практичних експериментів для обґрунтування вибору оптимальної технології передачі даних між IoT пристроєм та сервером, а також для мікросервісної комунікації.

4. Виконано програмну реалізацію компонентів телекомунікаційної мережі на основі концепції розподіленої мікросервісної архітектури. Для оцінки впливу архітектури, а також конфігурації окремих компонентів на пропускну здатність системи та її загальну продуктивність, розроблено математичну модель для моделювання поведінки IoT системи на основі розподіленої архітектури.

5. Запропоновано архітектуру системи збереження та обробки даних, сформульовано основні завдання даної системи, а також побудовано її функціональну схему.

6. На основі запропонованої математичної моделі розроблено метод для визначення оптимальної кількості обчислювальних контейнерів в розподілених системах Інтернету речей. Виконано програмну реалізацію даного методу. Даний метод на відміну від існуючих методів враховує динамічні вимоги до обчислювальних систем, які генеруються IoT пристроєм, що дозволяє знаходити оптимальну кількість обчислювальних контейнерів. Застосування даного методу дозволяє підвищити ефективність використання обчислювальних ресурсів на 30-65%, що підтверджують результати експериментів проведених у Розділі 4. Даний метод може бути застосований не лише при побудові Інтелектуальних систем громадського транспорту, а і при побудові систем Інтернету речей в цілому, оскільки є незалежним до типів та структур даних, що передаються в системі.

7. Розроблено математичну модель для оцінки завантаженості обчислювальних контейнерів, а також модель для оцінки рівномірності розподілу навантаження в системі. Запропонована модель, на відміну від існуючих виконує багатопараметричний моніторинг обчислювальних контейнерів (серверів), що дозволяє підвищити точність та ефективність балансування в системах з динамічним навантаженням. На основі запропонованих моделей розроблено алгоритм балансування навантаження. Розроблено покращену архітектуру MQTT брокера, який виконує балансування навантаження, згідно з розробленим алгоритмом. Виконано програмну реалізацію MQTT брокера, а також бібліотеки для підключення клієнтів (підписників). Запропоновано методику тестування, а також виконано програмну реалізацію середовища тестування. Проведено комплекс експериментів для порівняння запропонованих методів та існуючих. Аналіз результатів експериментів підтвердив, що запропоновані методи підвищують ефективність використання основних ресурсів системи і демонструють перевагу запропонованого методу на існуючими. Для запропонованого методу значення коефіцієнта рівномірності розподілу

навантаження в середньому на 50% вище ніж існуючого методу. Цей коефіцієнт відображає загальну ефективність та прогнозовану продуктивність системи, а його підвищення сприяє зниженню затрат на ресурси, оскільки система працює більш ефективно та стабільно. Запропоновані моделі та алгоритми також можуть бути застосовані при побудові систем з динамічним навантаженням та розподіленою архітектурою, де необхідно виконувати рівномірне балансування навантаження.

8. Розроблено алгоритм формування даних телекомунікаційної мережі для навчання нейронних мереж. На основі даного алгоритму виконано програмну реалізацію та отримано моделі для прогнозування пасажиропотоку та визначення ймовірності виникнення ДТП.

9. Виконано практичну реалізацію системи на основі запропонованої архітектури та методів. На основі розробленої архітектури системи виконано програмну реалізацію її компонентів, а також розроблено алгоритм процесу розгортання програмного забезпечення на серверах. Проведено аналіз та обґрунтовано відповідність розробленої системи основним стандартам захисту інформації. Для проведення оцінки ефективності розробленої системи, а також її порівняння з існуючими комерційними рішеннями запропоновано методику тестування. Суть даної методики полягає у відтворенні навантаження, яке створює транспортна система міста Києва, для двох систем – розробленої на основі запропонованої архітектури та методів, а також системи на основі існуючих комерційних рішень. Оцінка ефективності вищенаведених систем проводилась по ключових характеристиках робастності системи, які були виділенні в завданні даного дослідження.

Система розроблена на основі запропонованих методів, показала перевагу по вищенаведених характеристиках. А саме: рівномірність розподілу навантаження є вищою на 40-55%, а об’єм використаних обчислювальних ресурсів є меншим на 50-60%. Після оцінки технічних характеристик, проведено аналіз економічного ефекту від запропонованих методів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Overview. World Bank. (2023). https://www.worldbank.org/en/topic/urbandevelopment/overview.

2. Державна підтримка українського експорту. (2020). http://ukrexport.gov.ua/ukr/about_ukraine/population/ukr/86.html.

3. Xiaobo Yang. (n.d.). Advanced public transport system in Singapore. In Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems. 2003 IEEE International Conference on Intelligent Transportation Systems. IEEE. https://doi.org/10.1109/itsc.2003.1252765

4. McCullogh, E., Macpherson, A., Hagel, B., Giles, A., Fuselli, P., Pike, I., Torres, J., & Richmond, S. A. (2023). Road safety, health equity, and the built environment: perspectives of transport and injury prevention professionals in five Canadian municipalities. In BMC Public Health (Vol. 23, Issue 1). Springer Science and Business Media LLC. https://doi.org/10.1186/s12889-023-16115-7

5. Morri, N., Hadouaj, S., & Ben Said, L. (2021). An Approach to Intelligent Control Public Transportation System Using a Multi-agent System. In Enterprise Information Systems (pp. 242–267). Springer International Publishing. https://doi.org/10.1007/978-3-030-75418-1_12

6. Jin, S. T., Kong, H., & Sui, D. Z. (2019). Uber, Public Transit, and Urban Transportation Equity: A Case Study in New York City. In The Professional Geographer (Vol. 71, Issue 2, pp. 315–330). Informa UK Limited. https://doi.org/10.1080/00330124.2018.1531038

7. Wachs, M. (1993). Learning from Los Angeles: transport, urban form, and air quality. In Transportation (Vol. 20, Issue 4, pp. 329–354). Springer Science and Business Media LLC. https://doi.org/10.1007/bf01100463

8. Guzman, L., Oviedo, D., & Cardona, R. (2018). Accessibility Changes: Analysis of the Integrated Public Transport System of Bogotá. In Sustainability (Vol. 10, Issue 11, p. 3958). MDPI AG. https://doi.org/10.3390/su10113958

9. Rodriguez-Valencia, A., Rosas-Satizábal, D., & Hidalgo, D. (2023). Big effort, little gain for users: lessons from the public transport system reform in Bogotá. In Public Transport (Vol. 15, Issue 2, pp. 411–433). Springer Science and Business Media LLC. https://doi.org/10.1007/s12469-022-00308-1

10. Cabrera-Moya, D. R. R., & Prieto-Rodríguez, G. A. (2022). On the need for structuring of the Integrated Public Transport System -IPTS of Bogotá, Colombia as a Viable System. In Revista de Investigaciones Universidad del Quindío (Vol. 34, Issue 2, pp. 440–461). Universidad del Quindio. https://doi.org/10.33975/riuq.vol34n2.1070

11. Triviño, J. M., & García, W. C. (2021). General guidelines for the design of BRT routes in the Public Transport Integrated System of Bogotá. In Transportation Research Procedia (Vol. 58, pp. 622–629). Elsevier BV. https://doi.org/10.1016/j.trpro.2021.11.082

12. Aragón-Osejo, J. (2018). Development of a tool for determining speeds and road directions using GPS vehicle devices and free resources. Unpublished. https://doi.org/10.13140/RG.2.2.23205.96487

13. Passenger information system (PIS). AMiT Transportation. (2022, June 22). https://amit-transportation.com/en/reseni/passenger-information-system-pis/

14. Collins, K., & Muntean, G.-M. (2007). TraffCon: an intelligent traffic control system for wireless vehicular networks. In China-Ireland International Conference on Information and Communications Technologies (CIICT 2007). China-Ireland International Conference on Information and Communications Technologies (CIICT 2007). IEE. https://doi.org/10.1049/cp:20070766

15. Fare collection. Bus Rapid Transit Planning Guide. (2022). https://brtguide.itdp.org/branch/master/guide/technology/fare-collection

16. Siemens Mobility GmbH. (2020, November 17). Siemens Mobility. Connected vehicles system. Press Release. https://press.siemens.com/global/en/pressrelease/siemens-mobility-provides-connected-vehicles-system-austrian-highways

17. Cubic NextBus. TECHSEEN. (2017, March 7). https://techseen.com/2017/03/07/cubic-tony-gale-nextbus/

18. Vidyasagaran, S., Devi, S. R., Varma, A., Rajesh, A., & Charan, H. (2017). A low cost IoT based crowd management system for public transport. In 2017 International Conference on Inventive Computing and Informatics (ICICI). 2017 International Conference on Inventive Computing and Informatics (ICICI). IEEE. https://doi.org/10.1109/icici.2017.8365342

19. Singh, K., & Jindal, S. K. (2022). Design and implementation of IoT enabled ultraviolet C(UVC) irradiation-based sanitizer system for public transport with remote monitoring. In 2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE). 2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE). IEEE. https://doi.org/10.1109/icraie56454.2022.10054295

20. Arslan, S., & Kardas, G. (2021). The Need for Model-driven Engineering in the Development of IoT Software for Public Transportation Systems. In 2021 15th Turkish National Software Engineering Symposium (UYMS). 2021 15th Turkish National Software Engineering Symposium (UYMS). IEEE. https://doi.org/10.1109/uyms54260.2021.9659613

21. Palconit, M. G. B., & Nunez, W. A. (2018). Statistical analysis of CO<inf>2</inf> emission based on road grade, acceleration and vehicle specific power for public utility vehicles: An IoT application. In 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). IEEE. https://doi.org/10.1109/wf-iot.2018.8355235

22. Lohokare, J., Dani, R., Sontakke, S., & Adhao, R. (2017). Scalable tracking system for public buses using IoT technologies. In 2017 International Conference on Emerging Trends & Innovation in ICT (ICEI). 2017 International Conference on Emerging Trends & Innovation in ICT (ICEI). IEEE. https://doi.org/10.1109/etiict.2017.7977019

23. Rohit, M. H. (2020). An IoT based System for Public Transport Surveillance using real-time Data Analysis and Computer Vision. In 2020 Third

International Conference on Advances in Electronics, Computers and Communications (ICAECC). 2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC). IEEE. https://doi.org/10.1109/icaecc50550.2020.9339485

24. Bhatia, S., Gautam, D., Kumar, S., & Verma, S. (2023). Automatic Seat Identification System in Smart Transport using IoT and Image Processing. In 2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT). 2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT). IEEE. https://doi.org/10.1109/icct56969.2023.10075664

25. Cabrera, R. S., & de la Cruz, A. P. (2018). Public Transport Vehicle Tracking Service for Intermediate Cities of Developing Countries, based on ITS Architecture using Internet of Things (IoT). In 2018 21st International Conference on Intelligent Transportation Systems (ITSC). 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE. https://doi.org/10.1109/itsc.2018.8569906

26. Chavhan, S., Gupta, D., Chandana, B. N., Khanna, A., & Rodrigues, J. J. P. C. (2020). IoT-Based Context-Aware Intelligent Public Transport System in a Metropolitan Area. In IEEE Internet of Things Journal (Vol. 7, Issue 7, pp. 6023–6034). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/jiot.2019.2955102

27. Vieira, D. F., & Júnior, H. G. (2022). Public Transportation Passengers Accounting at University by IoT Device. In 2022 Symposium on Internet of Things (SIoT). 2022 Symposium on Internet of Things (SIoT). IEEE. https://doi.org/10.1109/siot56383.2022.10070138

28. Skarga-Bandurova, I., Derkach, M., & Kotsiuba, I. (2018). The Information Service for Delivering Arrival Public Transport Prediction. In 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS). IEEE. https://doi.org/10.1109/idaacs-sws.2018.8525787

29. Enciso-Quispe, L., Barba-Guaman, L., Sanchez, J., & Quezada-Sarmiento, P. A. (2018). Simulation of people counter for public service buses of Loja with IoT concept applying the Viola-Jones algorithm. In 2018 13th Iberian Conference on Information Systems and Technologies (CISTI). 2018 13th Iberian Conference on Information Systems and Technologies (CISTI). IEEE. https://doi.org/10.23919/cisti.2018.8399322

30. Geetha, S., & Cicilia, D. (2017). IoT enabled intelligent bus transportation system. In 2017 2nd International Conference on Communication and Electronics Systems (ICCES). 2017 2nd International Conference on Communication and Electronics Systems (ICCES). IEEE. https://doi.org/10.1109/cesys.2017.8321235

31. James, J. G., & Nair, S. (2017). Efficient, real-time tracking of public transport, using LoRaWAN and RF transceivers. In TENCON 2017 - 2017 IEEE Region 10 Conference. TENCON 2017 - 2017 IEEE Region 10 Conference. IEEE. https://doi.org/10.1109/tencon.2017.8228237

32. Masram, B., Nimje, A., Raut, A., Mehatre, N., & Humane, S. (2023). IoT based Overload Detection System in Public Transportation Vehicles. In 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI). 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI). IEEE. https://doi.org/10.1109/icoei56765.2023.10125693

33. Lushi, A., Daas, D., & Nadeem, M. (2022). IoT-Based Public Transport Management System. In 2022 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT). 2022 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT). IEEE. https://doi.org/10.1109/gcaiot57150.2022.10019029

34. Kadam, A. J., Patil, V., Kaith, K., Patil, D., & Sham. (2018). Developing a Smart Bus for Smart City using IOT Technology. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE. https://doi.org/10.1109/iceca.2018.8474819

35. Puiu, D., Bischof, S., Serbanescu, B., Nechifor, S., Parreira, J., & Schreiner, H. (2017). A public transportation journey planner enabled by IoT data analytics. In 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN). 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN). IEEE. https://doi.org/10.1109/icin.2017.7899440

36. Hattarge, S., Kekre, A., & Kothari, A. (2018). LoRaWAN based GPS tracking of city-buses for smart public transport system. In 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC). 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC). IEEE. https://doi.org/10.1109/icsccc.2018.8703356

37. Sundar, G. V., & Rajagopal, B. G. (2017). IoT based passenger information system optimized for Indian metros. In 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA). 2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA). IEEE. https://doi.org/10.1109/iceca.2017.8203650

38. Gill, K. S., Sharma, A., Anand, V., & Gupta, S. (2023). Automated Fare Collection System for Public Transport using Intelligent IoT based System. In 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF). 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF). IEEE. https://doi.org/10.1109/iceconf57129.2023.10083627

39. Sohaib, M. d., Aarthi, V. P. M. B., Laxmi, M. N., Shivaji, G. G., Kumar, T. A., & Ramesh, M. (2023). IoT-based Web Application for Passenger Travel Tracking System. In 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI). 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI). IEEE. https://doi.org/10.1109/icoei56765.2023.10125610

40. Pawar, V., & Bhosale, N. P. (2018). Internet-of-Things Based Smart Local Bus Transport Management System. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). 2018 Second

International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE. https://doi.org/10.1109/iceca.2018.8474728

41. Scott, T. L., & Eleyan, A. (2019). CoAP based IoT data transfer from a Raspberry Pi to Cloud. In 2019 International Symposium on Networks, Computers and Communications (ISNCC). 2019 International Symposium on Networks, Computers and Communications (ISNCC). IEEE. https://doi.org/10.1109/isncc.2019.8909150

42. Al-Jabi, M. (2017). Toward an IoT-enabled adaptive interactive bus transportation system. In 2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS). 2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS). IEEE. https://doi.org/10.1109/it-dreps.2017.8277802

43. Gunady, S., & Keoh, S. L. (2019). A Non-GPS based Location Tracking of Public Buses using Bluetooth Proximity Beacons. In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). 2019 IEEE 5th World Forum on Internet of Things (WF-IoT'19). IEEE. https://doi.org/10.1109/wf-iot.2019.8767198

44. Pokric, B., Krco, S., & Pokric, M. (2014). Augmented Reality Based Smart City Services Using Secure IoT Infrastructure. In 2014 28th International Conference on Advanced Information Networking and Applications Workshops. 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA). IEEE. https://doi.org/10.1109/waina.2014.127

45. Jain, M., & Kulkarni, P. (2022). Application of AI, IOT and ML for Business Transformation of The Automotive Sector. In 2022 International Conference on Decision Aid Sciences and Applications (DASA). 2022 International Conference on Decision Aid Sciences and Applications (DASA). IEEE. https://doi.org/10.1109/dasa54658.2022.9765294

46. Sulo, I., Keskin, S. R., Dogan, G., & Brown, T. (2019). Energy Efficient Smart Buildings: LSTM Neural Networks for Time Series Prediction. In 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML). 2019 International Conference on Deep Learning and

Machine Learning in Emerging Applications (Deep-ML). IEEE. https://doi.org/10.1109/deep-ml.2019.00012

47. Jafari, F., Ponnambalam, K., Mousavi, J., & Karray, F. (2020). Yield forecast of California strawberry: Time-series Models vs. ML Tools. In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE. https://doi.org/10.1109/smc42975.2020.9283138

48. Nilesh, N., Narang, J., Parmar, A., & Chaudhari, S. (2022). IoT and ML-based AQI Estimation using Real-time Traffic Data. In 2022 IEEE 8th World Forum on Internet of Things (WF-IoT). 2022 IEEE 8th World Forum on Internet of Things (WF-IoT). IEEE. https://doi.org/10.1109/wf-iot54382.2022.10152160

49. Subudhi, M., Charan Bhuyan, K., & Dastidar, A. (2022). IoT Assisted Farming using ML techniques. In 2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC). 2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC). IEEE. https://doi.org/10.1109/isssc56467.2022.10051428

50. Ksentini, A., Jebalia, M., & Tabbane, S. (2020). Fog-enabled Industrial IoT Network Slicing model based on ML-enabled Multi-objective Optimization. In 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). IEEE. https://doi.org/10.1109/wetice49692.2020.00042

51. Naveen Ananda Kumar, J., & Suresh, S. (2019). A Proposal of smart hospital management using hybrid Cloud, IoT, ML, and AI. In 2019 International Conference on Communication and Electronics Systems (ICCES). 2019 International Conference on Communication and Electronics Systems (ICCES). IEEE. https://doi.org/10.1109/icces45898.2019.9002098

52. Uddin, G. (2021). Security and Machine Learning Adoption in IoT: A Preliminary Study of IoT Developer Discussions. In 2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT).

2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT). IEEE. https://doi.org/10.1109/serp4iot52556.2021.00013

53. Gundala, J. R., Varsha Potluri, S. S., Damle, S. V., & Hashmi, M. F. (2022). IoT & ML-Based Healthcare Monitoring System-Review. In 2022 IEEE International Symposium on Smart Electronic Systems (iSES). 2022 IEEE International Symposium on Smart Electronic Systems (iSES). IEEE. https://doi.org/10.1109/ises54909.2022.00137

54. Syed, F. K., Paul, A., Kumar, A., & Cherukuri, J. (2019). Low-cost IoT+ML design for smart farming with multiple applications. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE. https://doi.org/10.1109/icccnt45670.2019.8944791

55. Mchergui, A., Hajlaoui, R., Moulahi, T., Alabdulatif, A., & Lorenz, P. (2023). Steam computing paradigm: Cross-layer solutions over cloud, fog, and edge computing. In IET Wireless Sensor Systems. Institution of Engineering and Technology (IET). https://doi.org/10.1049/wss2.12051

56. Porru, S., Misso, F. E., Pani, F. E., & Repetto, C. (2020). Smart mobility and public transport: Opportunities and challenges in rural and urban areas. In Journal of Traffic and Transportation Engineering (English Edition) (Vol. 7, Issue 1, pp. 88–97). Elsevier BV. https://doi.org/10.1016/j.jtte.2019.10.002

57. Farkas, K., Feher, G., Benczur, A., & Sidlo, C. (2015). Crowdsending based public transport information service in smart cities. In IEEE Communications Magazine (Vol. 53, Issue 8, pp. 158–165). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/mcom.2015.7180523

58. Vieira, E., Almeida, J., Ferreira, J., Dias, T., Vieira Silva, A., & Moura, L. (2023). A Roadside and Cloud-Based Vehicular Communications Framework for the Provision of C-ITS Services. In Information (Vol. 14, Issue 3, p. 153). MDPI AG. https://doi.org/10.3390/info14030153

59. Metzger, F., Hobfeld, T., Bauer, A., Kounev, S., & Heegaard, P. E. (2019). Modeling of Aggregated IoT Traffic and Its Application to an IoT Cloud. In Proceedings of the IEEE (Vol. 107, Issue 4, pp. 679–694). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/jproc.2019.2901578

60. Khan, M. A., Nawaz, T., Khan, U. S., Hamza, A., & Rashid, N. (2023). IoT-Based Non-Intrusive Automated Driver Drowsiness Monitoring Framework for Logistics and Public Transport Applications to Enhance Road Safety. In IEEE Access (Vol. 11, pp. 14385–14397). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2023.3244008

61. Hind, M., Noura, O., Sanae, M., & Abraham, A. (2023). A Comparative Study for Modeling IoT Security Systems. In Intelligent Systems Design and Applications (pp. 258–269). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-35510-3_25

62. Ahmad, W., Rasool, A., Javed, A. R., Baker, T., & Jalil, Z. (2021). Cyber Security in IoT-Based Cloud Computing: A Comprehensive Survey. In Electronics (Vol. 11, Issue 1, p. 16). MDPI AG. https://doi.org/10.3390/electronics11010016

63. Siwakoti, Y. R., Bhurtel, M., Rawat, D. B., Oest, A., & Johnson, R. C. (2023). Advances in IoT Security: Vulnerabilities, Enabled Criminal Services, Attacks, and Countermeasures. In IEEE Internet of Things Journal (Vol. 10, Issue 13, pp. 11224–11239). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/jiot.2023.3252594

64. Zakutynskyi, I., Sibruk, L., & Kokarieva, A. (2023). IoT System for Monitoring and Managing Public Transport Data. In WSEAS TRANSACTIONS ON SYSTEMS (Vol. 22, pp. 242–248). World Scientific and Engineering Academy and Society (WSEAS). https://doi.org/10.37394/23202.2023.22.25

65. Liaqat, M., Naveed, A., Ali, R. L., Shuja, J., & Ko, K.-M. (2019). Characterizing Dynamic Load Balancing in Cloud Environments Using Virtual Machine Deployment Models. In IEEE Access (Vol. 7, pp. 145767–145776). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2019.2945499

66. Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. In IEEE Access (Vol. 9, pp. 41731–41744). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2021.3065308

67. Goncalves, D., Puliafito, C., Mingozzi, E., Rana, O., Bittencourt, L., & Madeira, E. (2020). Dynamic Network Slicing in Fog Computing for Mobile Users in MobFogSim. In 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC). 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC). IEEE. https://doi.org/10.1109/ucc48980.2020.00042

68. Yuan, H., Bi, J., & Zhou, M. (2022). Geography-Aware Task Scheduling for Profit Maximization in Distributed Green Data Centers. In IEEE Transactions on Cloud Computing (Vol. 10, Issue 3, pp. 1864–1874). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/tcc.2020.3001051

69. Bogdanov, K. L., Reda, W., Maguire, G. Q., Jr., Kostić, D., & Canini, M. (2018). Fast and Accurate Load Balancing for Geo-Distributed Storage Systems. In Proceedings of the ACM Symposium on Cloud Computing. SoCC '18: ACM Symposium on Cloud Computing. ACM. https://doi.org/10.1145/3267809.3267820

70. Srinivas, J., Qyser, A. A. M., & Reddy, B. E. (2015). Exploiting Geo Distributed datacenters of a cloud for load balancing. In 2015 IEEE International Advance Computing Conference (IACC). 2015 IEEE International Advance Computing Conference (IACC). IEEE. https://doi.org/10.1109/iadcc.2015.7154780

71. Shuaib, M., Bhatia, S., Alam, S., Masih, R. K., Alqahtani, N., Basheer, S., & Alam, M. S. (2023). An Optimized, Dynamic, and Efficient Load-Balancing Framework for Resource Management in the Internet of Things (IoT) Environment. In Electronics (Vol. 12, Issue 5, p. 1104). MDPI AG. https://doi.org/10.3390/electronics12051104

72. Lim, J. (2021). Scalable Fog Computing Orchestration for Reliable Cloud Task Scheduling. In Applied Sciences (Vol. 11, Issue 22, p. 10996). MDPI AG. https://doi.org/10.3390/app112210996

73. Singh, S. P., Kumar, R., Sharma, A., & Nayyar, A. (2020). Leveraging energy-efficient load balancing algorithms in fog computing. In Concurrency and Computation: Practice and Experience (Vol. 34, Issue 13). Wiley. https://doi.org/10.1002/cpe.5913

74. Fan, Q., & Ansari, N. (2020). Towards Workload Balancing in Fog Computing Empowered IoT. In IEEE Transactions on Network Science and Engineering (Vol. 7, Issue 1, pp. 253–262). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/tnse.2018.2852762

75. Kim, H.-Y., & Kim, J.-M. (2016). A load balancing scheme based on deep-learning in IoT. In Cluster Computing (Vol. 20, Issue 1, pp. 873–878). Springer Science and Business Media LLC. https://doi.org/10.1007/s10586-016-0667-5

76. Gomez, C., Shami, A., & Wang, X. (2018). Machine Learning Aided Scheme for Load Balancing in Dense IoT Networks. In Sensors (Vol. 18, Issue 11, p. 3779). MDPI AG. https://doi.org/10.3390/s18113779

77. Adil, M. (2021). Congestion free opportunistic multipath routing load balancing scheme for Internet of Things (IoT). In Computer Networks (Vol. 184, p. 107707). Elsevier BV. https://doi.org/10.1016/j.comnet.2020.107707

78. Tonguz, O. K., & Yanmaz, E. (2008). The Mathematical Theory of Dynamic Load Balancing in Cellular Networks. In IEEE Transactions on Mobile Computing (Vol. 7, Issue 12, pp. 1504–1518). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/tmc.2008.66

79. Latchoumi, T. P., & Parthiban, L. (2021). Quasi Oppositional Dragonfly Algorithm for Load Balancing in Cloud Computing Environment. In Wireless Personal Communications (Vol. 122, Issue 3, pp. 2639–2656). Springer Science and Business Media LLC. https://doi.org/10.1007/s11277-021-09022-w

80. Zakutynskyi, I. (2023). Finding the Optimal Number of Computing Containers in IoT Systems: Application of Mathematical Modeling Methods. In Electronics and Control Systems (Vol. 2, Issue 76, pp. 9–14). National Aviation University. https://doi.org/10.18372/1990-5548.76.17661

81. Alakbarov, R. (2022). An Optimization Model for Task Scheduling in Mobile Cloud Computing. In International Journal of Cloud Applications and Computing (Vol. 12, Issue 1, pp. 1–17). IGI Global. https://doi.org/10.4018/ijcac.297102

82. Kaveri, P. R., & Chavan, V. (2013). Mathematical model for higher utilization of database resources in cloud computing. In 2013 Nirma University International Conference on Engineering (NUiCONE). 2013 Nirma University International Conference on Engineering (NUiCONE). IEEE. https://doi.org/10.1109/nuicone.2013.6780095

83. Díaz, C., Leitão, C., Marques, C., Alberto, N., Domingues, M., Ribeiro, T., Pontes, M., Frizera, A., Antunes, P., André, P., & Ribeiro, M. (2019). IoToF: A Long-Reach Fully Passive Low-Rate Upstream PHY for IoT over Fiber. In Electronics (Vol. 8, Issue 3, p. 359). MDPI AG. https://doi.org/10.3390/electronics8030359

84. Dahlman, E., Parkvall, S., & Sköld, J. (2016). Introduction. In 4g, LTE Evolution and the Road to 5G (pp. 1–5). Elsevier. https://doi.org/10.1016/b978-0-12-804575-6.00001-7

85. Dahlman, E., Parkvall, S., & Sköld, J. (2018). What Is 5G? In 5G NR: the Next Generation Wireless Access Technology (pp. 1–6). Elsevier. https://doi.org/10.1016/b978-0-12-814323-0.00001-6

86. ElNashar, A., & El-saidny, M. (Eds.). (2018). Practical Guide to LTE-A, VoLTE and IoT. Wiley. https://doi.org/10.1002/9781119063407

87. Fattah, H. (2018). 5G LTE Narrowband Internet of Things (NB-IoT). CRC Press. https://doi.org/10.1201/9780429455056

88. Soldani, D., Shore, M., Mitchell, J., & Gregory, M. A. (2018). The 4G to 5G Network Architecture Evolution in Australia. In Journal of Telecommunications and the Digital Economy (Vol. 6, Issue 4, pp. 1–30). Telecommunications Association Inc. https://doi.org/10.18080/jtde.v6n4.161

89. IOT coverage. Narrowband state. (2023). https://www.narrowband.com/nb-iot-coverage

90. Shop-Gsm. (2020, June 2). Що таке мережа NB-IOT і які її особливості. GSM Ua. https://shop-gsm.ua/blog/chto-takoe-set-nb-iot-i-kakovy-ee-osobennosti/

91. Al-Sarawi, S., Anbar, M., Alieyan, K., & Alzubaidi, M. (2017). Internet of Things (IoT) communication protocols: Review. In 2017 8th International Conference on Information Technology (ICIT). 2017 8th International Conference on Information Technology (ICIT). IEEE. https://doi.org/10.1109/icitech.2017.8079928

92. Swamy, S. N., Jadhav, D., & Kulkarni, N. (2017). Security threats in the application layer in IOT applications. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). IEEE. https://doi.org/10.1109/i-smac.2017.8058395

93. Ebrahimpour, H., Ashtiani, M., Bakhshi, F., & Bakhtiariazad, G. (2022). A heuristic-based package-aware function scheduling approach for creating a trade-off between cold-start time and cost in FaaS computing environments. Research Square Platform LLC. https://doi.org/10.21203/rs.3.rs-1725753/v2

94. Raschbichler, F. (2023, June 29). Mqtt 5: How the new shared subscriptions feature works. MQTT Shared Subscriptions – MQTT 5 Essentials Part 7. https://www.hivemq.com/blog/mqtt5-essentials-part7-shared-subscriptions/

95. MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

96. ReLU. (2023). Вікіпедія. https://uk.wikipedia.org/w/index.php?title=ReLU

97. Public Barcelona government data portal. (2022). Accidents by driver cause managed by the Guàrdia - open data BCN. OpenDataBCN . https://opendata-ajuntament.barcelona.cat/data/en/dataset/accidents_causa_conductor_gu_bcn

98. Wikipedia contributors. (2023, August 9). Mean absolute percentage error. In Wikipedia, The Free Encyclopedia.

https://en.wikipedia.org/w/index.php?title=Mean_absolute_percentage_error&oldid=1169528649

99. Wikipedia contributors. (2023, September 21). JSON Web Token. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=JSON_Web_Token

100. Amazon. "Data modeling". Available at: https://docs.aws.amazon.com/timestream/latest/developerguide/data-modeling.html

101. Wikipedia contributors. (2023, August 27). Test-driven development. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Test-driven_development

102. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1603.04467

103. Ketkar, N. (2017). Introduction to Keras. In Deep Learning with Python (pp. 97–111). Apress. https://doi.org/10.1007/978-1-4842-2766-4_7

104. Bernstein, D. (2014). Containers and Cloud: From LXC to Docker to Kubernetes. In IEEE Cloud Computing (Vol. 1, Issue 3, pp. 81–84). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/mcc.2014.51

105. Wikipedia contributors. (2023, September 2). CI/CD. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=CI/CD

106. Wikipedia. "Kyivpastrans". Avalaible at: https://en.wikipedia.org/wiki/Kyivpastrans

107. AWS Pricing (2022). Amazon. https://aws.amazon.com/pricing

108. Economics 2: EC2 (2023). Amazon. https://aws.amazon.com/ec2/pricing/on-demand/

ДОДАТКИ

Акт впровадження результатів дисертаційного дослідження в систему громадського транспорту м. Києва Департаментом транспортної інфраструктури Київської міської державної адміністрації.	1 сторінка
Акт впровадження результатів дисертаційного дослідження в проектах компанії ТОВ «Уатроніка».	1 сторінка
Акт впровадження результатів дисертаційного дослідження в проектах та процесах компанії LLC «FOUR AGES», Велика Британія.	1 сторінка
Акт впровадження результатів дисертаційного дослідження в проектах та процесах компанії LLC «Sirin Software», США.	1 сторінка
Акт впровадження результатів дисертаційного дослідження в проектах та процесах компанії LLC «E-SYNERGO», Естонія.	1 сторінка
Програмна реалізація MQTT брокера	1 сторінка
Програмна реалізація MQTT клієнта	1 сторінка
Програмна реалізація функції визначення коефіцієнту розподілу навантаження	1 сторінка
Програмна реалізація емулятора IoT модуля	1 сторінка
Програмна реалізація модуля моніторингу	1 сторінка
Програмна реалізація MQTT підписника	1 сторінка
Програмна реалізація алгоритму (рис. 3.9)	1 сторінка
Список публікацій за темою дисертації	2 сторінки

УКРАЇНА

ВИКОНАВЧИЙ ОРГАН КИЇВСЬКОЇ МІСЬКОЇ РАДИ
(КИЇВСЬКА МІСЬКА ДЕРЖАВНА АДМІНІСТРАЦІЯ)

ДЕПАРТАМЕНТ ТРАНСПОРТНОЇ ІНФРАСТРУКТУРИ

вул. Леонтовича, 6, м. Київ, 01030 тел.(044) 366 63 05, (044) 366 63 43
Контактний центр міста Києва (044) 15 51 E-mail: transport@kyivcity.gov.ua
Код ЄДРПОУ 37405284

26.06.2023 № 053-7017

«ЗАТВЕРДЖУЮ»
Заступник директора - начальник управління дорожньої інфраструктури Департаменту транспортної інфраструктури, к.т.н.
_____ Валентин КУЛЬБАКО
«26» 06 2023 р.

АКТ
реалізації результатів дисертаційної роботи
Закутинського Ігоря Володимировича
на тему: «Система інтернету речей для моніторингу та управління громадським транспортом»

Даний акт складено про те, що в результаті проведених досліджень Закутинським Ігорем Володимировичем у дисертаційній роботі “Система інтернету речей для моніторингу та управління громадським транспортом” були розроблені методи для побудови інтелектуальних систем громадського транспорту на основі концепції Інтернету речей, а також запропоновано загальну структуру та архітектуру системи.

Результати дисертаційного дослідження:

- Містять практичну цінність та наукову новизну;
- Передано у використання Департаменту транспортної інфраструктури КМДА для подальшого впровадження (модернізації) системи громадського транспорту м. Києва.
- Запропонована концепція Інтелектуальної системи громадського транспорту відповідає та сприяє розвитку Національної транспортної стратегії України на період до 2030 року.

Начальник відділу пасажирських перевезень управління транспорту та міської мобільності Департаменту транспортної інфраструктури 26.06.23 Антон ЄРОФЄЄНКОВ

20 Червня 2023

Акт впровадження

Даний документ підтверджує, що результати отримані Закутинським Ігорем Володимировичем в ході дисертаційного дослідження "Система Інтернету речей для моніторингу та управління громадським транспортом" містять значну практичну цінність та використовуються в діяльності компанії "Уатроніка".

Практично вагомим є такі результати:

1. Алгоритм автоматичного розгортання розподіленої архітектури мережі Інтернету речей.

2. Експериментальні дослідження застосування NB-IoT стандарту для побудови телекомунікаційної мережі транспортних систем.

3. Метод балансування навантаження в розподілених системах Інтернету речей на основі багатокритеріального моніторингу стану обчислювальних контейнерів.

4. Метод визначення оптимальної кількості обчислювальних ресурсів відповідно до кількості активних пристроїв Інтернету речей.

Впровадження вищенаведених результатів дозволило покращити ефективність розробки інтелектуальних систем на основі концепції Інтернету речей, а також покращити характеристики вже наявних систем.

к.т.н. Богдан Бойко
Генеральний директор
ТОВ "Уатроніка"

+38(044)-229-46-89
www.uatronica.com
contact@uatronica.com
Druzhkivska Street, 10,
Kyiv, 03113

FOUR AGES
SOFTWARE

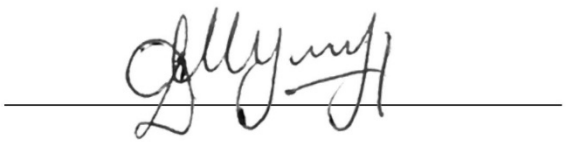
IMPLEMENTATION PAPER

In this letter, we acknowledge that the results obtained by Ihor Zakutynskyi in the Ph.D. dissertation research are performed at a high scientific level and contain practical value that will be used in the «Four Ages Software» company activities.

The outcomes achieved through the extensive research conducted in the dissertation «IoT System for Monitoring and Management Public Transport» hold significant importance due to their inherent innovativeness and potential impact on the IoT industry. The innovative nature of the results lies in their ability to offer fresh perspectives and novel solutions to existing challenges.

In our opinion, **improved methods of load balancing**, as well as **mathematical models for Internet of Things systems simulation**, are especially valuable. Also, research related to the use of **Neural Networks for passenger traffic forecasting** is valuable and can be used for a broader task - time series forecasting. By adopting a comprehensive approach and leveraging cutting-edge methodologies, this research has unveiled insights that were previously unexplored.

These insights not only advance the theoretical understanding of the subject matter but also open up avenues for practical application within the company's projects.



DMYTRO SHULHA
CHIEF EXECUTIVE OFFICER

124 City Road,
London, United Kingdom,
EC1V 2NX

contact@four-ages.com

https://four-ages.com

SIRIN SOFTWARE

Phone: +1 321 328 8379
Email: info@sirinsoftware.com
13920 Landstar Blvd 101-0060,
Orlando, FL 32824, US
https://sirinsoftware.com

Implementation act

This document certifies that the results obtained in the Ph.D. dissertation thesis «IoT System for Monitoring and Management Public Transport» by Ihor Zakutynskyi were performed at a high technical level and hold substantial practical significance.

Sirin Software is a Florida-based software/hardware engineering company. We are a reliable partner of such companies as Apple, Microsoft, Cypress, and Infineon. At Sirin Software we build a network of connected sensors, embedded software, cloud data processing pipelines, and other technologies that allow you to collect, connect, and analyze data from devices and systems via the Internet. The integration of Internet of Things (IoT) technologies into the monitoring and management of public transport has been an increasing trend in the last decade. Therefore, the topic of this study is relevant and important today. The improved load-balancing methods proposed in this study have a high potential and will help to improve the fault-tolerance of IoT systems. Also, modern IoT systems generate large volumes of data, therefore, the methods of data processing based on neural networks proposed in this work, are particularly important. The insights received from this dissertation possess a direct relevance to real-world challenges faced by our company and the industry overall.

Alex Nikitenko
Chief executive officer

(Signature)

25.08.2023

E-SYNERGO OU

E-SYNERGO

Estonia,
Kakumae tee 226, Tallinna linn, 13516
Phone: +3 (725) 494 -18 - 62
info@e-synergo.eu

Implementation act

This document confirmed the use of the results obtained by Ihor Zakutynskyi during his Ph.D. thesis «IoT system for monitoring and management Public Transport» in the projects of the company E-Synergo.

In the course of the work, the author developed a complex of innovative methods for the optimal construction of Intelligent transport systems based on the concept of the Internet of Things.

The following studies are particularly important:

1. Research of microservice communication for IoT-based Systems.
2. Methods for building a dynamic IoT-based architecture.
3. Mathematical models for optimizing load distribution in the computing containers.

The above research was used to improve the company's projects and allowed to optimize IoT systems resource utilization and make the architecture more resistant to load changes. The proposed methods are incredibly relevant today and significantly contribute to the development of Internet of Things technologies.

EESTI TALLINN OÜ E-SYNERGO REGISTRIKOOD 12504763

Managing Director
Andrey Sherstyuk
02/07/2023

(Signature)

Програмна реалізація MQTT брокера

```
const parser = mqtt.parser({
  protocolVersion: 4
});

const mqttServer = createServer((connection) => {
  connection.on('end', () => {
    console.log('MQTT server: client disconnected');
  });

  connection.on('data', (data) => {
    parser.once('packet', packet => {
      if (packet.cmd === 'connect') {
        console.log("MQTT server: new client connected: ",
            packet.clientId);
        const workerIndex = workers.findIndex(item => item.workerId
            === packet.clientId);
        if (workerIndex === -1) workers.push({ workerId:
            packet.clientId, connection })
      }

      if (packet.cmd === 'publish') {

        /** Different balancing method */
        // const worker = minAvgCpuUsageWorker();
        const worker = roundRobinWorker();

        const task = JSON.parse(packet.payload.toString());
        console.log({ worker: worker.workerId, taskId: task.taskId,
            time: Date.now() });
        worker.connection.write(data);
      }

    });
    parser.parse(data);
  });
});

mqttServer.on('error', (err) => {
  console.log(`MQTT server: `, err);
  throw err;
});

mqttServer.listen(process.env.MQTT_PORT, () => {
  console.log(`MQTT server: start at ${process.env.MQTT_PORT}`);
});
```

Програмна реалізація MQTT клієнта

```
const net = require("net");
const mqtt = require("mqtt-packet");
const { readFileSync } = require('fs');
require('dotenv').config()

const taskFile = readFileSync('./task_v2.json').toString();
const tasks = JSON.parse(taskFile);

const delay = (timeMs) => new Promise((resolve, _) => setTimeout(() =>
    resolve(true), timeMs));

const socket = net.Socket();
const MODULE = 'MQTT Publisher';

const MQTT_BROKER_HOST = process.env.BROKER_HOST;
const MQTT_BROKER_PORT = process.env.MQTT_PORT;

socket.on("close", () => {
    console.log(MODULE, "connection closed");
});

socket.on("connect", async () => {
    console.log(MODULE, `connected to tcp://${MQTT_BROKER_HOST}/${
        MQTT_BROKER_PORT}`);

    for (let i=0; i<tasks.length; i++) {
        const task = tasks[i];
        let packet = mqtt.generate({
            cmd: 'publish',
            messageId: task.taskId,
            topic: 'compute',
            payload: JSON.stringify({  taskId: task.taskId,
                complexity: task.complexity }),
            retain: false
        });
        socket.write(packet);
        console.log("Publish task: ", task.taskId)
        await delay(1500);
    }
});

socket.connect(MQTT_BROKER_PORT, MQTT_BROKER_HOST);
```

Програмна реалізація функції визначення коефіцієнту розподілу навантаження

```
const calculateLoadDistribution = (workers) => {
    const dataset = this.buildDatasetFromWorkersStructure(workers);

    const result = [];
    for (let i=0; i<dataset.length; i++) {
      const instantValue = dataset[i];
      const distances = [];

      const calculateDistance = (vector1, vector2) => {
        const res = Math.sqrt(vector1.reduce((prev, current, index) =>
            {
          return prev += Math.pow((current - vector2[index]), 2)
        }, 0))
        return res;
      };

      const getAverageDistance = (distances, workersCount) =>
          distances.reduce((prev, current) => prev += current, 0) / (
          workersCount * (workersCount-1))

      for (let i=0; i<instantValue.length; i++) {
        for (let j=i+1; j<instantValue.length; j++) {
          distances.push(calculateDistance(instantValue[i],
              instantValue[j]));
        }
      }

      result.push(getAverageDistance(distances, workers.length));
    };

    return result;
}

export default calculateLoadDistribution;
```

Програмна реалізація емулятора IoT модуля

```
const run = async () => {
    const connection = build_connection({
      endpoint,
      key,
      cert,
      ca_file,
      client_id
    });

    try {
        await connection.connect()
        console.log("Connection completed. Process");
    } catch (err) {
        console.log(err);
    }

    const geolocation = new GeolocationModule();
    const fuelModule = new FuelModule();
    const climateModule = new ClimateModule();
    const passengersModule = new PassengersModule();

    const interval = setInterval(() => {
        const currentPosition = geolocation.getCurrentPosition();
        const fuelLevel = fuelModule.getFuelLevel();
        const temperature = climateModule.getCurrentTemperature();
        const humidity = climateModule.getCurrentHumidity();
        const passengers = passengersModule.getPassengersCount();

        const data = {
            payload: {
              deviceid,
              timestamp: Math.ceil(Date.now() / 1000),
              location: {
                lat: parseFloat(currentPosition.split(',')[0]),
                long: parseFloat(currentPosition.split(',')[1])
              },
              fuelLevel,
              temperature,
              humidity,
              passengers
            }
        }

        for (let i = 0; i<PROCESSES_COUNT; i++) {
            connection.publish(topic, JSON.stringify(data),
                mqtt.QoS.AtLeastOnce).then((value) => {});
        }
    }, 1000)
}
```

Програмна реалізація модуля моніторингу

```
const loadMonitoringServer = createServer((connection) => {
  connection.on('end', () => {
    console.log('Load monitoring server: client disconnected');
  });

  connection.on('data', (data) => {

    try {
      const measure = JSON.parse(data.toString());
      const workerIndex = workers.findIndex(item => item.workerId ===
          measure.workerId);

      if (workerIndex !== -1) {
        workers[workerIndex].avgCpuUsage = measure.avgCpuUsage;
      }

    } catch(err) {}
  });
});

loadMonitoringServer.on('error', (err) => {
  console.log(`Load monitoring server: `, err);
  throw err;
});


loadMonitoringServer.listen(process.env.LOAD_MONITORING_PORT);
```

Програмна реалізація MQTT підписника

```
const mqttSocket = net.Socket();
const parser = mqtt.parser({
    protocolVersion: 4
});

mqttSocket.on("close", (...props) => {
    console.log(`MQTT channel => connection closed: ${props}`);
});

mqttSocket.on("connect", () => {
    console.log("MQTT channel => connected. Worker: ", WORKER_ID);

    let mqttConnectPacket = mqtt.generate(connection);

    mqttSocket.write(mqttConnectPacket);
});

mqttSocket.on("data", (data) => {
    parser.once('packet', packet => {
        try {
            const task = JSON.parse(packet.payload.toString());
            if (task.complexity) {
                highUsageTask(task);
            }
        } catch (err) {
            console.log(err);
        }
    });
    parser.parse(data);
});

mqttSocket.connect(process.env.MQTT_PORT, process.env.BROKER_HOST);
```

Програмна реалізація алгоритму (рис. 3.9)

```
from scipy.optimize import minimize

def utility(X):
    return sum(np.log(1 + a_m * X) for a_m in [a_1, a_2, a_3])

def capacity_constraint(X):
    return N - (f_1_inv(X[0]) + f_2_inv(X[1]) + f_3_inv(X[2]))

initial_guess = [10, 10, 10]  # Initial guess for X values
constraints = {'type': 'ineq', 'fun': capacity_constraint}

result = minimize(lambda X: -utility(X), initial_guess, constraints=
    constraints)
optimal_X = result.x
optimal_utility = -result.fun

print("Optimal Overall Throughput:", optimal_X)
print("Optimal Utility:", optimal_utility)
```

Список публікацій здобувача за темою дисертації

1. Sibruk, L., & Zakutynskyi, I. (2022). Recurrent Neural Networks for Time Series Forecasting. Choosing the best Architecture for Passenger Traffic Data. In Electronics and Control Systems (Vol. 2, Issue 72, pp. 38–44). National Aviation University.

2. Zakutynskyi, I., & Rabodzei, I. (2022). Microservice Communication for IoT-based Systems. Architecture Review and Performance Test. In Electronics and Control Systems (Vol. 4, Issue 74, pp. 73–78). National Aviation University.

3. Zakutynskyi, I., Sibruk, L., & Kokarieva, A. (2023). IoT System for Monitoring and Managing Public Transport Data. In WSEAS TRANSACTIONS ON SYSTEMS (Vol. 22, pp. 242–248). World Scientific and Engineering Academy and Society (WSEAS).

4. Zakutynskyi, I. (2023). Finding the Optimal Number of Computing Containers in IoT Systems: Application of Mathematical Modeling Methods. In Electronics and Control Systems (Vol. 2, Issue 76, pp. 9–14). National Aviation University.

5. Закутинський, І. (2022). ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ПЕРЕДБАЧЕННЯ ТА АНАЛІЗУ ДОРОЖНЬО-ТРАНСПОРТНИХ ПРИГОД. In Наука і техніка сьогодні (Issue 13(13)). Ukrainian Assembly of Doctors of Science in Public Administration

6. Zakutynskyi, I., & Rabodzei, I. (2023). IoT system architecture for monitoring and analyzing public transport data. Multidisciplinary Science Journal, 5, 2023.

7. Zakutynskyi, I., & Sibruk L, Rabodzei, I. (2023). Performance evaluation of the cloud computing application for IoT-based public transport systems. Eastern-European Journal of Enterprise Technologies, 4, pp. 6-13.

8. Zakutynskyi, I., Rabodzei, I., Burmakin, S., Kalishuk, O., Nebylytsia, V. (2023).Improving a procedure of load balancing in distributed IoT systems. Eastern-European Journal of Enterprise Technologies, 5 (2 (125)).

9. Zakutynskyi, I., Sibruk, L. (2022). Aviation in the XXI-st century. Safety in Aviation and Space Technologies: Proceeding of The Tenth World Congress (Kyiv, 28 –30 September 2022), 2022. P. 2.1.9-2.1.13.

10. Zakutynskyi, I. Neural networks for road accident predictions and analysis. AUTOMATION, COMPUTER-INTEGRATED TECHNOLOGIES, AND PROBLEMS OF ENERGY EFFICIENCY IN INDUSTRY: Conference proceeding, Kropyvnytskyi, 10-11 November 2022. P. 166-168.

11. Закутинський І.В. IoT система для моніторингу та аналізу даних громадського транспорту. Інноваційні технології розвитку та ефективності функціонування автомобільного транспорту: Збірник матеріалів Міжнародної науково-практичної інтернет-конференції, Кропивницький, 17 – 19 листопада 2022 р. С. 34-37.

12. Zakutynskyi, I., & Rabodzei, I. IoT system architecture for monitoring and analyzing public transport data. 4th International Conference on Multidisciplinary Innovation in Academic Research (ICMIAR-2023): Conference proceedings, 17th & 18th March 2023, Chennai, India. P. 18.

13. Zakutynskyi, I., Sibruk, L. Security framework for IoT-Based public transport systems: A comprehensive analysis and design. International Conference on Research in Engineering, Technology and Science (ICRETS): Abstract Book July 6-9, 2023 - Budapest, Hungary. P. 37.