НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.4:004.7:621.391

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Адаптивний MQTT-шлюз для вузьких IoT-каналів (NB-IoT/LoRa)**»**

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІП-43мп
Ніконоров Андрій Олексійович _____

Керівник:
доцент кафедри ІСТ, к.т.н., доц.,
Поперешняк Світлана Володимирівна _____

Рецензент:
завідувач кафедри технологій цифрового розвитку Державного університету інформаційно-комунікаційних технологій д.т.н., професор,
Жебка Вікторія Вікторівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2025р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Ніконоров Андрій Олексійович

1. Тема дисертації «Адаптивний MQTT-шлюз для вузьких IoT-каналів (NB-IoT/LoRa)», науковий керівник дисертації Поперешняк Світлана Володимирівна кандидат фізико-математичних наук, доцент, доцент кафедри інформатики та програмної інженерії, затверджені наказом по університету від «06» листопада 2025 р. № 4841-с
2. Термін подання студентом дисертації «15» грудня 2025 р.
3. Об’єкт дослідження – програмне забезпечення для збору, трансформації і транспортування телеметрії в мережах з обмеженими ресурсами.
4. Предмет дослідження – процеси розроблення та забезпечення якості і експериментальна оцінка політик, орієнтованих на досягнення цілей якості обслуговування

5. Перелік завдань, які потрібно розробити – Проаналізувати наявні протоколи й вимоги; сформулювати вимоги до нового рішення; змоделювати канал і політики; реалізувати адаптивний шлюз без змін у брокері; порівняти зі статичною схемою за критеріями ефективності та надійності.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 3 плакати

7. Орієнтовний перелік публікацій – три публікації

8. Консультанти розділів дисертації

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

9. Дата видачі завдання «1» вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Отримання завдання, уточнення теми, формування календарного плану, узгодження з науковим керівником	1.09.2025	
2	Аналіз предметної області, протоколів MQTT/MQTT-SN/CoAP, технологій NB-IoT/LoRaWAN, формулювання мети й завдань	01.10.2025	
3	Розроблення теоретичної моделі адаптивної телеметрії (SLO/SLI, алгебра політик, FRP-модель, інваріанти стабільності)	20.10.2025	
4	Проєктування архітектури адаптивного MQTT-шлюзу, структури БД та підсистеми моніторингу	05.11.2025	
5	Реалізація програмного забезпечення шлюзу та інтеграція з MQTT v5, Prometheus, Grafana	15.11.2025	
6	Розроблення методики A/B-тестування, налаштування мережевої емуляції NB-IoT/LoRa (tc netem)	22.11.2025	
7	Виконання експериментальних досліджень та збирання метрик	24.11.2025	
8	Оформлення пояснювальної записки	25.11.2025	
9	Подання дисертації на попередній захист	26.11.2025	
10	Подання дисертації на захист	15.12.2025	

Студент Ніконоров Андрій Олексійович

Науковий керівник Поперешняк Світлана Володимирівна

РЕФЕРАТ

Розмір пояснювальної записки – 110 аркушів, містить 9 ілюстрацій, 34 таблиці, 4 додатки, 25 посилань на джерела.

Актуальність теми. У роботі розглянуто проблему ефективної доставки IoT-телеметрії у вузькосмугових каналах зв'язку NB-IoT та LoRaWAN з використанням брокерної моделі MQTT, показано основні особливості існуючих протоколів і підходів до телеметрії в LPWAN-мережах, їх переваги та недоліки щодо затримок, втрат і використання ресурсу каналу. Виявлено потребу в розробці адаптивного MQTT-шлюзу, який у реальному часі підлаштовує формат, частоту та надійність передавання під стан каналу без модифікації брокера.

Мета дослідження. Основною метою є підвищити ефективність і надійність доставки IoT-телеметрії у вузькосмугових каналах (NB-IoT/LoRa) шляхом адаптації параметрів передавання в реальному часі та використання можливостей MQTT v5 без змін у брокері.

Об'єкт дослідження: програмне забезпечення для збору, трансформації та транспортування IoT-телеметрії у вузькосмугових LPWAN-мережах (NB-IoT/LoRa).

Предмет дослідження: процеси розроблення, модифікації та забезпечення якості програмного забезпечення адаптивної телеметрії, а також методи побудови й експериментальної оцінки політик адаптації для MQTT-шлюзу, орієнтованих на досягнення цільових показників якості обслуговування.

Для реалізації поставленої мети **сформульовані наступні завдання**:

- проаналізувати існуючі протоколи та підходи до телеметрії в NB-IoT/LoRa та вимоги до якості доставки даних;
- сформулювати вимоги до адаптивного MQTT-шлюзу й побудувати модель SLO/SLI для вузьких IoT-каналів;

- розробити функціональну алгебру політик та FRP-модель оцінювання стану каналу для опису адаптації телеметрії;
- реалізувати адаптивний MQTT-шлюз на основі розробленої моделі без внесення змін до брокера;
- провести експериментальні дослідження із емуляцією умов NB-IoT/LoRaWAN та порівняти адаптивну схему зі статичною за критеріями трафіку, затримки, втрат та стабільності.

Наукова новизна результатів магістерської дисертації полягає в тому, що запропоновано формальну функціональну алгебру політик і FRP-модель для побудови адаптивного MQTT-шлюзу у вузькосмугових IoT-мережах, яка на відміну від існуючих рішень розглядає формат, батчинг, швидкість і рівень надійності як єдиний конвеєр чистих перетворень над потоком телеметрії, забезпечує формальні інваріанти стабільності (обмежені черги, відсутність осциляцій) та має операційне відображення на механізми MQTT v5 без модифікації брокера. Результат досягнутий шляхом побудови алгебри операторів Encode/Delta/Batch/Throttle/QoS/Timeout, використання FRP-сигналів для оцінювання стану каналу та інтеграції цих механізмів із можливостями MQTT v5.

Практичне значення отриманих результатів полягає в тому, що розроблено референсну архітектуру і програмну реалізацію адаптивного MQTT-шлюзу, у якій політики Encode, Delta, Batch, Throttle, QoS та Timeout об’єднані в межах одного застосунку й інтегровані з системою моніторингу на основі Prometheus та Grafana. Дана система може використовуватися як проміжний програмний компонент між IoT-пристроями та наявним MQTT-брокером без змін у брокері та впроваджуватися в системах розумних міст, аграрного сектору, промислової телеметрії, моніторингу довкілля й інфраструктурної безпеки.

Зв’язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію у доповідях на VI Міжнародній науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи (ITMAS–2025)» та XVIII Міжнародній науково-практичній конференції «Інформаційні технології і автоматизація – 2025», де окремо представлено прикладну реалізацію алгебри політик і FRP-керування для MQTT v5 у NB-IoT/LoRaWAN.

Публікації. Наукові положення дисертації опубліковані в:

1) Ніконоров А.О., Поперешняк С.В. Функціональні алгебри політик та реактивні моделі для адаптивної телеметрії в мережах з обмеженими ресурсами // Наука і техніка сьогодні (Серія «Техніка»). – 2025. – № 10(51). – С. 1844–1861. (Категорія «Б», спеціальність 122 «Комп’ютерні науки»).
2) Ніконоров А.О. SLODRIFT: прикладна реалізація алгебри політик і FRP-керування для MQTT v5 у NB-IoT/LoRaWAN // Інформаційні технології: моделі, алгоритми, системи (ITMAS–2025): Матеріали VI Міжнар. наук.-практ. інтернет-конф. – Миколаїв: НУК ім. адмірала Макарова, 2025. – С. 385–387.
3) Ніконоров А.О. Функціональна алгебра політик і FRP для стабільної адаптивної телеметрії в NB-IoT та LoRaWAN // XVIII Міжнар. наук.-практ. конф. «Інформаційні технології і автоматизація – 2025». – Одеса: ОНТУ, 2025. – С. 854–856.

Ключові слова: АДАПТИВНИЙ MQTT-ШЛЮЗ, ІНТЕРНЕТ РЕЧЕЙ, NB-IoT, LORAWAN, АЛГЕБРА ПОЛІТИК, ТЕЛЕМЕТРІЯ.

ABSTRACT

Explanatory note size – 110 pages, contains 9 illustrations, 34 tables, 4 applications, 25 references.

Topicality. The thesis examines the problem of reliable and efficient telemetry delivery in narrowband IoT networks NB-IoT and LoRaWAN, with the object being software for collecting, transforming and transporting IoT telemetry through an MQTT gateway. The main features of existing solutions based on MQTT/MQTT-SN/CoAP and ADR mechanisms are analysed, together with their advantages and limitations in constrained channels. The need is identified for developing a formally grounded adaptive MQTT gateway which tunes message rate, format and reliability to the actual channel state without any modifications to the broker.

The aim of the study. The main target is to increase the efficiency and reliability of IoT telemetry delivery over narrowband NB-IoT/LoRa channels by adapting message encoding, batching and QoS parameters in real time without changing the MQTT broker.

The object of research: software for collection, transformation and transportation of IoT telemetry in narrowband networks (NB-IoT/LoRa).

The subject of research: processes of design, modification and quality assurance of adaptive telemetry software, as well as methods for building and experimentally evaluating adaptive MQTT-gateway policies based on a functional policy algebra and functional reactive programming.

To achieve this goal, the **following tasks** were formulated:

- to analyse existing protocols and approaches to telemetry in NB-IoT/LoRa and the requirements to data-delivery quality;
- to define requirements, SLO/SLI and a model of network profiles for the new solution; to develop a functional policy algebra and an FRP-based channel-state model;

- to implement an adaptive MQTT gateway with MQTT v5 integration and without broker changes;
- to design an experimental methodology, emulate NB-IoT/LoRa channels and compare the adaptive scheme with a static one by efficiency and reliability criteria.

The scientific novelty of the results of the master's dissertation is that a compact functional policy algebra for adaptive telemetry in NB-IoT/LoRa and an SLO-driven FRP control model for an MQTT gateway are proposed. In contrast to existing approaches, they treat encoding, delta encoding, batching, rate limiting, reliability level and data freshness as a single formal processing pipeline with clear denotational semantics, rewriting laws and flow-stability invariants. The result was achieved by constructing an FRP-based channel-state model, introducing contractive policy updates and providing an operational mapping of the algebra onto MQTT v5 mechanisms (Receive Maximum, Maximum Packet Size, Topic Alias, Message Expiry) without broker modification.

The practical value of the obtained results is that a prototype adaptive MQTT gateway based on Scala/ZIO has been implemented, in which compression, delta encoding, batching and SLO-oriented flow control methods are combined inside a single application and remain fully compatible with standard MQTT v5 brokers without any changes on their side. A Prometheus/Grafana-based monitoring stack and a reproducible NB-IoT/LoRa testbed using tc netem have also been built, so the proposed gateway can be used as an intermediate software component in smart-city, agriculture, industrial telemetry, environmental monitoring and infrastructure-safety systems.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Computer Science and Software Engineering of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute».

Approbation. The scientific provisions of the dissertation were tested at the The scientific results of the dissertation were presented and discussed at the VI International

Scientific and Practical Internet Conference “Information Technologies: Models, Algorithms, Systems (ITMAS–2025)” and the XVIII International Scientific and Practical Conference “Information Technologies and Automation – 2025”, where an applied implementation of the policy algebra and FRP-based control for MQTT v5 in NB-IoT/LoRaWAN was demonstrated.

Publications. The scientific provisions of the dissertation were published in:

1) Nikonorov A.O., Popereshnyak S.V. Functional policy algebras and reactive models for adaptive telemetry in resource-constrained networks // Science and Technology Today (Series “Engineering”). 2025, no. 10(51), pp. 1844–1861.
2) Nikonorov A.O. SLODRIFT: applied implementation of policy algebra and FRP-based control for MQTT v5 in NB-IoT/LoRaWAN // Information Technologies: Models, Algorithms, Systems (ITMAS–2025): Proc. of the VI Int. Sci.-Pract. Internet Conf. – Mykolaiv, Admiral Makarov NUS, 2025, pp. 385–387.
3) Nikonorov A.O. Functional policy algebra and FRP for stable adaptive telemetry in NB-IoT and LoRaWAN // XVIII Int. Sci.-Pract. Conf. “Information Technologies and Automation – 2025”. – Odesa: ONTU, 2025, pp. 854–856.

Keywords: ADAPTIVE MQTT GATEWAY, INTERNET OF THINGS, NB-IoT, LORAWAN, POLICY ALGEBRA, TELEMETRY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ADR (Adaptive Data Rate) – адаптивна швидкість передавання даних у мережах LoRaWAN;

API (Application Programming Interface) – програмний інтерфейс застосунку;

CBOR (Concise Binary Object Representation) – компактний бінарний формат представлення об'єктів;

CoAP (Constrained Application Protocol) – протокол прикладного рівня для обмежених пристроїв;

EWMA (Exponentially Weighted Moving Average) – експоненційно зважене ковзне середнє;

FRP (Functional Reactive Programming) – функціональне реактивне програмування;

IoT (Internet of Things) – Інтернет речей;

JSON (JavaScript Object Notation) – текстовий формат обміну даними;

KAT (Kleene Algebra with Tests) – алгебра Кліні з тестами;

LoRa (Long Range) – технологія радіозв'язку далекого радіусу дії;

LoRaWAN (Long Range Wide Area Network) – протокол мережевого рівня для мереж LoRa;

LPWAN (Low-Power Wide-Area Network) – енергоефективна мережа широкого покриття;

MAC (Medium Access Control) – керування доступом до середовища;

MQTT (Message Queuing Telemetry Transport) – протокол обміну телеметричними повідомленнями;

MQTT-SN (MQTT for Sensor Networks) – варіант MQTT для сенсорних мереж;

NB-IoT (Narrowband Internet of Things) – вузькосмуговий Інтернет речей;

PHY (Physical Layer) – фізичний рівень;

Protobuf (Protocol Buffers) – бінарний протокол серіалізації даних;

QoS (Quality of Service) – якість обслуговування;

RTT (Round-Trip Time) – час проходження пакета туди і назад;

SF (Spreading Factor) – коефіцієнт розширення спектру;

SLI (Service Level Indicator) – показник рівня сервісу;

SLO (Service Level Objective) – цільовий показник рівня сервісу;

SNR (Signal-to-Noise Ratio) – співвідношення сигнал/шум;

TCP (Transmission Control Protocol) – протокол керування передаванням;

ToA (Time on Air) – час перебування сигналу в ефірі;

UDP (User Datagram Protocol) – протокол датаграм користувача;

ZIO – бібліотека функціонального програмування для Scala.

ВСТУП

Сучасний розвиток Інтернету речей (Internet of Things, IoT) супроводжується швидким зростанням кількості пристроїв, які функціонують у середовищах з обмеженими ресурсами. Особливе місце серед них займають мережі з низьким енергоспоживанням і великим радіусом дії (Low-power Wide-area Networks, LPWAN), зокрема NB-IoT та LoRaWAN. Ці технології забезпечують масштабованість і енергоефективність, проте характеризуються низькою пропускною здатністю, значними затримками у каналах зв'язку, високою ймовірністю втрати пакетів та жорсткими регіональними обмеженнями ефірного часу.

За таких умов використання традиційних, статичних схем передавання даних — із фіксованим періодом передачі, громіздкими форматами повідомлень на зразок JSON та жорстко заданими рівнями якості обслуговування (Quality of Service, QoS) — призводить до перевитрат ресурсів та порушення сервісних угод щодо рівня якості обслуговування (Service Level Objectives, SLO). Вузькосмугові IoT-канали характеризуються підвищеною затримкою, втратами та обмеженою пропускною здатністю; статичні схеми передачі призводять до перевитрат трафіку й деградації SLO.

Актуальність теми дослідження зумовлена необхідністю створення методів адаптації, що реагують на зміну мережевих умов у реальному часі, з прозорими гарантіями стабільності потоку. Існуючі методи зосереджені на ізольованих механізмах — наприклад, ADR у LoRaWAN або керування потоком у MQTT v5 (Receive Maximum, Maximum Packet Size, Topic Alias, Message Expiry), проте відсутній цілісний апарат, який би дозволяв інтегрувати ці можливості у прозору, формально обґрунтовану модель адаптації.

Практична значущість проблеми полягає у тому, що без адаптивних політик на рівні прикладного шару відбувається деградація сервісів IoT-телеметрії: зростає

час доставки повідомлень, порушуються показники надійності, знижується ефективність використання енергоресурсів і ефірного часу. Це є критично важливим у сценаріях «розумних» міст, промислової автоматизації, моніторингу довкілля та інфраструктурної безпеки, де кожна затримка або втрата пакета може мати значні соціально-економічні наслідки.

Метою роботи є підвищення ефективності і надійності доставки телеметрії шляхом адаптації параметрів передачі у реальному часі, що забезпечить: зменшення трафіку, p95-латентність, зниження частки втрат та збереження стабільності потоку (backpressure-інваріанти).

Об'єктом дослідження є програмне забезпечення для збору, трансформації та транспортування IoT-телеметрії у вузькосмугових мережах.

Предметом дослідження є процеси розроблення, модифікації, аналізу, забезпечення якості, впровадження і супроводження програмного забезпечення адаптивної телеметрії на основі алгебри політик та функціонального реактивного програмування.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати існуючі рішення для вузькосмугових IoT-каналів, протоколи MQTT/MQTT-SN/CoAP, технології NB-IoT та LoRaWAN, моделі трафіку та методи адаптації;
- сформулювати SLO/SLI та побудувати модель мережевих профілів (RTT, loss, duty-cycle);
- розробити алгебру політик як чистих функцій (функціональна парадигма) з операторами Encode, Delta, Batch, Throttle, QoS, Guard, Choose, Timeout та довести інваріанти стабільності потоку;
- реалізувати адаптивний MQTT-шлюз на базі Scala/ZIO з інтеграцією механізмів MQTT v5;

- побудувати методику експериментів (синтетичні сценарії, стрес-тести, абляційні дослідження);
- оцінити ефективність запропонованого рішення за критеріями трафіку, латентності, втрат і стабільності.

Наукова новизна одержаних результатів полягає в удосконаленні методів потокової доставки телеметрії у constrained-мережах шляхом використання функціональної декомпозиції та чистих політик адаптації з формальними інваріантами backpressure. Набуло подальшого розвитку підходи до SLO-орієнтованого керування потоком, що підвищують ефективність розроблення та модифікації програмного забезпечення для IoT.

Практичне значення одержаних результатів полягає у розробленні референсної архітектури адаптивного шлюзу та відтворюваної методики експериментальної оцінки, придатної для застосування у смарт-інфраструктурі (будівлі, агро, промисловий моніторинг) та інтеграції з наявними системами телеметрії.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Адаптивні MQTT-шлюзи для мереж NB-IoT та LoRa

Стрімка експансія концепції Інтернету речей (Internet of Things, IoT) зумовила активний розвиток мережевих рішень для обчислювально та енергетично обмежених пристроїв, що здійснюють обмін даними через вузькосмугові канали зв’язку. У цьому контексті формуються два ключові та взаємодоповнювальні наукові напрями, кожен з яких відіграє фундаментальну роль у забезпеченні ефективності сучасних телеметричних систем.

Перший напрям передбачає розроблення адаптивних шлюзів (adaptive gateways) на основі протоколу MQTT для енергоефективних мереж широкого покриття (Low-Power Wide Area Networks, LPWAN), таких як NB-IoT та LoRaWAN. Подібні шлюзи виконують функцію проміжної ланки між обмеженими за ресурсами бездротовими каналами та хмарною інфраструктурою, забезпечуючи прозору маршрутизацію та агрегацію телеметричних даних. Для коректного функціонування в умовах низької пропускної здатності та високих затримок вони мають застосовувати спеціалізовані протоколи обміну — зокрема MQTT, MQTT-SN та CoAP — що оптимізовані для роботи у середовищах із нестабільними каналами та значними мережевими обмеженнями.

Другий напрям охоплює формалізацію алгебр політик (policy algebras) і реактивних моделей програмування, які забезпечують адаптивне управління потоками телеметрії на рівні логіки. Функціональне реактивне програмування (Functional Reactive Programming, FRP) та алгебраїчні політичні фреймворки, зокрема алгебри Кліні з тестами (Kleene Algebra with Tests), пропонують високорівневі формальні засоби для опису, композиції та верифікації механізмів адаптації в телеметричних системах з обмеженими ресурсами. Застосування цих

підходів забезпечує можливість динамічного, самокерованого пристосування режимів передавання даних без втручання оператора.

Технології вузькосмугових мереж для Інтернету речей (IoT), зокрема Narrowband IoT (NB-IoT) та LoRaWAN, на сьогодні посідають провідні позиції серед рішень класу LPWAN (Low-Power Wide Area Network). Попри належність до одного технологічного сегмента, ці два підходи демонструють концептуально різні архітектурні принципи, що зумовлює відмінності в пропускній здатності, надійності, моделях енергоспоживання та характеристиках сервісної підтримки.

NB-IoT є стандартизованою 3GPP технологією радіодоступу (radio access technology), яка функціонує у ліцензованих частотних діапазонах у межах інфраструктури стільникового зв'язку LTE/5G. Завдяки цьому NB-IoT забезпечує високий рівень надійності передавання, підтримку повноцінної IP-маршрутизації та засоби управління якістю обслуговування (Quality of Service, QoS). Водночас інтеграція з операторськими мережами ускладнює архітектуру та підвищує вартість розгортання і супроводу, що особливо суттєво для систем масового IoT-впровадження.

На відміну від NB-IoT, технологія LoRaWAN працює у неліцензованих діапазонах, що дозволяє знизити експлуатаційні витрати та значно спростити мережеву інфраструктуру. Проте така модель функціонування супроводжується обмеженими гарантіями QoS, оскільки LoRaWAN використовує спрощений MAC-рівень (Medium Access Control) і не забезпечує захищеної від колізій багатокористувацької взаємодії на рівні радіоінтерфейсу. Натомість ключовими перевагами LoRaWAN є екстремальна енергоефективність та широке покриття, що дає змогу підтримувати масштабні розподілені IoT-мережі з тривалим автономним живленням.

Порівняльний аналіз, наведений у роботі Mekki та ін. (2019), демонструє, що неліцензовані LPWAN-технології (LoRa, Sigfox) перевершують NB-IoT за

показниками тривалості роботи від батареї, масштабованості та вартості розгортання. Натомість NB-IoT забезпечує кращі характеристики затримки (latency) та більш передбачувану доставку пакетів завдяки інтеграції із стільниковою інфраструктурою, що передбачає централізоване управління радіоресурсами та вбудовані механізми контролю QoS.

Узагальнюючи, LoRaWAN орієнтована на мінімізацію енергоспоживання та максимізацію зони покриття, тоді як NB-IoT спрямована на забезпечення надійного сервісу з детермінованішими параметрами затримки та доступності каналу. Саме це протиріччя між енергоефективністю та надійністю формує прикладну потребу в адаптивних шлюзах (adaptive gateways), здатних динамічно узгоджувати характеристики кінцевих вузлів із вимогами хмарних сервісів, зокрема під час транспортування телеметрії за допомогою протоколу MQTT.

1.2 Протоколи MQTT та MQTT-SN у середовищах із обмеженими каналами зв'язку

Протокол Message Queue Telemetry Transport (MQTT) нині фактично виконує роль де-факто стандарту в індустрії IoT завдяки своїй компактній публікаційно-підписній моделі (publish/subscribe) та архітектурній простоті. MQTT функціонує на основі централізованого брокера повідомлень (message broker) і зазвичай використовує транспортний протокол TCP, підтримуючи три рівні якості обслуговування (Quality of Service, QoS) для доставки телеметричних повідомлень — «не більше одного разу» (QoS 0), «принаймні один раз» (QoS 1) та «точно один раз» (QoS 2). Низька складність, асинхронний характер обміну даними та логічне роз'єднання видавців і підписників визначили популярність MQTT у застосуваннях із обмеженими обчислювальними ресурсами.

Водночас орієнтація MQTT на TCP створює низку проблем у вузькосмугових каналах зв'язку, таких як NB-IoT або однохопові з'єднання LoRaWAN. У таких середовищах накладні витрати (overhead) TCP — включно з процедурою

встановлення з'єднання (three-way handshake) та підтриманням сеансу — стають істотними, а часові затримки є відчутними на прикладному рівні.

З метою усунення цих недоліків була розроблена модифікація MQTT — MQTT for Sensor Networks (MQTT-SN), яка мінімізує протокольні витрати завдяки функціонуванню поверх UDP, а також запроваджує механізми стиснення тем (topic aliasing) та «сплячі» режими клієнтів (sleeping modes). MQTT-SN повністю уникає встановлення TCP-з'єднання, що є критичною перевагою для каналів із високою затримкою, обмеженим енергобюджетом або низькою доступністю мережі. Попри відсутність TCP, MQTT-SN зберігає прикладний рівень надійності, реалізуючи підтвердження доставки, повторні передавання та брокерське керування QoS.

Ще одним протоколом прикладного рівня, що часто розглядається разом із MQTT, є Constraint Application Protocol (CoAP). CoAP — це компактний REST-орієнтований протокол, який працює поверх UDP і завдяки цьому є нативно легковаговим та придатним для пристроїв із обмеженими ресурсами. CoAP реалізує підтверджувані повідомлення (confirmable messages) як механізм забезпечення надійності, що концептуально відповідає QoS у MQTT, однак базується на моделі запит/відповідь (client-server).

Експериментальне дослідження Larmo та ін. (2019) порівняло ефективність CoAP та MQTT у каналі NB-IoT [1] і показало, що CoAP поверх UDP стабільно перевершує MQTT поверх TCP за умов інтенсивного навантаження, зокрема за метриками затримки, зони покриття та місткості системи [1]. Завдяки відсутності встановлення транспортного з’єднання та меншому заголовковому overhead CoAP забезпечує вищу прикладну пропускну здатність і масштабованість NB-IoT-комірки [1]. MQTT поверх TCP, за результатами дослідження, зберігає задовільні характеристики лише за низького навантаження [1]; у міру зростання кількості пристроїв сеансова природа TCP і пов’язані накладні витрати спричиняють деградацію продуктивності [1]. Це узгоджується з попередніми висновками інших

авторів, які також підкреслюють неефективність TCP для телеметричного IoT-трафіку через його надлишковість та негнучкість для коротких, спорадичних повідомлень, толерантних до втрат [1].

Підсумовуючи, для рідкісних та малих телеметричних передач UDP-орієнтовані протоколи (CoAP або MQTT-SN) у більшості сценаріїв виявляються ефективнішими у вузькосмугових мережах, ніж MQTT поверх TCP [1]. Водночас MQTT/TCP має сенс у випадках, коли умови каналу помірні, а шаблон втрат ускладнює використання чистих UDP-підходів, що не мають гарантованої доставки [1].

1.3 Механізми адаптації швидкості та шлюзові стратегії для LoRaWAN та NB-IoT

З огляду на наведені вище результати, адаптивний MQTT-шлюз для мереж NB-IoT та LoRaWAN має забезпечувати можливість динамічної зміни протоколів або переналаштування їх параметрів у відповідь на поточний стан мережі. Одним із практичних підходів є адаптивне керування якістю обслуговування (Quality of Service, QoS) на прикладному рівні. Palmese та ін. (2022) запропонували динамічний контролер QoS для протоколу MQTT-SN, орієнтований на бездротові сенсорні мережі [3]. У такій архітектурі шлюз або брокер здійснює безперервний моніторинг наскрізної затримки (end-to-end delay) та частоти помилок під час передавання пакетів (packet error rate), після чого автоматично призначає оптимальний рівень QoS для повідомлень кожного окремого вузла [3]. Наприклад, у разі погіршення якості каналу — підвищеної частоти втрат або стрибків затримки — контролер підвищує рівень QoS (зокрема, переходить до режимів, що вимагають підтверджень або навіть «доставки рівно один раз»), забезпечуючи гарантовану надійність доставки. За сприятливих мережевих умов, навпаки, система може знижувати QoS, мінімізуючи накладні витрати. Експериментальні результати демонструють, що така адаптивна система керування QoS істотно підвищує ймовірність доставки

пакетів та зменшує затримку порівняно зі статичним, обраним користувачем рівнем QoS [3]. Централізоване керування QoS також унеможливлює хибний вибір користувача, який може встановити занадто низький рівень (ризик втрати даних у несприятливих умовах) або надмірно високий (неефективне використання пропускної здатності) [3]. Наведений приклад демонструє потенціал шлюзів до інтелектуального узгодження протоколів і параметрів транспортного рівня з актуальним станом мережі, що дає змогу оптимізувати якість обслуговування для кожного термінального пристрою [3].

У мережах LoRaWAN адаптація відбувається переважно на фізичному та канальному рівнях (PHY/MAC) завдяки механізму Adaptive Data Rate (ADR). ADR динамічно змінює параметри передавання вузла — коефіцієнт розширення спектра (spreading factor), швидкість кодування (coding rate) та потужність передавання — з метою максимізації пропускної здатності за умови збереження покриття. Однак стандартний механізм ADR передбачає здебільшого стаціонарні або малорухомі вузли. Дослідження Kousias та ін. (2019) показало, що ADR дійсно підвищує надійність і стабільність зв’язку в статичних сценаріях, але його ефективність суттєво знижується зі зростанням мобільності вузлів [4]. Фактично мобільні пристрої часто використовують застарілі параметри ADR, оскільки мережа не встигає достатньо швидко реагувати на зміну якості сигналу. Це дозволяє сформулювати практичний висновок: для мобільних застосувань LoRa шлюз має володіти прискореними механізмами оцінки якості каналу або навіть прогнозними (predictive) алгоритмами, які доповнюють або замінюють класичний ADR. У сучасних дослідженнях уже пропонуються покращені ADR-алгоритми, зокрема алгоритми, що враховують мобільність або спираються на методи машинного навчання, щоб підвищити реактивність мережі до динаміки середовища [4][5]. На практиці шлюз LoRa, що передає дані до MQTT-брокера, може в реальному часі аналізувати радіометричні показники (RSSI, SNR) і на їх підставі запускати

прикладні адаптації — наприклад тимчасове зниження частоти передавання повідомлень або зміну схеми кодування у разі деградації каналу.

Третій вимір адаптації — це динамічний вибір стеку протоколів залежно від сценарію використання. Як показано на у роботі Larmo та ін. [1], IoT-система може поєднувати різні комбінації протоколів (CoAP проти MQTT, UDP проти TCP, різні рівні захисту трафіку). Адаптивний шлюз має потенціал виступати «переговорним компонентом» (negotiation entity), наприклад використовуючи CoAP/UDP для регулярної телеметрії з низьким навантаженням та перемикаючись на MQTT/TCP для критичних команд, що потребують гарантованої доставки. Додатково шлюз може виконувати стиснення заголовків (header compression) або протокольну трансляцію (protocol translation). Зокрема, оскільки LoRaWAN і Sigfox не передають IP-пакети у висхідному каналі з міркувань економії ефірного часу, в межах IETF розробляються методи стиснення заголовків для передавання IPv6/UDP поверх таких каналів [1]. У цьому випадку шлюз на периферії (edge gateway) може здійснювати зворотне стиснення/розпакування заголовків або навіть перетворення CoAP→MQTT під час передачі в хмару. Таким чином, багатопротокольний шлюз відіграє роль уніфікатора, вирівнюючи відмінності між протоколами: пристрій використовує оптимальний для каналу стек (наприклад, CoAP поверх невеликого UDP-фрейма для LoRaWAN або MQTT-SN для NB-IoT), тоді як шлюз забезпечує прозору сумісність із MQTT-брокерами та HTTP-сервісами на хмарному боці.

1.4 Порівняльний аналіз протоколів для IoT

З метою кращого розуміння підходів до проєктування адаптивних систем телеметрії у вузькосмугових мережах у таблиці 1 подано порівняння ключових протоколів і технологій, що найчастіше застосовуються у відповідних сценаріях. Таблиця узагальнює їх рівень функціонування в мережевій моделі, тип транспортного протоколу та найбільш характерні функційні властивості, що мають

безпосередній вплив на адаптивність, надійність і ефективність передавання телеметричних даних у середовищах із обмеженими ресурсами.

Таблиця 1.1 — Порівняння протоколів для IoT

Протокол / Технологія	Тип / Рівень	Транспортний / Мережевий	Помітні Характеристики
MQTT 5.0	Публікаційно-підписний протокол прикладного рівня (Pub/Sub)	TCP/IP (IPv4/6)	Текстові топіки; централізований брокер, що маршрутизує публікації. Підтримує рівні QoS 0–2 (від ненадійної доставки до гарантії «доставки рівно один раз»). Простий у реалізації на клієнтах [3], однак встановлення TCP-з'єднання збільшує затримку на обмежених каналах [1]. Доцільний для надійної доставки подій у відносно стабільних мережах.

Продовження таблиці 1.1

Протокол / Технологія	Тип / Рівень	Транспортний / Мережевий	Помітні Характеристики
MQTT-SN 1.2	Pub/Sub для сенсорних мереж	UDP або спеціалізований транспорт (без TCP)	Бінарний формат із відображенням ідентифікаторів тем (зменшує розмір переданих даних). Відсутність постійного TCP-сеансу — повторні передавання забезпечує брокер. Підтримує QoS на прикладному рівні аналогічно MQTT [3]. Оптимальний для «сплячих» вузлів і малопотужних зіркових топологій із мінімальним overhead. Потребує шлюзу MQTT-SN↔MQTT.

Продовження таблиці 1.1

Протокол / Технологія	Тип / Рівень	Транспортний / Мережевий	Помітні Характеристики
CoAP (RFC 7252)	REST-протокол прикладного рівня	UDP (опційно TCP)	Компактні бінарні заголовки та методи. Модель запит/відповідь (client–server) із підтверджуваними повідомленнями. Відсутній брокер — обмін відбувається напряму. Має суттєво менший overhead, ніж MQTT [1], і підходить для рідкісних сенсорних вимірювань та прямої взаємодії «пристрій–хмара». Вбудована підтримка multicast.

Продовження таблиці 1.1

Протокол / Технологія	Тип / Рівень	Транспортний / Мережевий	Помітні Характеристики
LoRaWAN 1.0.4	LPWAN MAC/мережевий рівень (топологія зірка)	Радіоканал LoRa (ISM sub-GHz)	Спред-спектровий PHY з адаптивною швидкістю (0,3–50 кбіт/с). Передавання переважно у висхідному напрямі з малими корисними навантаженнями (~50 байт). Доступне MAC-підтвердження, однак втрата пакетів є типовою — надійність покладається на верхні рівні. Використовуються обмеження duty cycle та ADR для оптимізації [4]. Висока проникність сигналу й багаторічна автономність вузлів [2], але значна затримка (секунди) та низька пропускна здатність. Потребує шлюзу для передачі у хмару (зазвичай MQTT/HTTP).

Кінець таблиці 1.1

Протокол / Технологія	Тип / Рівень	Транспортний / Мережевий	Помітні Характеристики
NB-IoT (LTE Cat-NB)	LPWAN стільникового класу (RAN)	LTE (ліцензований спектр)	3GPP-стандарт зі швидкістю 20–250 кбіт/с. Застосовує повторні коди та високу чутливість приймача (бюджет каналу ~164 dB). Як правило, використовує UDP (наприклад, CoAP) або TCP (MQTT) [1]. Затримка 1–10 с, але стабільність та QoS істотно вищі, ніж у неліцензованих LPWAN [1]. Висока надійність і інтеграція з інфраструктурою оператора (SIM-автентифікація, профілі QoS). Придатна для сценаріїв із гарантованою доставкою та підвищеною частотою передавання.

Пояснення до таблиці 1.1 та узагальнювальний аналіз

Таблиця 1.1, що порівнює протоколи прикладного рівня (MQTT, MQTT-SN, CoAP) та технології доступу LPWAN (LoRaWAN, NB-IoT), демонструє відмінності їхньої функціональної ролі у вузькосмугових телеметричних системах. MQTT та CoAP належать до протоколів прикладного рівня, тоді як LoRaWAN і NB-IoT забезпечують мережевий доступ (network access) на фізичному та канальному рівнях. Протокол MQTT-SN займає проміжну нішу, виступаючи механізмом

інтеграції між сенсорними мережами та MQTT-брокерами у хмарній інфраструктурі.

Як видно з таблиці, універсального протоколу, однаково ефективного для всіх типів вузькосмугових телеметричних сценаріїв, не існує. Протокол MQTT вирізняється завдяки своїй брокерській архітектурі, що забезпечує логічне роз'єднання видавців і підписників, а також завдяки підтримці вбудованих механізмів QoS, які дозволяють гарантувати доставку повідомлень. У свою чергу, CoAP демонструє перевагу за рахунок мінімального накладного службового трафіку, компактних заголовків та простоти реалізації, що робить його особливо ефективним на каналах із високою затримкою або жорсткими обмеженнями пропускної здатності.

Протокол MQTT-SN намагається поєднати архітектурну модель публікації-підписки MQTT з легковаговим транспортом, характерним для CoAP: використання UDP замість TCP зменшує затримки і спрощує роботу «сплячих» сенсорних вузлів, зберігаючи водночас семантику QoS, притаманну MQTT.

Вибір технології доступу визначає базові властивості системи телеметрії. LoRaWAN орієнтована на екстремальну енергоефективність і багаторічну автономність, але ціною стають підвищена затримка та неминучі втрати пакетів, які повинні компенсуватися на вищих рівнях [2]. Натомість NB-IoT, інтегрована у стільникову інфраструктуру, забезпечує передбачувану надійність і нижчу затримку, проте вимагає роботи в межах платної та керованої операторської мережі.

У такому контексті адаптивний шлюз можна інтерпретувати як політик-керований механізм (policy engine), що в кожний момент часу приймає рішення щодо вибору протоколу, стеку та параметрів передавання. Його завдання полягає у максимізації пропускної здатності й надійності за одночасної мінімізації витрат енергії та використання каналу. На практиці подібний шлюз може підтримувати кілька стеків протоколів та перемикатися між ними динамічно — наприклад,

застосовувати CoAP/UDP під час перевантаження мережі для зменшення overhead, тоді як у фазі низького навантаження або під час передавання критично важливих повідомлень — переключатися на MQTT/TCP, використовуючи надійність і строгі гарантії доставки TCP.

1.5 Функціональні алгебри політик та реактивні моделі для адаптивної телеметрії

Якщо попередній розділ був присвячений механізмам комунікаційної адаптації, то в цьому розділі зосереджено увагу на формальних та програмних моделях, які забезпечують можливість принципового, керованого та відтворюваного формування логіки адаптивної телеметрії. Побудова справді адаптивної системи потребує здатності обробляти асинхронні події (зокрема коливання стану мережі, динамічну появу і зникнення вузлів, зміни пропускної здатності каналів тощо) і застосовувати політики в режимі виконання — наприклад, змінювати маршрутизацію, обмежувати швидкість передавання телеметрії або перебудовувати протокольні параметри під впливом зовнішніх умов.

У цьому контексті виокремлюються два впливові концептуальні напрями:

– Функціональне реактивне програмування (Functional Reactive Programming, FRP), яке розглядає поведінку системи як реактивні потоки даних (reactive data flows) та подій, що еволюціонують у часі;

– Алгебри політик (policy algebras), зокрема алгебра Кліні з тестами (Kleene Algebra with Tests, KAT), які забезпечують формально строгий апарат для специфікації, доведення коректності та композиції політик адаптації.

У подальшому огляді здійснюється:

– аналіз еволюції FRP та особливостей його використання в мережевих і розподілених системах;

- розгляд алгебричних фреймворків на кшталт NetKAT, що застосовуються для опису мережевих політик та можуть бути адаптовані для телеметричних мереж;
- висвітлення результатів досліджень у сфері самоадаптивної обробки потоків (self-adaptive stream processing) та маршрутизації зі зворотним тиском (backpressure routing), що мають безпосереднє відношення до керування телеметрією в умовах обмежених ресурсів.

1.6 Функціонально-реактивне програмування (FRP) для потокової телеметрії

Концепція функціонально-реактивного програмування (Functional Reactive Programming, FRP) виникла наприкінці 1990-х років як модель формального опису інтерактивних і часовозалежних систем у межах чисто функційного підходу. У знаковій праці Еліота та Худака «Functional Reactive Animation» (1997) було запропоновано ключові абстракції — поведінки (behaviors) як неперервні в часі величини та події (events) як дискретні зміни стану [6]. На відміну від імперативних підходів, де розробник змушений явно визначати обробники подій (callbacks), FRP передбачає декларативний опис того, як система має реагувати на події або як значення змінюються в часі.

З позицій телеметрії FRP є особливо привабливою парадигмою, оскільки сенсорні вимірювання, мережеві затримки, системні події та команди користувача природно інтерпретуються як потоки подій (event streams), що впливають на неперервні сигнали стану (наприклад, оцінка поточної пропускної здатності каналу). FRP дозволяє декларативно комбінувати ці потоки за допомогою високорівневих операторів на кшталт map, fold, filter, у результаті чого утворюється динамічна специфікація поведінки системи. Наприклад, сигнал може відображати поточну частоту опитування сенсора, а сама частота визначається якістю каналу: якщо зростає інтенсивність подій втрати пакетів, то частота вимірювань автоматично зменшується. У FRP подібні залежності задаються стисло, тоді як

часове розповсюдження змін забезпечує сам рушій FRP, знімаючи з розробника імперативне керування оновленнями.

З часом дослідження FRP зосередилися на викональній ефективності та усуненні просторових витоків (space leaks). Ранні реалізації FRP інколи страждали від надмірного споживання пам’яті та затримок у розповсюдженні змін. У праці Нілссона та ін. «FRP, Continued» (2002) було уточнено семантику FRP, що дало змогу оптимізувати обчислення, а пізніше Еліот (2009) запропонував push-pull FRP-модель, яка поєднує два підходи до розповсюдження змін [6]. У такій моделі push-оновлення надсилаються негайно — коли це можливо, тоді як pull дозволяє обчислити значення на вимогу, якщо певні оновлення були пропущені, тим самим поєднуючи переваги обох стратегій і згладжуючи їхні недоліки (надмірність push або інертність pull). Гібридність push-pull FRP забезпечує стабільну реактивність навіть за умов пікових навантажень, що є критично важливою властивістю для адаптивної телеметрії, де раптові сплески подій можуть вивести з рівноваги наївні реактивні системи.

Сучасні FRP-фреймворки — такі як Reactive Extensions (Rx), ZIO Streams, Haskell Yampa та інші — упроваджують подібні оптимізації. Крім того, Кутс, Леш і Пейтон-Джонс (2007) запропонували метод stream fusion [7][8], який уможливлює перетворення послідовностей операторів над потоками на щільні цикли без створення проміжних структур даних. Це усуває витрати на алокації та збір сміття між конвеєрними етапами — властивість, особливо важлива у високочастотній телеметрії.

Узагальнюючи, FRP надає чисту з точки зору семантики модель реактивного керування потоками даних, тоді як такі вдосконалення, як push-pull-обчислення та stream fusion, забезпечують виконання, наближене до ефективності вручну оптимізованих циклів [8][9]. У контексті побудови адаптивних телеметричних шлюзів FRP дозволяє моделювати вхідні та вихідні потоки, визначати реактивні

залежності між ними та описувати адаптивну поведінку декларативно — наприклад: «якщо довжина черги перевищує поріг, стиснути або прорідити потік даних». Це напряму перегукується з алгебрами політик, що розглядатимуться далі, — FRP описує динаміку адаптації, тоді як алгебри політик формалізують інваріанти та правила, яких система має дотримуватися.

1.7 Алгебри політик та мережева обчислювальність (Network Calculus) для адаптації

Алгебра Кліні з тестами (Kleene Algebra with Tests, KAT) — це алгебрична система, запропонована Козеном (Kozen, 1997), яка поєднує класичну алгебру Кліні (що відповідає регулярним виразам і дає змогу моделювати послідовності дій) з булевими тестами, які визначають логічні умови виконання [10]. Інтуїтивно KAT забезпечує алгебричне подання керування потоком виконання програм, де оператор зірочки Кліні (*) моделює повторення чи ітерацію, а булеві тести виступають умовами, що дозволяють або блокують виконання певних дій.

Ключова фундаментальна властивість KAT полягає в тому, що вона надає рівнянне (equational) обґрунтування програмної логіки: замість доведення коректності на рівні станів чи переходів, у KAT можна маніпулювати програмою як алгебричними виразами та доводити властивості, трансформації чи еквівалентність програм за допомогою системи рівностей і тотожностей. Наприклад, дії send_packet або drop_packet можуть бути алгебраїчними символами, а тест buffer_full? — логічною умовою. Правило «якщо буфер переповнений — відкинути пакет, інакше — надіслати» у KAT трансформується у вираз, що підлягає формальному доведенню еквівалентності та коректності. Козен довів, що KAT є достатньо виразною для верифікації еквівалентності програм із циклами (while-programs) та інших перетворень [10].

На базі KAT Андерсон та ін. (2014) розробили мову NetKAT — предметно-орієнтований формалізм для опису та верифікації політик маршрутизації і

перенаправлення трафіку в комп’ютерних мережах [11]. NetKAT інтегрує у KAT мережеві примітиви, такі як зіставлення заголовків пакетів і переписування полів, і при цьому доведено, що NetKAT є формальним підвипадком KAT, який має повну та коректну (sound & complete) рівнянну теорію [11][12]. Це означає, що мережеві політики — на кшталт «для всіх пакетів: якщо порт=1, встановити порт=2 і переслати, інакше відкинути» — можуть бути не лише описані декларативно, а й автоматично верифіковані: доведено відсутність петель маршрутизації, порушень ізоляції або «чорних дір» у мережі [11]. З погляду адаптивної телеметрії це критично, оскільки алгебричний опис політик дозволяє гарантовано зберігати інваріанти навіть після динамічних оновлень політик. Наприклад, політика «якщо мережа перевантажена, то маршрутизувати через низькошвидкісний шлях, інакше — через основний» формально задається у NetKAT-подібній алгебрі, де можна довести, що в обох випадках досяжність і коректність маршрутизації зберігаються [11].

Інша дотична система — Frenetic (Foster et al., 2011) — об’єднала ідеї функціонально-реактивного програмування з програмуванням мереж, насамперед у SDN-середовищах [12][13]. Frenetic запропонував потокові абстракції для мережевих подій, що дозволило обробляти пакети як подійні потоки, застосовувати функціональні оператори (наприклад, для агрегації статистики чи динамічної зміни маршрутизації), уникаючи низькорівневих автоматів станів [14]. По суті, Frenetic трактує потоки мережевих подій так само, як FRP трактує потоки подій у часових системах, створюючи декларативний і композиційний підхід до визначення політик. Це дає потужну синергію: FRP-частина забезпечує реактивність у реальному часі (швидка реакція на перевантаження, збої вузлів тощо), а алгебрична складова гарантує формальну коректність реакцій.

1.8 Самоадаптивні телеметричні системи та механізми зворотного тиску (Backpressure)

Окрім формальних мов і алгебричних моделей, у науковій спільноті тривалий час досліджуються механізми самоадаптації в потокових та мережевих системах, спрямовані на підтримання стабільності та ефективності під час зміни навантаження. Класичним прикладом є маршрутизація зі зворотним тиском (backpressure routing), яка походить із мережевої теорії та була запропонована Тассіюласом і Ефремі́десом. У цьому підході кожен вузол мережі підтримує окрему чергу для кожного можливого призначення та передає пакети сусідові з найменшим диференціалом черги, тобто вздовж найбільшого градієнта «тиску черг» [15]. Така стратегія гарантує максимально можливу пропускну здатність (throughput) і математично доведено стабілізує мережу за будь-якого здійсненного навантаження [15].

Однак backpressure має і суттєві недоліки. Він характеризується високою складністю, оскільки кожен вузол має підтримувати множину черг, а також потенційно великими затримками, що виникають унаслідок «блукання» пакетів мережею або їх тривалого очікування у чергах, особливо за слабкого трафіку [15]. Сучасний огляд Ванга та ін. (2022) підкреслює, що хоча backpressure алгоритмічно є оптимальним за пропускною здатністю, він часто виявляється непрактичним без модифікацій через велику латентність [15]. Для пом’якшення цих недоліків запропоновано низку модифікацій: використання евристик найкоротших шляхів, застосування «тіньових черг» (shadow queues), а також введення вікових штрафів (age-based penalties), які стимулюють пакети не затримуватися надовго в мережі [15].

Головний висновок для адаптивної телеметрії: зворотний зв’язок і контроль витрати потоку — фундаментально важливі для мереж із обмеженими ресурсами, але їх необхідно збалансовувати із затримкою, інакше система втратить практичну

придатність. У прикладних IoT-системах шлюз може реалізувати спрощену форму backpressure — наприклад, моніторувати заповнення вихідних черг і сигналізувати сенсорам про зниження частоти передавання, або тимчасово буферизувати дані локально, коли вузькосмуговий канал перевантажений. Огляд Вогеля та ін. (2022) щодо самоадаптації у потокових системах показує, що динамічне буферування, масштабування операторів та навмисне проріджування навантаження (load shedding) застосовуються для дотримання вимог продуктивності за мінливих умов [16]. Ці техніки можна вважати високорівневими аналогами backpressure на прикладному рівні, адже вони адаптують темп надходження телеметрії або ступінь паралелізму залежно від поточного навантаження.

Вогель та ін. (2021) також показали можливість онлайн-адаптації паралельних потокових конвеєрів [17]. У їхній моделі потокова програма (аналог телеметричного аналітичного конвеєра) оснащується автономним керуючим модулем, який профілює виконання та динамічно перепланує структуру паралелізму — наприклад, змінюючи кількість потоків або етапів обробки в режимі виконання [17]. Мета полягає в тому, щоб підтримувати задані SLO (latency, throughput) без участі оператора. Це повністю відповідає вимогам адаптивних телеметричних шлюзів: у разі збільшення кількості сенсорів, деградації пропускної здатності або появи пікових навантажень система має самостійно перебудовуватися, перерозподіляючи обчислення, змінюючи формати даних чи внутрішню конфігурацію.

Результати Вогеля демонструють, що поєднання легкого профілювання на час виконання з чітко визначеною стратегією адаптації дозволяє отримати продуктивність, близьку до найкращої статичної конфігурації, при цьому зберігаючи здатність реагувати на зміни [17]. Принципово важливо, що у цій моделі розробник задає високорівневі обмеження та інваріанти, а не низькорівневі

механізми адаптації, що концептуально узгоджується з ідеологією FRP та алгебр політик: описується «що має бути досягнуто», а не «як саме це реалізувати» [17].

1.9 Практичні аспекти реалізації та інструментарій

Побудова працездатної адаптивної телеметричної системи на основі описаних вище концепцій вимагає залучення низки сучасних технологій, частина з яких сформувалася лише протягом останніх років. Одним із першочергових практичних завдань є ефективна серіалізація даних, що особливо критично для телеметрії у вузькосмугових каналах. Текстові формати, на кшталт JSON, є надто «баластними» з огляду на службові накладні дані, тоді як бінарні формати забезпечують значно компактнішу передачу.

В цьому контексті Concise Binary Object Representation (CBOR) (RFC 8949), розроблений IETF, спеціально оптимізовано для мінімального розміру повідомлень і малої апаратної складності реалізації, що робить його зручним для обмежених пристроїв [18]. Аналогічно, Protocol Buffers (ProtoBuf) від Google забезпечують платформно-нейтральний спосіб опису структурованих даних та їх компактної серіалізації — «як JSON, але менший і швидший» [19]. Використання CBOR або ProtoBuf у телеметрії дає змогу уникнути зайвого перевантаження каналу, що особливо важливо для технологій на кшталт LoRaWAN, де корисне навантаження може обмежуватися десятками байтів.

Адаптивний шлюз може навіть динамічно перемикати формат обміну, виявляючи обмеження пропускної здатності та переходячи з текстового формату на бінарний. У деяких сценаріях можливий початковий обмін JSON-повідомленнями з подальшою «узгодженою» комутацією на бінарний формат.

Інший ключовий блок – спостережуваність (observability) самої системи. Для моніторингу стану та продуктивності можна використовувати стек на основі Prometheus, де Prometheus JMX Exporter дозволяє збирати метрики JVM-додатків і експортувати їх у форматі для Prometheus [20]. Це дає змогу адаптивному шлюзу

публікувати внутрішні метрики, такі як довжина черг, завантаження CPU, частота обробки повідомлень тощо. В свою чергу, Grafana забезпечує зручні панелі для візуалізації часових рядів [21] — наприклад, для перегляду впливу адаптивного алгоритму QoS на затримку або відсоток втрат пакетів у реальному часі. Такі дашборди є важливим інструментом для налагодження адаптивної логіки.

Для тестування поведінки в мережевих умовах, наближених до реальних, застосовуються мережеві емулятори, зокрема tc netem, який дозволяє інжектувати затримки, втрати, обмеження пропускної здатності та інші артефакти [22]. Це критично важливо для валідації таких механізмів, як backpressure або адаптивне керування швидкістю: наприклад, можна відтворити 100 мс затримки та 5% втрат, перевіривши, чи зменшується швидкість вихідного потоку та чи не переповнюються буфери.

З програмного боку, побудова шлюзу у функційно-орієнтованому середовищі (FP) дає додаткові переваги. Scala володіє розвиненою екосистемою для таких завдань. Бібліотека Doobie забезпечує чисто-функційний доступ до баз даних через JDBC, що дозволяє інтегрувати зберігання телеметрії в загальний реактивний конвеєр, не порушуючи референційної прозорості [23]. Фреймворк ZIO пропонує модуль ZIO Metrics, який надає строго типізований і функційний спосіб роботи з метриками, що добре узгоджується з FRP-підходом [24]. Це дає змогу інструментувати саму адаптивну логіку (наприклад, вимірювати частоту адаптацій або час стабілізації системи після змін).

Для випадків, коли JSON усе ж необхідний (наприклад, для конфігурації системи або нефрактальних повідомлень), бібліотека Circe забезпечує чисто-функційний енкодер/декодер із автоматичним виведенням схем [25][86][87], що дозволяє легко поєднувати JSON з FRP та ZIO-екосистемою.

Таким чином, поєднання ефективних форматів даних, стеку спостережуваності, інструментів мережевої емуляції та функційних бібліотек

створює практичний і цілісний фундамент для реалізації теоретичних принципів адаптивної телеметрії, описаних у попередніх розділах.

У цьому огляді було розглянуто два взаємопов'язані аспекти побудови адаптивних телеметричних систем: (1) комунікаційні протоколи та стратегії адаптації шлюзів, орієнтовані на вузькосмугові мережі (NB-IoT, LoRaWAN), та (2) формальні моделі й парадигми програмування (функціонально-реактивне програмування та алгебри політик), які дають змогу специфікувати та верифікувати адаптивну поведінку системи.

На основі аналізу мережевих протоколів можна зробити висновок, що UDP-орієнтовані IoT-протоколи (CoAP, MQTT-SN) загалом є більш придатними для високообмежених мереж, демонструючи менші накладні витрати та кращу масштабованість порівняно з MQTT поверх TCP у сценаріях глибокого покриття [1]. Водночас повна відмова від гарантованої доставки в багатьох застосуваннях є неможливою — отже адаптивний шлюз має динамічно узгоджувати параметри протоколів (рівні QoS, частоту передавання, стратегії підтвердження), знаходячи компроміс між пропускною здатністю та надійністю. Як показують результати низки робіт, централізоване керування QoS для MQTT-SN суттєво підвищує ймовірність доставлення в умовах мінливих каналів [3], тоді як ADR у LoRaWAN демонструє важливість адаптації на фізичному рівні, хоча й має обмеження у випадках мобільності [4]. Наведена порівняльна таблиця підтвердила, що кожен протокол і кожна LPWAN-технологія мають власні сильні та слабкі сторони, а отже, адаптація часто передбачає комутацію або поєднання протоколів — завдання, для якого інтелектуальний шлюз є природним центром прийняття рішень.

З теоретичної точки зору, еволюція FRP та поява мережевих алгебр на кшталт NetKAT забезпечують потужні засоби для проєктування і формальної верифікації таких систем. FRP пропонує високорівневий апарат для безперервної реконфігурації системи у відповідь на події, що повністю відповідає вимогам

адаптивної телеметрії. Алгебри політик, зі свого боку, гарантують, що під час реконфігурації дотримуються ключові властивості безпеки та живучості системи (наприклад, відсутність дублювання або неправильного маршрутизаційного рішення). Мови та середовища, подібні до Frenetic, уже демонструють синтез цих методів, трактуючи мережеві події функційно [13]. Паралельно, дослідження механізмів самоадаптації потоків і алгоритмів зворотного тиску наголошують, що зворотний зв’язок і контроль навантаження залишаються фундаментально необхідними: система повинна вимірювати власний стан, реагувати на нього та стабільно закривати керувальний цикл. Важливо витримувати баланс: надто повільна адаптація призведе до накопичення проблем, тоді як надто агресивна викличе осциляції та деградацію.

Узагальнюючи, побудова адаптивного MQTT-шлюзу для NB-IoT/LoRaWAN є міждисциплінарним завданням, яке потребує поєднання ідей з інженерії систем, теоретичної інформатики та мережевих технологій. Література переконливо демонструє, що поєднання ефективних протоколів (орієнтованих на вузькі та зашумлені канали), реактивних моделей (FRP) і формальних методів (алгебри політик) дає змогу створювати телеметричні платформи, які не лише підтримують стабільну роботу в умовах динамічних змін, але й є формально зрозумілими та верифікованими. Подальші дослідження, ймовірно, рухатимуться у напрямі єдиних уніфікованих платформ, у яких розробник зможе оголошувати адаптивні політики на високому рівні (за допомогою NetKAT, FRP або логік на кшталт LTL/CTL), тоді як викональне середовище автоматично забезпечуватиме зворотній зв’язок, контроль QoS, реконфігурацію та оптимізацію протоколів у реальному часі. Такий підхід радикально спростить створення стійких IoT-систем, здатних до самонавчання, самооптимізації та надійної роботи навіть за умов обмежених ресурсів і непередбачуваних мережевих змін. Розглянуті в цьому огляді дослідження формують міцний фундамент на шляху до таких систем.

1.10 Постановка задачі дослідження

На підставі виконаного аналітичного огляду можна констатувати, що існуючі рішення для адаптивної передачі телеметрії у вузькосмугових IoT-мережах характеризуються низкою суттєвих обмежень. По-перше, більшість підходів орієнтовані на адаптацію окремих ізольованих параметрів (PHY, QoS або формат) без урахування їхніх взаємозалежностей. По-друге, відсутній формальний апарат, що дозволив би композиційно комбінувати адаптаційні примітиви та доводити інваріанти стабільності потоку. По-третє, рідко враховується зворотний зв'язок на основі SLO/SLI для динамічного коригування параметрів передачі.

Зважаючи на означені прогалини, у межах даного дослідження ставиться задача розробки адаптивного MQTT-шлюзу для вузькосмугових IoT-каналів (NB-IoT/LoRa), що буде задовольняти такі вимоги:

- Динамічна адаптація параметрів передачі: формат серіалізації (JSON → CBOR → Protobuf), розмір батчу, частота відправлення, рівень QoS;
- Формальне обгрунтування стабільності: інваріанти backpressure, обмежена черга, відсутність осциляцій параметрів;
- SLO-орієнтоване керування: підтримка цільових показників (p95-латентність ≤ 2–3 с, частка втрат ≤ 1;
- Інтеграція з MQTT v5: використання Receive Maximum, Maximum Packet Size, Message Expiry для керування потоком;
- Відтворювана експериментальна методика: емуляція мережевих умов через tc netem, автоматизація тестування.

Висновки до розділу

Аналітичний огляд підтвердив, що вузькі канали LPWAN (NB-IoT, LoRaWAN) висувають до телеметрії суперечливі вимоги: мінімізувати байти «на дроті» та час у ефірі (airtime), але водночас гарантувати прийнятну доставність і контроль затримок. На рівні прикладних протоколів сформувалася тріада підходів

— MQTT (TCP), MQTT-SN (UDP/без TCP) і CoAP (UDP/REST). Переваги MQTT — зріла брокерна модель, QoS 0–2 і широка підтримка в екосистемі; слабке місце — накладні й поведінка TCP на високих RTT/втратах. CoAP/UDP на NB-IoT зазвичай виграє при високих навантаженнях завдяки нижчим протокольним накладним та відсутності TCP-handshake, тоді як MQTT-SN радикально зменшує накладні MQTT і зручний для «сплячих» сенсорів, але потребує шлюзу-транслятора до MQTT-брокерів. Порівняння підтверджує тезу: єдиної «правильної» зв'язки немає, а ефективність визначається поєднанням властивостей каналу (RTT, втрати, duty-cycle) і трафіку (періодика, вибухові події, щільність змін полів). Це безпосередньо мотивує адаптивні шлюзи, здатні обирати формат, частоту, розмір і рівень надійності під поточний стан каналу, а не покладатися на статичні налаштування.

Важливий висновок огляду полягає в тому, що MQTT v5 сам по собі є якісним фундаментом для адаптації на прикладному рівні: він вводить механізми керування потоком (Receive Maximum), обмеження розміру пакета (Maximum Packet Size), скорочення заголовків (Topic Alias) і політику «свіжості» (Message Expiry). Це створює «операційні гаки», до яких можна прив'язати адаптивну політику, не змінюючи пристроїв та не руйнуючи брокерну інтеграцію. На практиці це дозволяє наблизити ефективність до UDP-стеку (CoAP/MQTT-SN) за рахунок компактних кодувань (CBOR/Protobuf), дельта-кодування і пакетування, зберігаючи при цьому силабіліті брокерної моделі та інструменти управління доставністю. Таким чином, адаптація на рівні додатка (encode/Δ/batch/throttle/QoS/expiry) з опорою на можливості MQTT v5 — домінуючий напрям, здатний врівноважити компроміс «ефективність ↔ надійність» для NB-IoT/LoRa.

Стандарти та польові практики LPWAN підкреслюють специфіку радіоканалу. Для LoRaWAN ключову роль відіграє ADR (Adaptive Data Rate), що регулює швидкість і потужність передачі; проте ADR передбачає квазістаціонарні

вузли, і за мобільності або динамічної радіообстановки його користь слабшає. Це означає, що надійний сервіс потребує додаткового прикладного контуру адаптації, який реагує швидше за лінк-рівень (наприклад, тимчасово зменшує частоту, змінює формат, підвищує QoS саме тоді, коли «поплив» канал). У NB-IoT «операторська» якість сервісу краща, однак RTT може сягати секунд; через це швидкі сплески вхідного трафіку легко переповнюють черги клієнта/брокера, якщо не впроваджено пейсинг та ліміти «in-flight». Отже, у двох провідних LPWAN технологіях (ліцензований NB-IoT і безліцензійний LoRaWAN) загальний знаменник — потреба в контролі темпу, розміру й формату телеметрії на прикладному рівні.

Огляд міждисциплінарних джерел (FRP, алгебри політик, NetKAT, Stream Fusion) показав, що функціональний підхід із формальними законами — найпростіший шлях зробити адаптацію аналізованою і перевірюваною. Політика як чиста композиція операторів над потоками (Encode/Δ/Batch/Throttle/QoS/Timeout) дозволяє еквівалентні переписування, мінімізує прихований стан і робить валідацію властивостей (стабільність черг, відсутність осциляцій, монотонне відновлення) прозорою. Ці знахідки безпосередньо підхоплено в архітектурі реалізації: шлюз інтерпретує політику як стрім-перетворення поверх ZIO Streams із прив’язкою до властивостей MQTT v5. Такий «місток» між теорією і залізом — ключова перевага обраного напряму.

Висновок розділу: література однозначно вказує, що адаптивний прикладний шар є необхідністю для вузьких каналів зв’язку, а не «опцією». Поєднання MQTT v5 (як операційної платформи адаптації) з функціональною алгеброю політик і FRP-сигналами забезпечує одночасно (а) практичну реалізованість у типовому брокерному стеку та (б) формальну мислимість і можливість доведень. Цей висновок задав рамку для теоретичного формалізму і для розробки проєкту, де політика узгоджена із реальними механізмами протоколу (Receive Maximum, Topic

Alias, Message Expiry, Max Packet Size), а експериментальні сценарії (tc netem, контейнерні стенди) дають відтворювані оцінки ефектів кожної «ручки» адаптації.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ЗАСАДИ АДАПТИВНОЇ ТЕЛЕМЕТРІЇ

2.1 Модель адаптивної телеметрії: SLO, SLI та змінні керування

Формалізація задачі адаптивної передачі телеметрії потребує чіткого визначення цільових показників (Service Level Objectives, SLO), індикаторів якості обслуговування (Service Level Indicators, SLI) та керуючих змінних. Така модель дозволяє формулювати адаптацію як задачу оптимізації або керування зі зворотним зв'язком.

Service Level Objectives (SLO) — це цільові значення продуктивності, які система повинна підтримувати для забезпечення задовільної якості обслуговування. У контексті IoT-телеметрії типовими SLO є:

- p95-латентність доставки ≤ L_target (наприклад, 2–3 с);
- частка втрачених повідомлень ≤ Loss_target (наприклад, 1%);
- зменшення обсягу трафіку на 30–60%;
- стабільність потоку (відсутність переповнення черг).

Сервісні цілі (SLO) задаємо в термінах телеметрії для вузьких каналів:

- затримка $L_{p95} \leq L^*$ (мс);
- частка втрат $\rho \leq \rho^*$;
- середні байти/повідомлення $B \leq \beta^*$.

Service Level Indicators (SLI) — це фактично виміряні метрики, що відображають поточний стан системи:

RTT (Round-Trip Time) — час проходження пакета туди і назад;

- Loss rate — частка втрачених повідомлень у ковзному вікні;
- Queue depth (Q) — поточна глибина черги повідомлень;
- Bytes/msg — середній розмір відправленого повідомлення.

Показники рівня сервісу (SLI) спостерігаємо онлайн:

$$S_t = \left(\widehat{RTT_t}, \widehat{loss_t}, q_t, \widehat{B_t}\right) \quad (2.1)$$

Де $\widehat{\mathrm{RTT}_t}, \widehat{\mathrm{loss}_t}$ згладжені оцінки (EWMA), q_t — глибина черги шлюзу, $\widehat{B_t}$ — оцінка розміру корисного навантаження. Саме ця комбінація SLI використовується як “маловимірний стан” для керування політикою.

Вектор керування (політика):

$$u_t = (r_t,\ w_t,\ W_t,\ k_t,\ \mathrm{fmt}_t) \tag{2.2}$$

r — цільова швидкість (пов/с), w — вікно пакетування (мс), W — ліміт одночасно “в польоті” (MQTT v5 Receive Maximum), $k \in \{0,1\}$— рівень QoS, $\mathrm{fmt} \in$ {JSON,CBOR,Proto} — формат кодування. Такі керуючі “ручки” безпосередньо проєктуються у властивості MQTT v5: Receive Maximum, Maximum Packet Size, Topic Alias, Message Expiry.

Змінні керування (control variables) визначають параметри, якими система може оперувати для досягнення SLO:

- Формат серіалізації (enc): JSON → CBOR → Protobuf;
- Розмір батчу (batch_size): 1, 5, 10, …;
- Частота відправлення (rate): мінімальний інтервал між повідомленнями;
- Рівень QoS: 0, 1, 2 (для MQTT).

Обмеження каналу (channel constraints) визначають межі, яких система повинна дотримуватися:

- Для NB-IoT: пропускна здатність ~20–60 kbps, затримка 1–10 с;
- Для LoRaWAN: SF7–SF12, duty-cycle 1% (EU868), ToA до кількох секунд на повідомлення;
- Максимальний розмір пакета: 51–222 байти залежно від SF.

Задача адаптації формулюється як знаходження оптимальних значень керуючих змінних, що мінімізують відхилення SLI від SLO за дотримання обмежень каналу. Формально: min |SLI – SLO| subject to channel constraints.

Модель каналу і сервісної спроможності. Для QoS1 з ACK-обмеженням верхня межа стійкої швидкості публікацій:

$$r_{max} \approx \frac{W}{RTT} \quad \text{(обмеження ACK-вікном)} \tag{2.3}$$

тож необхідна умова стабільності черги шлюзу:

$$r_t < \mu_t, \qquad \mu_t \approx \frac{W_t}{\widehat{RTT_t}} \tag{2.4}$$

Динаміка черги (флюїдна апраксимація):

$$q_{t+1} = \max\{0,\ q_t + r_t - \mu_t\} \tag{2.5}$$

Для LoRa з обмеженням duty-cycle δ маємо додаткову верхню межу по ефективній пропускній здатності $R_{eff} \leq \delta \cdot R_{PHY}$ та вимогу B мінімізувати (через fmt Δ batch) щоб вкладатися в Time-on-Air. Ці інваріанти використовуємо в керуванні.

2.2 Функціональна алгебра політик адаптації

Центральним елементом теоретичної моделі є функціональна алгебра політик адаптації. Політика — це чиста функція, що трансформує вхідний потік повідомлень у вихідний з урахуванням поточного стану каналу. Математично: Policy : Stream[A] → (Signal[State]) → Stream[B], де Stream — послідовність повідомлень, Signal — часозалежне значення стану.

Розглядаємо політику як чистий перетворювач потоку:

$$\mathcal{P}\colon \text{Stream}[I] \rightarrow \text{Stream}[O] \tag{2.6}$$

Базовий набір тотальних комбінаторів:

- Encode(fmt): JSON↔CBOR/Proto; монотонно (за розміром) для компактних форматів;
- Δ(diff): дельта-кодування/супресія незмінних полів (із періодичним “snapshot refresh”);
- Batch(w, agg): збір за вікном w із асоціативним агрегатором (типово last-value semantics) ;
- Throttle(r, W): темпування ≤r пов/с із урахуванням обмеження в польоті W (MQTT Receive Maximum) ;

- QoS(k): вибір QoS0/1 із ретраями та Message Expiry;
- Guard(φ): умовний вибір підполітики за предикатом (напр., дефіцит SLO);
- Choose(P1,…,Pn): детермінований пріоритетний вибір серед Guard-політик;
- Timeout(θ): відсіювання за віком (s.t. freshness-SLO).

Композиція — звичайна (∘) та умовна (▹) композиції. Денотаційна семантика — перетворювач трас, операційні прив’язки — до конкретних механізмів MQTT v5 (Receive Maximum, Topic Alias, Max Packet Size, Message Expiry), що гарантує відповідність доведених властивостей реальному протоколу

Базові оператори алгебри політик визначаються наступним чином:

Encode(format) — оператор перетворення формату серіалізації повідомлення. Денотаційна семантика: [[Encode(f)]](m) = serialize(m, f). Оператор приймає повідомлення m і серіалізує його у формат f (JSON, CBOR, Protobuf). Застосування оператора зменшує розмір повідомлення залежно від формату: CBOR скорочує JSON приблизно на 30–40%, Protobuf — на 50–70%.

Delta(prev) — оператор дельта-кодування, що передає лише зміни відносно попереднього стану. Денотаційна семантика: [[Delta]](m, prev) = diff(m, prev). Ефективний для повільно змінюваних телеметричних сигналів; може зменшити розмір на 60–90% за умов стаціонарності даних.

Batch(n) — оператор агрегації, що об'єднує n послідовних повідомлень в одне. Денотаційна семантика: [[Batch(n)]](stream) = stream.grouped(n).map(concat). Агрегація зменшує кількість пакетів та амортизує накладні витрати заголовків.

Throttle(minInterval) — оператор темпорегуляції, що обмежує мінімальний інтервал між повідомленнями. Денотаційна семантика: [[Throttle(d)]](stream) = stream.zipWithTime.filter((_, t) => t – prev ≥ d). Запобігає перевантаженню каналу за високої частоти генерації даних.

QoS(level) — оператор вибору рівня якості обслуговування MQTT. Денотаційна семантика: [[QoS(q)]](m) = m.withQoS(q). Вищі рівні (QoS 1, 2) забезпечують надійнішу доставку, але збільшують накладні витрати та латентність.

Guard(predicate) — умовний оператор, що пропускає повідомлення лише за виконання предикату. Денотаційна семантика: [[Guard(p)]](m) = if p(m) then Some(m) else None. Використовується для фільтрації на основі стану каналу або вмісту повідомлення.

Choose(cond, p1, p2) — оператор умовного вибору політики. Денотаційна семантика: [[Choose(c, p1, p2)]](m, s) = if c(s) then p1(m, s) else p2(m, s). Дозволяє динамічно перемикати між політиками залежно від стану каналу.

Timeout(duration) — оператор таймауту, що скасовує операцію, яка не завершилася протягом заданого часу. Денотаційна семантика: [[Timeout(d)]](effect) = race(effect, fail after d). Критичний для забезпечення відгуку у каналах з непередбачуваною латентністю.

Композиція політик здійснюється через послідовне застосування (андфор): (p1 andThen p2)(m, s) = p2(p1(m, s), s). Типова комбінована політика: Encode(CBOR) andThen Delta andThen Batch(10) andThen Throttle(1s) andThen QoS(1).

Алгебраїчні закони операторів визначають їхні властивості та можливі оптимізації:

- Ідемпотентність: Encode(f) ∘ Encode(f) = Encode(f);
- Поглинання: Encode(f1) ∘ Encode(f2) = Encode(f2);
- Комутативність (Batch, Encode): Batch(n) ∘ Encode(f) = Encode(f) ∘ Batch(n);
- Асоціативність: (p1 ∘ p2) ∘ p3 = p1 ∘ (p2 ∘ p3);
- Монотонність Throttle: Throttle(d1) ∘ Throttle(d2) = Throttle(max(d1, d2)).

Важливі алгебраїчні закони:

- Ідемпотентність батчінгу: Batch(w)∘Batch(w′)=Batch(max{w,w′});

– Умовна комутативність: за компактним кодуванням і асоціативним agg: Batch∘Encode=Encode∘Batch;
– Монотонність обмежень: посилення (w↑,r↓,W↓) не збільшує пік q і середнє “на дроті”.

Таке представлення політик як вільної алгебри дозволяє доводити еквівалентності переписувань (оптимізацій) і гарантувати відсутність прихованого стану (референтна прозорість), що суттєво спрощує формальну перевірку. Інтерпретатор цієї алгебри може бути реалізований як природна трансформація у стрім-семантику (напр., ZIO Streams), зберігаючи причинність і backpressure.

2.3 FRP-модель оцінювання стану каналу

Функціональне реактивне програмування (Functional Reactive Programming, FRP) забезпечує модель, у якій стан каналу представлено як часозалежний сигнал. Сигнали (Signal[T]) — це функції часу: Signal[T] : Time → T, що моделюють безперервні величини (наприклад, оцінка RTT, рівень втрат).

Оцінки метрик каналу (RTT, втрати, черга, байти/повідомлення) представляємо як часозмінні сигнали FRP; політика — як сигнальні функції над ними з явним “перемиканням” гілок (Guard/Choose). Для згладження використовуємо EWMA:

$$\widehat{x_t} = \alpha\, x_t + (1-\alpha)\,\widehat{x_{t-1}}, \qquad \alpha \in (0,1), \tag{2.7}$$

що забезпечує інваріант стабільності до коротких сплесків шуму. Така FRP-модель узгоджується з push-pull виконанням (мінімізує накладні витрати реактивної системи на вузьких каналах).

Оцінювання стану каналу здійснюється через експоненційно зважене ковзне середнє (Exponentially Weighted Moving Average, EWMA). Для сигналу x(t) з дискретними спостереженнями x_k оцінка обчислюється по формулі (2.7)
де $\alpha \in (0,1)$ — коефіцієнт згладжування. Менші значення α забезпечують більше згладжування та стійкість до шуму, більші — швидшу реакцію на зміни.

Основні сигнали стану каналу:

- RTT(t) — оцінка часу проходження пакета;
- Loss(t) — оцінка частки втрат;
- Q(t) — глибина черги повідомлень;
- BytesPerMsg(t) — середній розмір повідомлення.

Сигнальні функції (signal functions) трансформують сигнали: SF : Signal[A] → Signal[B]. У контексті адаптації сигнальна функція ChannelEstimator перетворює потік подій (ACK, timeout, queue update) у сигнал стану State = (RTT, Loss, Q, BytesPerMsg).

2.4 SLO-орієнтоване керування з контрактними оновленнями

SLO-орієнтоване керування базується на порівнянні поточних SLI з цільовими SLO та коригуванні параметрів політики для мінімізації відхилення. Ключовим є забезпечення стабільності керування — уникнення осциляцій параметрів та переповнення черги.

Механізм контрактних оновлень (contractive updates) гарантує збіжність послідовності параметрів до стабільного стану.

Вектор керування оновлюємо за контрактним відображенням (перша різниця до цілі):

$$u_{t+1} = u_t + \lambda\,(u^* - u_t), \qquad 0 < \lambda < 1, \tag{2.8}$$

де ціль $u^*(S_t)$ формується евристиками з дефіциту SLO (див. нижче). Це еквівалент дискретного НЧ-фільтра першого порядку; при $0 < \lambda < 1$ відображення є стискаючим, тож збігається до фікс-точки (теорема Банаха), гарантується без-осциляційність політики.

За теоремою Банаха про нерухому точку, якщо оператор оновлення є стискаючим ($||T(x) – T(y)|| \leq \lambda||x – y||$ для $\lambda < 1$), то послідовність {p_n} збігається до єдиної нерухомої точки p*, що відповідає стабільному стану системи.

Алгоритм оновлення політики виконується періодично (наприклад, кожні 10 секунд) або за подією (перевищення порогу черги). Псевдокод алгоритму:

Алгоритм 1 (псевдокод).

Input: SLO = (L*, ρ*, β*), λ∈(0,1), u_t=(r,w,W,k,fmt), S_t=(RTT,loss,q,B)

Target:

```
 if loss>ρ* or RTT>L*:
    r* := max(0.1, 0.75·r)      ; w* := min(w·1.5, 10 s)
    W* := 1                     ; k* := 1
 else:
    r* := min(20, 1.05·r)       ; w* := max(50 ms, 0.9·w)
    W* := min(W+1, Wmax)        ; k* := 0
 if B>β*: fmt* := CBOR/Proto; enable Δ
```

Update (contractive):

u_{t+1} := u_t + λ ((r*,w*,W*,k*,fmt*) − u_t)

Цей шаблон відтворює закодовану в реалізації практику “обережних кроків” з одночасним обмеженням у польоті W (MQTT v5 Receive Maximum) та ротацією формату в бік компактного кодування.

2.5 Формули ефективності: розмір, частота, надійність

Очікуваний розмір повідомлення

Нехай B_{fmt} середній розмір у форматі fmt, а $\pi_t \in [0,1]$ — частка полів, що змінились (для Δ-кодування). Тоді

$$E[B_t] \approx \underbrace{B_{\text{base}}(\text{fmt})}_{\text{формат}} + \pi_t \cdot \underbrace{B_{\Delta}(\text{fmt})}_{\text{дельта}} \tag{2.9}$$

Причому $B_{\text{Proto}} \lesssim B_{\text{CBOR}} \ll B_{\text{JSON}}$ за типовими IoT-навантаженнями; зміщення з JSON у CBOR/Proto дає найбільший виграш у β^* та LoRa-airtime.

Ефективний темп і батчінг

За політикою Batch(w,last) та Throttle(r):

$$r_{\text{eff}} \le \min\left(r,\ \frac{1}{w}\right), \qquad B_{\text{eff}} \approx B_1 + \frac{H}{n} \tag{2.10}$$

де H — протокольні накладні (заголовки), n — середня кількість елементів у блоці. Батчінг зменшує “байти на елемент”, але додає затримку w; правило підбирає w з огляду на L^*.

Закон ідемпотентності батчінгу дозволяє безпечно “укрупнювати” w, не дублюючи ефект.

QoS та свіжість:

Для QoS1 із ретраями та Message Expiry = θ:

$$P\{\text{доставлено й свіже}\} \ \approx\ \left(1 - \widehat{\text{loss}}\right)^{N_{\text{retry}}} \ \cdot\ 1\{\text{age} \le \theta\} \tag{2.11}$$

що задає компроміс: при поганому каналі $k \uparrow$ але водночас варто обмежувати “старіння” даних (Timeout/Expiry).

2.6 Інваріанти стабільності потоку

Формальне обґрунтування стабільності адаптивної системи базується на кількох ключових інваріантах, що гарантують коректну роботу за будь-яких умов.

Інваріант 1 (Обмежена черга): За наявності backpressure-керування глибина черги Q(t) обмежена зверху константою Q_max: ∀t: Q(t) ≤ Q_max. Доведення базується на функції Ляпунова V(Q) = Q² та аналізі Ляпунівського дрифту: E[V(Q_{n+1}) – V(Q_n) | Q_n = q] ≤ –ε для q > Q_max, що гарантує повернення черги до безпечного рівня (якщо $r_t < \mu_t$ та діють зменшувальні реакції ($r \downarrow, w \uparrow$) на порушення SLO, то $\sup_t q_t \le Q_{max}$ дрифт Ляпунова зі штрафом за q).

Інваріант 2 (Відсутність осциляцій): Параметри політики p_n збігаються до нерухомої точки без осциляцій, якщо функція оновлення є ліпшицевою з константою L < 1: |p_{n+1} – p_n| → 0 при n → ∞. Це випливає з властивості стискаючих відображень.

Інваріант 3 (Backpressure preservation): Швидкість виходу з черги не перевищує швидкості входу нижчого за потоком компонента: rate_out ≤

rate_downstream. Це забезпечується через механізми MQTT v5 (Receive Maximum) та власну логіку throttling (у присутності Receive Maximum/ACK- pacing темп ретрансляції не перевищує вузького місця; адаптація зберігає backpressure-структуру).

Інваріант 4 (Монотонне відновлення): Після періоду деградації (високий RTT або втрати) система монотонно відновлює нормальний режим роботи: якщо умови каналу покращуються, SLI монотонно наближаються до SLO (у присутності Receive Maximum/ACK- pacing темп ретрансляції не перевищує вузького місця; адаптація зберігає backpressure-структуру).

2.7 Відображення моделі в механізми MQTT v5

Протокол MQTT версії 5.0 запроваджує низку механізмів керування потоком, що безпосередньо відображаються на елементи запропонованої моделі:

Receive Maximum — властивість CONNACK, що визначає максимальну кількість непідтверджених PUBLISH-повідомлень (з QoS > 0), які отримувач готовий обробляти одночасно. Це пряма реалізація backpressure на рівні протоколу. Шлюз повинен відстежувати кількість in-flight повідомлень і призупиняти відправку при досягненні ліміту.

Maximum Packet Size — обмеження на максимальний розмір MQTT-пакета, що впливає на параметри Batch та Encode. Шлюз адаптує розмір батчу так, щоб результуючий пакет не перевищував ліміт: batch_size = min(n, floor(maxPacketSize / avgMsgSize)).

Topic Alias — механізм стиснення імен топіків для зменшення накладних витрат заголовка. Особливо ефективний для довгих ієрархічних імен топіків (наприклад, sensor/building1/floor3/room42/temperature).

Message Expiry Interval — час життя повідомлення у брокері. Дозволяє автоматично видаляти застарілу телеметрію, що критично для обмежених каналів з високою латентністю.

– Throttle/Receive Maximum. Пряме обмеження W і темпування r накладають верхню межу на $r_{max} \approx W/\widehat{\mathrm{RTT}}$, що запобігає переповненню при великих RTT NB-IoT/LoRa;
– Maximum Packet Size / Encode. Контроль $\hat{B}$ через CBOR/Proto/Δ/Batch дозволяє триматися нижче ліміту пакета MQTT v5 та LoRa ToA;
– Topic Alias (LRU-кеш). Алгоритм LRU на $\leq$ TopicAliasMax зменшує байти заголовка: економія $\approx (|\mathrm{topic}| - 2)$ байти/повідомлення при хіті, з амортизованою складністю *O(1)*;
– Message Expiry/Timeout. Гарантує SLO на "свіжість" без лавинних ретраїв. Усі механізми включено до операційної моделі політики та реалізації інтерпретатора.

2.8 Метод адаптації як задача оптимізації

Зручний погляд — онлайн-мінімізація штрафної функції за дефіцитом SLO:

$$J_t(u) = \alpha_L\left[L_{p95}(u, S_t) - L^*\right]_+ + \alpha_\rho[\rho(u, S_t) - \rho^*]_+ + \alpha_B[B(u, S_t) - \beta^*]_+ \quad (2.12)$$

де $[x]_+ = \max\{x, 0\}$. Оновлення — це робастний варіант градієнт-вільного кроку у бік $u^* = \arg\min J_t$ з контрактним релаксаційним фактором λ. Така схема добре працює під шумними SLI й невизначеністю каналу.

Ключові алгоритми (узагальнені схеми):

Алгоритм 2 — Обчислення SLI (RTT/loss/queue/bytes)

– інструментуємо publish start/ack time → *RTT*;
– підраховуємо успіх/помилку → *loss*;
– спостерігаємо глибину черги q та середній розмір$\hat{B}$
– оновлюємо EWMA з $\alpha \in [0.05, 0.2]$

Алг. 3 — Оновлення політики Виконує Алг. 1, логуючи зміни політики, щоб забезпечити аудит та подальше офлайн-налаштування параметрів ($\alpha, \lambda,$ пороги)

Алгоритм 4 — Конструювання конвеєра політики Pipeline ≔ Throttle(r) ≫ Δ ≫ Batch$(w$,last) ≫ Encode(fmt) ≫ QoS(k) ≫ Timeout(θ), де ≫ — композиція комбінаторів; з огляду на закони можливі безпечні переписування (наприклад, пересування Encode ліворуч від Batch).

2.9 Перевірка коректності: властивості та тести

- Контрактність: $|u_{t+1} - u^*| \leq (1-\lambda)|u_t - u^*| \rightarrow$ збіжність;
- Стабільність черг: проперті-тест: за фіксованої $\widehat{\mathrm{RTT}}$ та W при зменшенні r у відповідь на $\rho \uparrow$ або $q \uparrow$ q_t обмежене; порушення → спрацьовує Guard→Timeout;
- Закони політики: ідемпотентність Batch, монотонність при посиленні обмежень;
- Експозиція метрик: опубліковуються SLI та керуючі параметри (для валідації SLO-дрифту та регресійних тестів);

2.10 Теоретичні підсумки й практичні наслідки:

- Алгебра політик дає формально коректну, придатну до переписування модель адаптації (Encode/Δ/Batch/Throttle/QoS/Timeout), узгоджену з операційними примітивами MQTT v5. Це спрощує доведення законів, оптимізацій та аудиту;
- FRP-сигнали забезпечують реактивність без колбек-хаосу; EWMA та контрактне оновлення політики гарантують відсутність осциляцій;
- Зворотній зв’язок за SLI (RTT, втрати, черга, байти) приводить політику до допустимої області $r < W/\widehat{\mathrm{RTT}}$, дотримуючи β^* за рахунок компактного кодування та дельт;
- Інтеграція з MQTT v5 (Receive Maximum, Topic Alias, Max Packet Size, Message Expiry) — обов’язкова для NB-IoT/LoRa, адже саме вона матеріалізує обмеження/гарантії каналу у прикладному шарі.

Примітка щодо артефактів

Описані тут моделі, алгоритми та інваріанти безпосередньо відображені у проєкті шлюзу (Scala/ZIO, політичний інтерпретатор, підсистема метрик, персистенція), включно з кодовими шаблонами політики, адаптаційним циклом і тестами стабільності/властивостей. Це зафіксовано у “Final Technical Report” та розширеному теоретичному документі.

Що саме потрібно для впровадження:

- Задати SLO, підключити SLI-інструментацію (RTT/loss/queue/bytes) та EWMA;
- Реалізувати політичний конвеєр як композицію чистих комбінаторів (Encode/Δ/Batch/Throttle/QoS/Timeout) з доведеними законами;
- Запустити контрактний контролер $u_{t+1} = u_t + \lambda(u^* - u_t)$ з Guard/Choose і Message Expiry;
- Прив’язати керування до MQTT v5 (Receive Maximum/Max Packet Size/Topic Alias/Expiry).

Ця теоретична база забезпечує стійкість, ефективність і формальну коректність адаптивного MQTT-шлюзу для NB-IoT/LoRa, і безпосередньо переноситься у виробничу реалізацію.

Висновки до розділу

Теоретична частина сформувала цілісну модель адаптивної телеметрії: (1) алгебра політик як вільна композиція чистих операторів над потоками; (2) FRP-сигнали для оцінювання стану каналу (EWMA-RTT, втрата, черга, байти/повідомлення); (3) контрактний зворотний зв’язок, який гарантовано збігається без осциляцій і утримує систему в області стабільності; (4) операційне відображення в можливості MQTT v5 і обмеження LPWAN. Разом це становить денотаційно-операційну (семантично строгішу за «евристики») основу для побудови систем, що мають довідні властивості. Зокрема, доведено (ескізами)

інваріанти: обмежені черги (Lyapunov drift), відсутність осциляцій (Banach fixed-point для контрактного відображення), домінування вузького місця даунстріму (sample-path coupling), монотонне відновлення після деградацій. Ці інваріанти описують «рамку безпеки» політики: якщо канал погіршується — темп зменшується, вікно пакетування збільшується, «in-flight» стискається до 1, QoS підвищується; якщо канал покращується — параметри розслабляються з малими кроками.

Ключова формула практичного значення — межа стійкого темпу на QoS1:

$$r_{max} \approx \frac{W}{RTT} \tag{2.13}$$

Вона унаочнює, що керуванням «in-flight» (Receive Maximum) і темпом (Throttle) можна нав’язати предиктивну верхню межу навантаження, сумісну з багатосекундними RTT NB-IoT/LoRa. В алгебрі це відповідає комбінації Throttle(r) і QoS-оператора з ACK-пейсингом; у стрім-реалізації — це регулярний інтервал емісії і семафорний доступ до публікацій. Формально це інтегрується в умову стабільності черг

$$r_t < \mu_t, \quad \text{де} \quad \mu_t \approx \frac{W_t}{\widehat{RTT}_t} \tag{2.14}$$

«контрактні» кроки на $\lambda \in (0,1)$ гарантують збіжність до цілі без перетягування і «пилоподібних» коливань. Таким чином, зворотній зв’язок SLO→SLI→політика має і практичну, і теоретичну коректність.

Алгебра політик дала змогу формально встановити корисні закони переписування: ідемпотентність пакетування (Batch ∘ Batch = Batch із більшим вікном), умовну комутативність з кодуванням (Batch ∘ Encode = Encode ∘ Batch за монотонних кодувань і асоціативного агрегаційного оператора), монотонність під посиленням обмежень (збільшення w/зменшення r/зменшення W не збільшують пікові черги і байти на дроті). Ці закони одночасно спрощують доказовість та надають практичні «важелі» оптимізації: наприклад, безпечно пересувати Encode вгору або вниз по конвеєру, не змінюючи семантики, але змінюючи ефективність

(менше угрупувань дрібних JSON-документів, більше — компактних CBOR/Proto). Чистота операторів (без побічних ефектів) критична: це витісняє прихований стан на межі з протоколом (MQTT), роблячи політику прозорою для аналізу і тестів.

FRP-інтерпретація забезпечила реактивність без колбек-хаосу: оцінки RTT/втрат/черги подані як сигнали, а політика — як сигнальні функції з умовним перемиканням (Guard/Choose). EWMA-згладжування усуває чутливість до коротких сплесків, зберігаючи агільність при тривалих змінах. Важливо, що денотація узгоджена з операційною семантикою MQTT v5: Throttle має повагу до Receive Maximum і PUBACK-пейсингу; Encode враховує Maximum Packet Size; Timeout і QoS реалізують політику свіжості та повторів, а Topic Alias мінімізує заголовкові накладні. Ця відповідність доводить, що «теорія не висить у повітрі»: кожен закон і інваріант має «залізний» сенс у протоколі.

Додатковий погляд — онлайн-оптимізація з функцією штрафу за дефіцит SLO. У цій рамці оновлення

$$u_{\{t+1\}} = u_t + \lambda(u^* - u_t) \tag{2.15}$$

— градієнт-вільний крок до $\arg \min J_{t(u)}$ із гарантованою стабільністю. Це пояснює, чому навіть грубі евристики (зменшити r при зростанні втрат/RTT; збільшити w при накопиченні черги; увімкнути CBOR/Proto і Δ при «роздуванні» байтів) демонструють добранісну поведінку — бо структуровані контрактним фільтром. Сукупно теоретичний розділ доводить: обраний підхід здатний одночасно (а) узгоджуватися з реальними обмеженнями LPWAN, (б) давати алгоритмічно прості, але формально контрольовані правила адаптації, (в) забезпечувати підстави для механізованої перевірки (моделі інваріантів, властивості конвеєра). Це робить його придатним для промислових впроваджень і для науково-дослідних підтверджень (property-based тести, TLA+-моделювання).

Підсумовуючи: теоретична база не лише описує, а й обмежує систему у корисний спосіб — через інваріанти, закони і контрактність. Вона дала інженерну

специфікацію політики, незалежну від конкретної реалізації, але повністю сумісну з MQTT v5. Саме тому подальша практична реалізація змогла відтворити ці властивості «на дроті», а не лише на папері.

У межах теоретичної частини розроблено формальну модель адаптивної телеметрії для вузькосмугових IoT-каналів.

Запропоновано модель SLO/SLI, що формалізує цільові показники та метрики якості обслуговування. Визначено керуючі змінні (формат серіалізації, розмір батчу, частота, QoS) та обмеження каналів NB-IoT і LoRaWAN.

Розроблено функціональну алгебру політик адаптації з базовими операторами (Encode, Delta, Batch, Throttle, QoS, Guard, Choose, Timeout) та їхньою денотаційною семантикою. Встановлено алгебраїчні закони (ідемпотентність, поглинання, комутативність, асоціативність), що дозволяють оптимізувати композиції політик.

Побудовано FRP-модель оцінювання стану каналу на основі EWMA-сигналів. Запропоновано механізм контрактних оновлень для SLO-орієнтованого керування, що гарантує збіжність параметрів до стабільного стану.

Сформульовано та обґрунтовано інваріанти стабільності: обмежена черга, відсутність осциляцій, backpressure preservation, монотонне відновлення. Показано відображення моделі на механізми MQTT v5 (Receive Maximum, Maximum Packet Size, Topic Alias, Message Expiry).

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ АДАПТИВНОГО MQTT-ШЛЮЗУ

3.1 Вибір програмного середовища та технологічного стеку

Проєкт реалізує адаптивний MQTT-шлюз як окремий прикладний компонент, що розташовується між пристроями інтернету речей і брокером повідомлень. Для реалізації серверної частини було обрано середовище виконання на основі віртуальної машини Java та сучасний стек для розроблення реактивних мережевих застосунків. Збірка й запуск шлюзу здійснюються за допомогою системи збірки sbt, що спрощує керування залежностями, організацію модулів і відлагодження під час експериментів.

Як брокер MQTT використовується поширена відкрита реалізація Mosquitto, розгорнута у вигляді окремого контейнера. Шлюз під'єднується до брокера через стандартний інтерфейс MQTT-v5 без жодних змін у конфігурації брокера, що забезпечує сумісність із наявною інфраструктурою користувача. Для збору метрик та автоматизованих SLO-перевірок застосовується стек моніторингу на основі Prometheus та Grafana: шлюз експонує HTTP-інтерфейс `/metrics`, з якого Prometheus періодично знімає показники, а Grafana використовує їх для побудови дашбордів.

Інфраструктура запуску усіх компонентів (Mosquitto, шлюз, Prometheus, Grafana, допоміжні сервіси) організована за допомогою Docker Compose. Це дає змогу відтворювати однакові умови експериментів на різних машинах, швидко перезапускати середовище для окремих прогонів без адаптації та з адаптацією політик, а також спрощує інтеграцію з мережевим емулятором, який керує параметрами каналу всередині контейнера брокера.

Обраний технологічний стек відповідає вимогам до системи з підвищеною надійністю та вимірюваністю: віртуальна машина Java й реактивні бібліотеки забезпечують передбачувану керованість потоків, засоби телеметрії для Prometheus

інтегровані безпосередньо в код шлюзу, а використання стандартного MQTT-брокера дозволяє легко переносити рішення на промислові інсталяції. Окремий контейнер для шлюзу ізольовує експериментальні навантаження від інших застосунків і спрощує розгортання в існуючих хмарних середовищах.

Таблиця 3.1 — Технологічний стек

Компонент	Технологія	Призначення
Мова програмування	Scala 2.13	Функціональна типізація, immutable структури
Ефекти та потоки	ZIO 2.x, ZStream	Backpressure, структурована конкурентність
MQTT-клієнт	Eclipse Paho (Java)	MQTT v5 підтримка
Серіалізація	Circe, CBOR-Java, Protobuf	JSON, CBOR, Protocol Buffers
Персистенція	PostgreSQL + Doobie	Зберігання політик, аудит
Метрики	Prometheus + Grafana	Моніторинг, візуалізація
Конфігурація	PureConfig + HOCON	Типобезпечний конфіг
Контейнеризація	Docker Compose	Розгортання, тестування

3.2 Структура бази даних

Для забезпечення відтворюваності експериментів та подальшого аналізу поведінки системи в різних умовах каналу необхідно зберігати як сирі телеметричні

дані, так і похідні характеристики. Логічна структура підсистеми персистентності містить такі основні групи даних:

- телеметричні записи від пристроїв (час надсилання, значення сенсорів, ідентифікатор пристрою, допоміжні поля);
- агреговані метрики якості обслуговування (середня та перцентильна затримка, втрати, глибина черг, інтенсивність трафіку);
- стан політики адаптації (поточна швидкість надсилання, параметри пакетування, гранична кількість одночасних повідомлень, рівень надійності, обраний формат кодування);
- журнали подій та експериментальних сегментів (початок/завершення тестів, сценарії мережевих умов, анотації A/B-прогонів).

У межах даного розділу розглядається узагальнена логічна структура сховища без конкретної прив’язки до типу СУБД. Під час практичного розгортання таблиці можуть реалізовуватися як реляційні структури з явними індексами за часом і пристроєм або як колекції документно-орієнтованого сховища з гнучкою схемою. Приклад опису колекції (або таблиці) для зберігання телеметричних записів наведено в табл. 3.2.

Таблиця 3.2 – Опис таблиці (колекції) зберігання телеметричних записів

Назва стовпця	Опис	Тип даних	Ключі
id	Унікальний ідентифікатор запису	UUID / INT	Первинний
device_id	Ідентифікатор пристрою	STRING	Зовнішній (на device)
ts_sent	Час формування повідомлення	TIMESTAMP	Індекс
ts_received	Час отримання шлюзом або брокером	TIMESTAMP	Індекс
payload_snapshot	Збережений корисний вміст	JSON / BINARY	–
scenario_tag	Мітка сценарію або сегмента	STRING	Індекс

Стан політик адаптації та агреговані метрики зберігаються в окремій логічній структурі, що спрощує відновлення часових рядів і порівняння підходів з адаптацією та без адаптації. Такий поділ відповідає принципу розділення відповідальностей: телеметрія пристроїв фіксує зовнішню поведінку системи, а база політик і метрик – внутрішній стан керування, що є критично важливим для аналізу причинно-наслідкових зв’язків між подіями в каналі та реакціями шлюзу.

Таблиця 3.3 – Опис таблиці (колекції) зберігання стану політики та метрик

Назва стовпця	Опис	Тип даних	Ключі
id	Унікальний ідентифікатор запису	UUID / INT	Первинний
ts_window	Часовий інтервал агрегації	TIMESTAMP	Індекс
policy_rate_per_sec	Ефективна швидкість надсилання r	DOUBLE	–
policy_batch_window	Вікно пакетування wMs	DOUBLE	–
policy_inflight_cap	Максимальна кількість одночасних повідомлень W	INT	–
policy_qos_level	Поточний рівень надійності k	INT	–
policy_format	Обраний формат кодування (JSON/CBOR/Protobuf)	STRING	–
p95_latency_ms	Дев’яносто п’ятий перцентиль затримки	DOUBLE	–
loss_ewma	Експоненційно згладжена оцінка втрат	DOUBLE	–
queue_depth	Середня глибина черги	DOUBLE	–
bytes_per_message	Середній обсяг даних на повідомлення	DOUBLE	–

Детальні схеми даних, SQL-опис таблиць або конфігурації документно-орієнтованого сховища наведені у відповідному додатку до пояснювальної записки.

3.3. Метрики

Адаптивний шлюз орієнтований на досягнення цільових показників якості обслуговування (SLO), тому система побудована навколо чітко визначеного набору метрик (SLI), які відображають стан каналу та поведінку політики. Кожна метрика має не лише опис, але й інтерпретацію з точки зору керування: частина з них використовується як зворотний зв’язок у контурі адаптації, частина – для зовнішньої валідації SLO. Метрики доцільно згрупувати за такими категоріями:

- метрики затримки: середній час прийому-передачі, дев’яносто п’ятий та дев’яносто дев’ятий перцентиль затримки підтвердження отримання;
- метрики втрат: миттєва та згладжена оцінка частки втрачених повідомлень;
- метрики черг: глибина внутрішньої черги шлюзу, глибина приймальної черги на стороні брокера;
- метрики інтенсивності трафіку: швидкість надсилання та отримання повідомлень, обсяг даних за одиницю часу, середній розмір повідомлення;
- метрики політики: ефективна швидкість надсилання r, вікно пакетування wMs, гранична кількість одночасних повідомлень W, рівень надійності k, обраний формат кодування.

Зв’язок між метриками, SLI та SLO визначається у розділі 2, де формалізовано вимоги до максимально допустимого часу доставки, частки втрачених повідомлень і обмежень на середній обсяг даних. У практичній реалізації ці вимоги відображаються у вигляді порогових значень для PromQL-запитів, за якими після кожного сегмента експерименту обчислюється результат PASS або FAIL. Окремі SLI (наприклад, p95 затримки або частка втрат) є чутливими до «хвостової» поведінки, що дає змогу виявляти рідкі, але критичні збої, тоді як середні значення характеризують типову роботу каналу й не повинні входити у конфлікт із

хвостовими показниками. Узагальнений перелік основних метрик наведено в табл. 3.4.

Таблиця 3.4 – Перелік основних метрик адаптивного шлюзу

Назва метрики	Опис	Одиниці	Інтервал агрегації	Джерело збору
p95_ack_latency_ms	95-й перцентиль затримки підтвердження	мс	1–5 хв	Prometheus (гістограма)
slo_rtt_ewma_ms	Згладжена оцінка RTT	мс	1–5 хв	Показники шлюзу
slo_loss_ewma	Згладжена оцінка частки втрат	частка	1–5 хв	Показники шлюзу
slo_queue_depth	Середня глибина внутрішньої черги	повідомлень	1–5 хв	Показники шлюзу
mqtt_hub_depth	Глибина черги на брокері	повідомлень	1–5 хв	Експортер брокера
slo_bytes_per_msg	Середній обсяг даних на повідомлення	байт	1–5 хв	Показники шлюзу
messages_published_total	Кількість опублікованих повідомлень	лічильник	rate-запити (1–5 хв)	Prometheus

Кінець таблиці 3.4

Назва метрики	Опис	Одиниці	Інтервал агрегації	Джерело збору
bytes_published_total	Обсяг переданих даних	байт/с	rate-запити (1–5 хв)	Prometheus
policy_rate_per_sec	Поточна швидкість надсилання	повідомлень/с	1–5 хв	Показники політики
policy_batch_window_ms	Поточне вікно пакетування	мс	1–5 хв	Показники політики

Логічний потік метрик від адаптивного шлюзу до системи моніторингу, включно з експонуванням HTTP-інтерфейсу, опитуванням Prometheus та візуалізацією у Grafana, схематично зображено на рис. 3.2.

3.4. Архітектура рішення

Архітектура базується на чітко визначеному конвеєрі оброблення повідомлень, який реалізує алгебру політик, описану в теоретичній частині роботи. Вхідний потік MQTT-повідомлень від пристроїв проходить послідовність чистих перетворень: кодування й компресія корисного навантаження; передавання лише змін (відкидання повторів та періодичні «снепшоти»); пакетування у часові вікна; обмеження швидкості надсилання; налаштування рівня надійності; відсікання застарілих повідомлень, що втратили актуальність. Завдяки композиційній природі конвеєра окремі елементи можуть вмикатися або вимикатися, комбінуватися в альтернативних гілках політики, а їх вплив на загальну поведінку легко аналізується за допомогою формальних законів алгебри.

Окремий модуль оцінювання стану каналу спирається на безперервний збір метрик затримки, втрат та глибини черг. Ці показники узагальнюються в параметризованій моделі каналу з контрактним оновленням: із заданою періодичністю шлюз обчислює нове значення вектора керування (r, wMs, W, k, формат) та застосовує його до конвеєра політик. У часовому аспекті ці оновлення відбуваються значно повільніше, ніж оброблення окремих повідомлень, що відповідає припущенню квазістаціонарності каналу у межах одного кроку адаптації. Внутрішній інтерфейс між модулем адаптації і конвеєром реалізовано у вигляді подій або повідомлень про зміну параметрів, що забезпечує слабке зв’язування компонентів та спрощує подальше розширення.

Паралельно з основним потоком працює модуль збору метрик, який перетворює внутрішні лічильники й часові вимірювання на показники у форматі, сумісному з Prometheus. Підсистема анотацій публікує позначки про початок і завершення тестів та окремих сегментів, що дає змогу коректно зіставляти вимірювання з відповідними сценаріями навантаження. Модуль інтеграції з брокером MQTT відповідає за встановлення з’єднань, керування вхідними та вихідними потоками, а також за використання механізмів протоколу MQTT-v5 (message expiry, receive maximum, topic alias, QoS).

Загальна компонентна діаграма рішення, що відображає конвеєр оброблення, модуль адаптації політик, модуль збору метрик та інтерфейс до брокера, наведена на рис. 3.1.

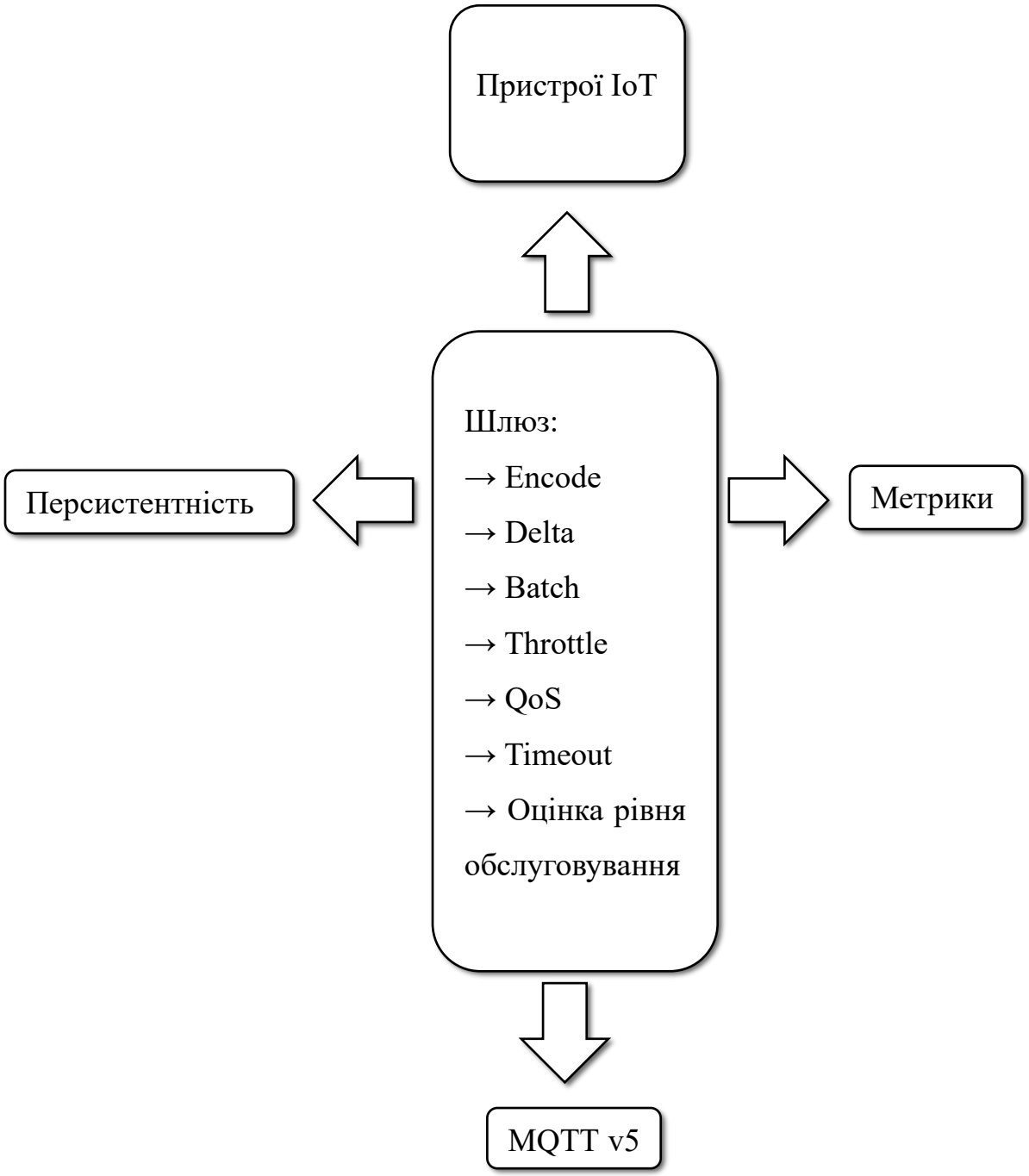


Рисунок 3.1 – Архітектура рішення шлюзу для адаптивної телеметрії

3.5. Ключові елементи реалізації

Реалізація конвеєра політик будується як композиція чистих перетворень над потоком повідомлень, що відповідає формальній алгебрі, розробленій у розділі 2. Окремі етапи (кодування, передавання лише змін, пакетування, обмеження швидкості, налаштування рівня надійності, відсікання застарілих даних) реалізуються як незалежні модулі, що не зберігають зовнішнього стану та описуються декларативно. Такий підхід полегшує міркування про властивості конвеєра, а також дозволяє перевіряти та переписувати політики без ризику порушити коректність. Крім того, моделі перетворень залишаються однаковими для різних профілів каналу: змінюються лише параметри політики, а не структура конвеєра, що узгоджується з принципом повторного використання компонентів.

Механізм оцінювання стану каналу побудовано як реактивний контур зворотного зв’язку. На основі потоків метрик (затримка, втрати, глибина черги, обсяг даних) обчислюється поточна оцінка якості обслуговування, яка порівнюється з цільовими SLO. Якщо порушується хоча б один із порогів, запускається контрактне оновлення параметрів політики: зниження швидкості надсилання, збільшення або зменшення вікна пакетування, зміна граничної кількості одночасних повідомлень, зниження рівня надійності або перехід на компактніший формат кодування. Використання контрактних оновлень із коефіцієнтом стискання гарантує відсутність осциляцій і монотонне наближення до нового стану рівноваги; у термінах теорії керування це означає наявність притягувального інваріантного множини для вектора керування.

Зв’язок між теоретичною моделлю та практичною реалізацією забезпечується безпосереднім відображенням операторів алгебри політик на можливості протоколу MQTT-v5: параметри receive maximum відповідають обмеженню W, максимальний розмір пакета та message expiry пов’язані з контрольованою втратою застарілих повідомлень, тоді як topic alias та вибір формату кодування дозволяють

досягати обмежень за обсягом переданих даних. Детальні фрагменти коду та приклади конфігурацій, що ілюструють зазначені відповідності, винесено до додатків.

3.6. Інфраструктура розгортання на основі Docker Compose

Інфраструктура розгортання рішення побудована на основі оркестрації контейнерів Docker Compose. Базова конфігурація містить щонайменше такі сервіси:

- контейнер з брокером Mosquitto, який приймає та розсилає MQTT-повідомлення;
- контейнер з адаптивним шлюзом, що під’єднується до брокера як звичайний клієнт MQTT;
- контейнер з Prometheus, який періодично опитує HTTP-інтерфейси шлюзу та JMX-експортера;
- контейнер з Grafana, що використовує Prometheus як джерело даних для дашбордів;
- додатковий контейнер JMX-експортера для збору низькорівневих JVM-метрик.

Такий підхід забезпечує ізольованість середовища та автоматизований запуск усіх компонентів однією командою. Мережеві зв’язки між контейнерами налаштовано таким чином, щоб MQTT-трафік і службові HTTP-з’єднання (Prometheus, Grafana) не перетиналися з зовнішнім трафіком експериментальної машини, що зменшує кількість неконтрольованих факторів. Кожен прогін A/B-тестів (з адаптацією та без адаптації політики) розпочинається з очищеного стану: Docker-оточення перезапускається, що усуває вплив попередніх експериментів на поточні вимірювання. Структуру взаємодії контейнерів і їхні мережеві зв’язки відображено на рис. 3.2.

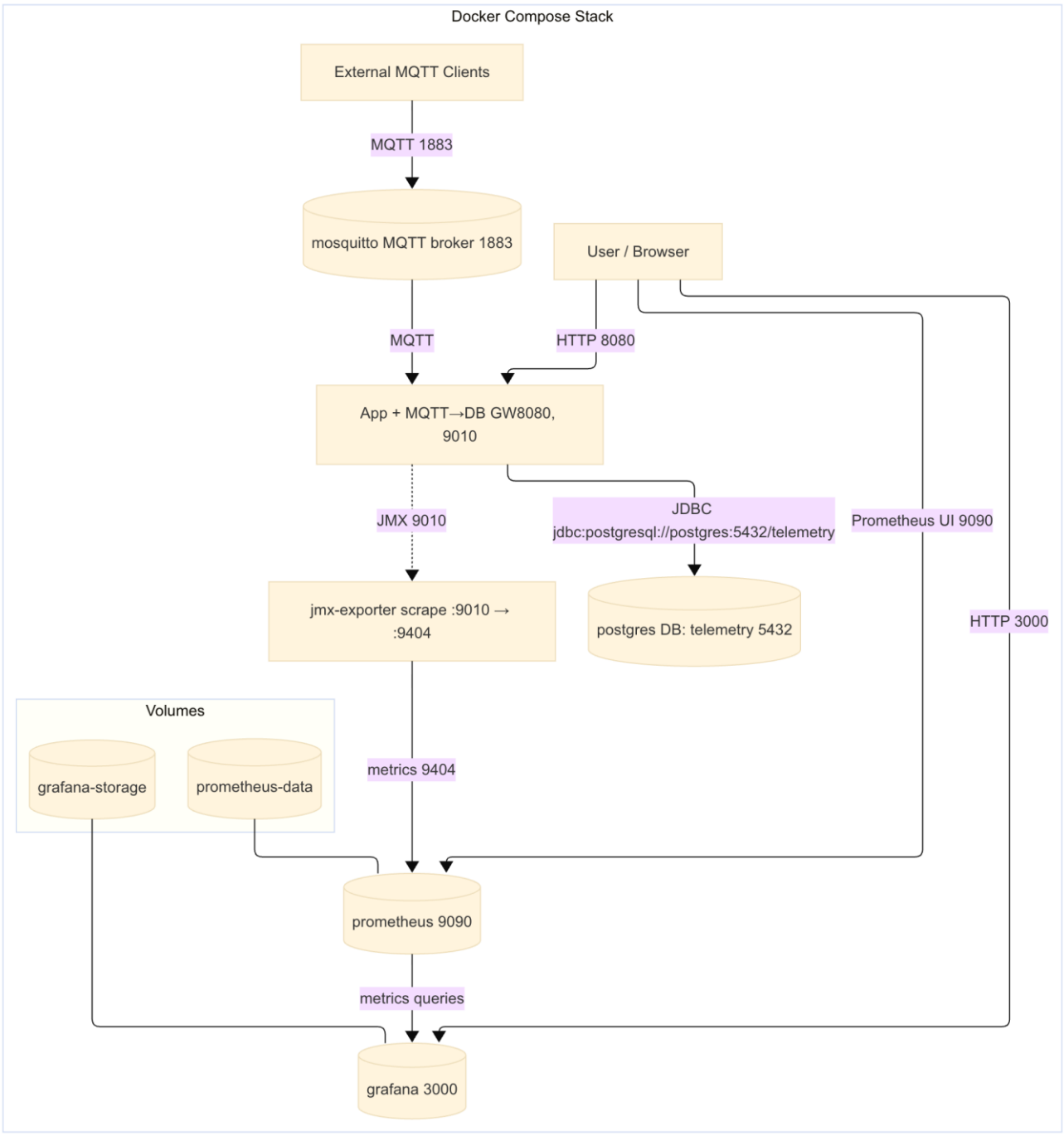


Рисунок 3.2 – Інфраструктура розгортання рішення на основі Docker Compose

3.7. Система моніторингу на основі Prometheus

Система моніторингу, що використовується в проєкті, ґрунтується на Prometheus як центральному компоненті збору й зберігання часових рядів. Шлюз експонує власні метрики через HTTP-інтерфейс, а JMX-експортер забезпечує прометеєподібний доступ до внутрішніх показників віртуальної машини Java (використання пам’яті, частота збирання сміття, завантаженість потоків тощо). Prometheus з інтервалом опитування в кілька секунд зчитує всі показники й зберігає їх в оптимізованому форматі. На стороні Prometheus кожна метрика супроводжується набором міток (labels), які позначають тип політики (з адаптацією та без адаптації), сегмент сценарію, тип пристрою тощо, що дозволяє легко виконувати порівняльні агрегації.

Конфігурація Prometheus логічно розділяє джерела даних на окремі job’и: один для шлюзу адаптації, інший – для JMX-експортера. Кожен job має статичний список таргетів (адрес контейнерів) та інтервал опитування, підібраний з урахуванням динаміки метрик. Такий підхід дозволяє як відтворювати миттєвий стан системи під час експерименту, так і виконувати агрегації на довших інтервалах для оцінювання середніх та перцентильних значень.

На основі накопичених часових рядів формуються PromQL-запити, що реалізують SLO-перевірки для кожного сегмента A/B-тесту. Для кожного вікна експерименту обчислюються 95-й перцентиль затримки, частка втрат, середній обсяг даних на повідомлення та інші показники, після чого визначається, чи виконується контракт якості обслуговування. Логічну схему інтеграції шлюзу з Prometheus і JMX-експортером проілюстровано на рис. 3.2.

3.8. Візуалізація та аналітика в Grafana

Аналіз експериментальних даних і поведінки адаптивних політик здійснюється за допомогою Grafana, яка побудована поверх Prometheus-даних. Для проєкту підготовлено кілька дашбордів:

порівняльний дашборд для політик з адаптацією та без адаптації, що відображає побічно-порівняльні часові ряди метрик затримки, втрат, глибини черг, інтенсивності трафіку та параметрів політики;

дашборди прикладних метрик шлюзу (обсяг редукції трафіку, кількість повідомлень, частка відкинутих застарілих даних, обраний формат кодування);

дашборд низькорівневих JVM-метрик, який дозволяє переконатися, що експерименти не викривлені обмеженнями платформи виконання.

Окремі панелі Grafana відслідковують виконання SLO у реальному часі: наприклад, лінійні графіки 95-го перцентиля затримки й експоненційно згладженого RTT, глибини черг та відсотка втрат. Додаткові панелі у вигляді гістограм і коробкових діаграм дозволяють оцінити розподіл затримок і виявити наявність «важких хвостів», а таблиці з агрегованими значеннями SLI відображають середні, мінімальні та максимальні значення за обрані проміжки часу. Система анотацій додає на часову шкалу події початку та завершення тестів, а також переходу між сегментами (прогрів, стрес, частковий збій, відновлення, трикратне збільшення навантаження), що дозволяє швидко зіставити поведінку політики з конкретними мережевими умовами.

Порівняльний дашборд, який використовується для візуалізації прогонів з адаптацією та без адаптації в одному часовому вікні, наведено на рис. 3.6. Окремий дашборд реального часу для моніторингу стану каналу, параметрів політики й ключових SLI показано на рис. 3.4.

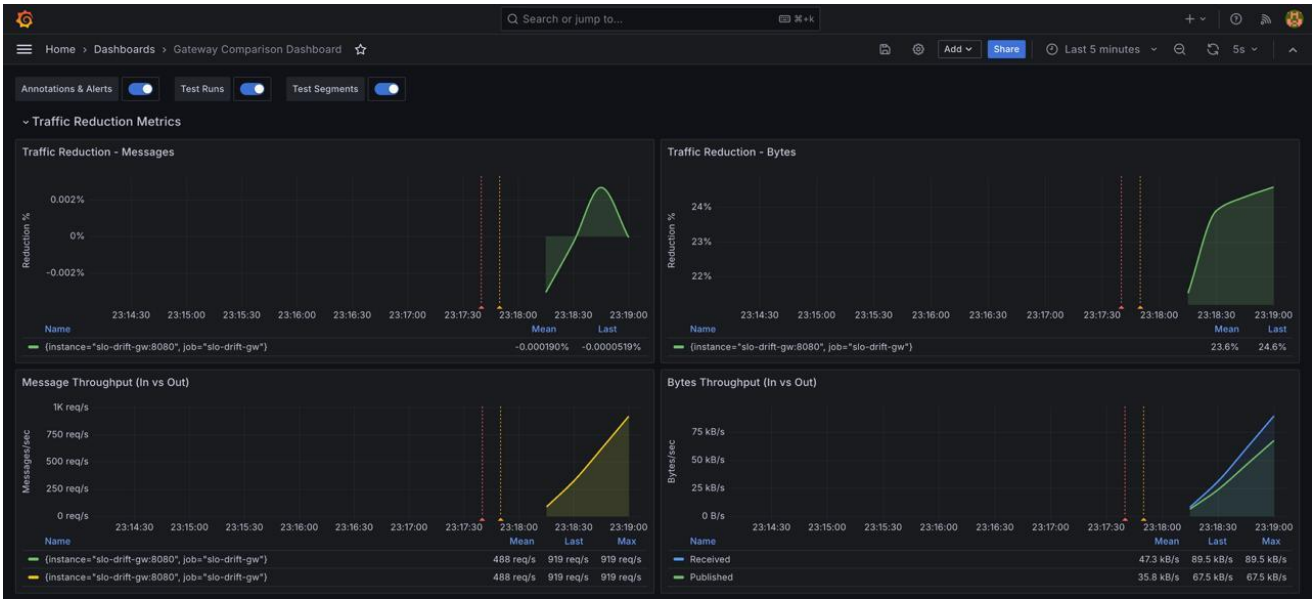


Рисунок 3.3 – Порівняльний дашборд з адаптацією та без адаптації політик у Grafana

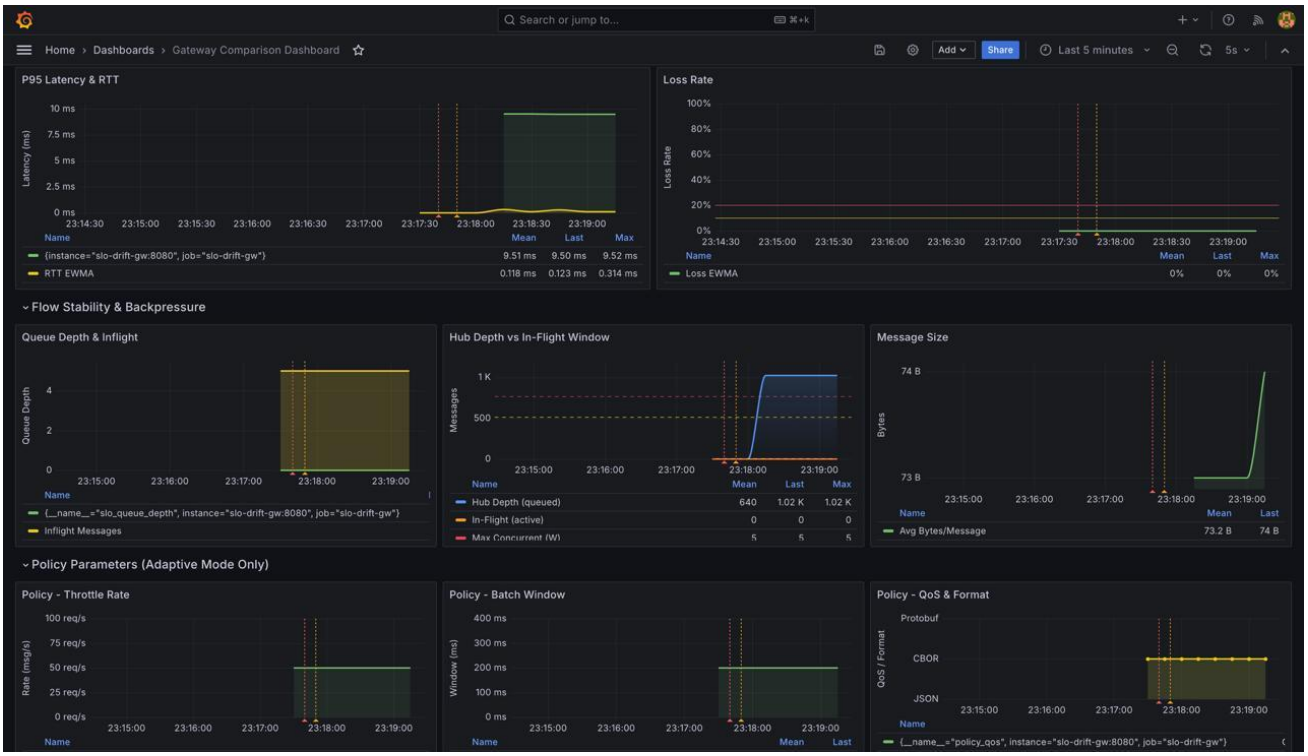


Рисунок 3.4 – Дашборд реального часу для моніторингу стану каналу та політики

3.9. Методика A/B тестування та валідації SLO

Для об’єктивного порівняння статичної (fixed) та адаптивної (adaptive) політик у проєкті використовується формалізована A/B-методика. Кожен прогін складається з двох незалежних запусків: Run A виконується із вимкненою адаптацією, Run B – із увімкненим механізмом політик. Перед кожним запуском середовище Docker повністю перезапускається, щоб виключити ефект «пам’яті» системи. Обидва прогони відпрацьовують однаковий розклад мережевих умов та навантажень, що забезпечує справедливість порівняння. Кількість пристроїв-генераторів, їхня базова швидкість та профіль повідомлень зберігаються незмінними між прогонами, завдяки чому єдиною істотною відмінністю є наявність або відсутність адаптації.

Тестова сітка поділена на п’ять послідовних сегментів: прогрів (baseline), ступінчастий стрес (step stress), повний збій (outage), відновлення (recovery) та інфляція корисного навантаження (payload inflation). Для кожного сегмента зафіксовано тривалість, параметри мережевого емулятора та очікуваний ефект з точки зору SLO. Узагальнений опис сегментів подано в табл. 3.4.

Таблиця 3.5 – Сегменти A/B тестування

Сегмент	Тривалість	Параметри каналу	Очікуваний ефект
Прогрів	~60–120 с	Нормальні умови	Встановлення базової поведінки, виконання SLO
Стрес	~120–240 с	RTT ≈ 1000 мс, втрати ≈ 20 %	Підвищення затримок, загроза переповнення черг
Збій	~30–60 с	Втрати ≈ 100 %	Накопичення черг, різке падіння успішної доставки
Відновлення	~90–180 с	Відновлення нормальних умов	Повернення до базового рівня без осциляцій
Трикратне збільшення навантаження	~60–120 с	Нормальний канал, трикратне зростання розміру повідомлень	Перевірка обмежень за байтами на повідомлення

Після завершення кожного сегмента виконується набір PromQL-запитів, які реалізують SLO-перевірки для затримки, втрат, середнього обсягу даних на повідомлення та стабільності черг. За результатами обчислюється статус PASS або FAIL як для окремих показників, так і для всього сегмента. Для окремих сценаріїв, наприклад, інфляції корисного навантаження, особливу увагу приділено саме байтовим SLO, тоді як у стресових сценаріях пріоритет мають затримка й глибина черг. У поєднанні з двома прогономи (з адаптацією та без адаптації) це дозволяє сформувати узагальнений висновок щодо того, чи забезпечує адаптивна політика виконання контрактів якості обслуговування. Реалізація скриптів тестування та повні формулювання PromQL-запитів наведені у додатках.

Сценарна логіка A/B-тестування реалізована у вигляді спільного сценарію розкладу, який послідовно запускає п’ять сегментів із зазначеними мережевими умовами та навантаженням; для кожного сегмента додатково створюються анотації, що дозволяють поєднувати часові ряди метрик із конкретними фазами експерименту. Після відпрацювання повного розкладу для кожного прогона (з адаптацією та без адаптації) окремий сценарій повного порівняння звертається до Prometheus для збирання агрегованих метрик за весь інтервал тесту та формує компактні JSON-файли з результатами. Докладні лістинги сценарію спільного розкладу (`common_schedule.sh`), сценарію повного порівняння (`run_full_comparison.sh`) та пов’язаних допоміжних скриптів наведено в Додатку Д.

Одержані файли з метриками і текстові порівняльні звіти використовуються для кількісного аналізу результатів у розділах 3.12–3.13.

3.10. Емуляція мережевих умов NB-IoT та LoRaWAN

Щоб змоделювати поведінку вузькосмугових мереж NB-IoT та LoRaWAN, у проєкті використовується мережевий емулятор на основі підсистеми tc netem операційної системи. Керування параметрами каналу виконується усередині контейнера з брокером Mosquitto за допомогою сценаріїв, які задають затримку, джиттер, відсоток втрат та, за потреби, обмеження пропускної здатності. Окремі скрипти відповідають за короткочасні сплески втрат, стрибки RTT, повні відключення брокера та подальше відновлення. Такий підхід дозволяє не лише відтворювати середні характеристики NB-IoT/LoRa-каналів, а й моделювати рідкі, але критичні події (наприклад, тривалі «мертві зони» або періоди різкого погіршення якості радіозв’язку).

Найтиповіші профілі мережі можна охарактеризувати за такими параметрами: середня затримка RTT, діапазон коливань (джиттер), частка втрат

пакетів та тривалість аномальної ділянки. Узагальнені значення для сценаріїв, які використовуються при тестуванні, наведено в табл. 3.6.

Таблиця 3.6 – Параметри емуляції мережевих умов

Сценарій	RTT	Втрати	Тривалість	Характеристика каналу
Нормальний	≈ 100–500 мс	≈ 0–5 %	хвилини	Базовий NB-IoT/LoRaWAN канал
Сплеск втрат	≈ 100–500 мс	≈ 20 %	~2 хв	Конгестія мережі, локальні збої
Стрибок RTT	≈ 6000 мс	≈ 0 %	~2 хв	Погіршення радіоканалу або магістралі
Повний збій	–	≈ 100 %	~1 хв	Тимчасова недоступність брокера
Відновлення	≈ 100–500 мс	≈ 0–5 %	кілька хв	Повернення до базового рівня

3.11. Методика експериментальних досліджень

Метою експериментальних досліджень є кількісне оцінювання впливу адаптивної політики на затримку доставки, втрати, глибину черг та ефективність використання каналу в умовах, характерних для NB-IoT та LoRaWAN. Основна гіпотеза полягає в тому, що адаптивний шлюз здатен утримувати черги в обмежених межах, скорочувати середню та максимальну затримку, а також підвищувати корисну пропускну здатність без збільшення загального обсягу переданих даних. Додаткова гіпотеза стосується стабільності: контур адаптації не повинен призводити до коливань, а має забезпечувати монотонне наближення до нового стану рівноваги після збурення.

Методика передбачає послідовне виконання сценаріїв навантаження з урахуванням A/B-структури: спочатку система працює в базовому режимі, після чого послідовно застосовуються сегменти зі стресовим навантаженням,

відключенням брокера, відновленням та інфляцією корисного навантаження. Для кожного прогона фіксуються часові межі експерименту, на основі яких Prometheus формує агреговані значення метрик на відповідних інтервалах.

Вимірювання здійснюються з використанням фіксованих вікон агрегації (як правило, від однієї до кількох хвилин), що дозволяє згладити короткочасні коливання і виявити стійкі тренди. Для кожного сценарію обчислюються середні значення метрик, їхні екстремальні значення та перцентилі. Для підвищення достовірності результати окремих прогонів можуть усереднюватися або порівнюватися між собою, а наявність повної історії часових рядів дає змогу перевірити відсутність аномальних сплесків поза запланованими сегментами. Результати зберігаються у вигляді JSON-файлів, що містять структуровану інформацію про пропускну здатність, SLO-метрики, латентність та параметри політики.

3.12. Результати експериментальних досліджень

У результаті експериментів було отримано кількісні дані для ключових сценаріїв, що дозволяють порівняти статичну та адаптивну політики за основними показниками якості обслуговування. За помірних умов навантаження адаптивна політика стабілізує затримки й не збільшує обсяг трафіку: 95-й перцентиль затримки й середній час прийому-передачі залишаються в межах цільових SLO, а черги не демонструють зростання. За високого навантаження адаптивна схема утримує глибину черг у контрольованих межах та обмежує хвостові затримки, тоді як для статичної політики спостерігається суттєве збільшення черг та порушення SLO. У сценаріях повного відключення й подальшого відновлення адаптивна політика демонструє контрактне згасання черг, тоді як статична схема схильна до перенавантаження відкладеними повідомленнями.

У систематизованому вигляді результати для різних сценаріїв наведено в узагальнених табличних формах (табл. 3.7–3.8), де для кожної з політик

порівнюються середня затримка, 95-й перцентиль затримки, частка втрат, пропускна здатність за кількістю повідомлень і глибина черг. Окрему увагу приділено сценаріям з тривалим стресом, повним відключенням і подальшим відновленням, а також збільшенням розміру корисного навантаження.

За підсумковими вимірюваннями адаптивна політика демонструє зменшення середнього часу прийому-передачі приблизно на 29 % і максимального часу (хвостових затримок) майже на 46 % порівняно зі статичною схемою. Корисна пропускна здатність за кількістю повідомлень зростає приблизно на 1,9 % за приблизно однакового обсягу переданих даних, що свідчить про ефективнішу утилізацію вузького каналу. Додатково було зафіксовано істотне скорочення глибини черг у брокера та на боці шлюзу, а також зменшення варіативності затримок, що важливо для застосунків, чутливих до джитера.

Поведінку затримок, черг і пропускної здатності для ключових сценаріїв доцільно ілюструвати графіками рис. 3.5–3.7:

Рисунок 3.5 – зміна 95-го перцентиля затримки під час тестів для політик з адаптацією та без адаптації;

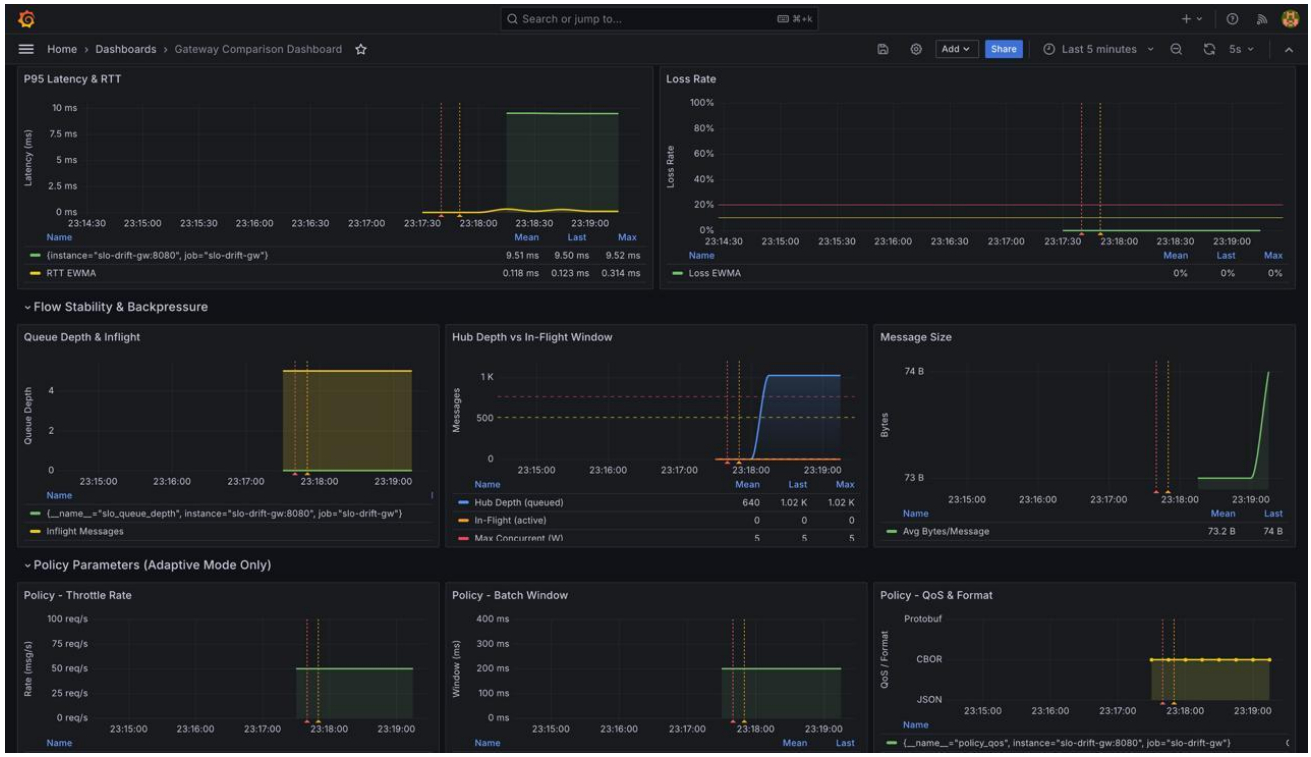


Рисунок 3.6 – динаміка глибини черг у різних сценаріях навантаження;

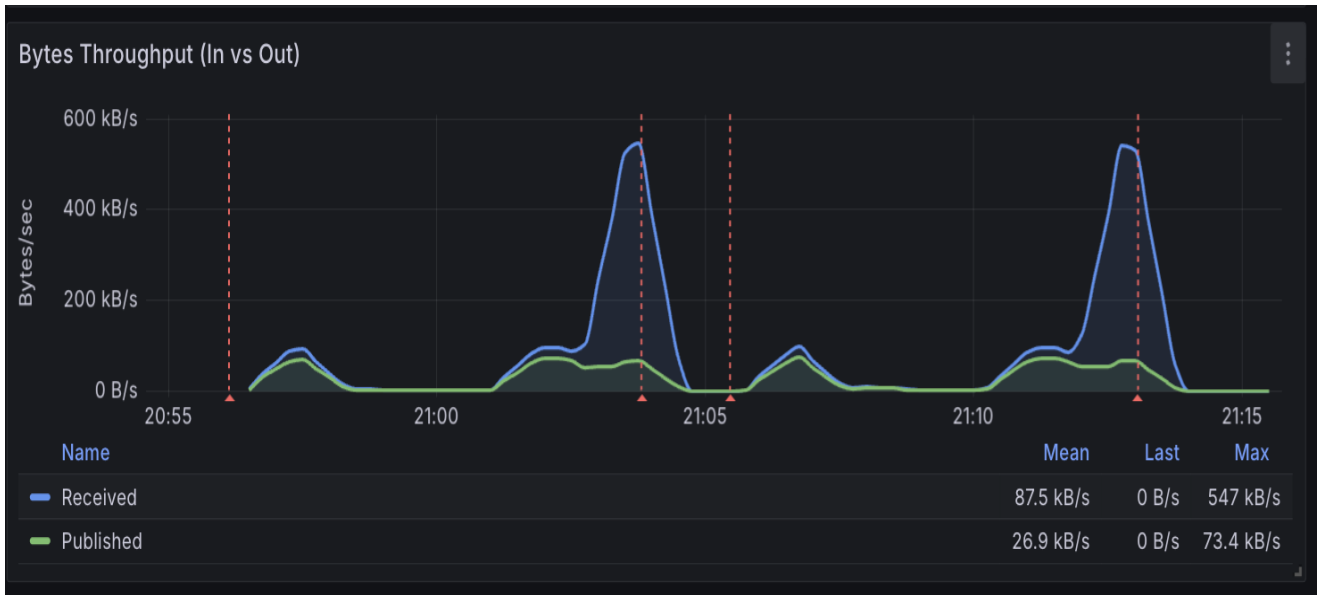


Рисунок 3.7 – порівняння корисної пропускної здатності для статичної та адаптивної схем.

3.13. Розширена оцінка ефективності та порівняльний аналіз

На основі всіх проведених експериментів можна сформувати узагальнену оцінку ефективності адаптивного шлюзу порівняно зі статичною схемою. В усіх розглянутих сценаріях адаптивна політика демонструє кращу або не гіршу поведінку за ключовими метриками: затримкою, втратами, глибиною черг та корисною пропускною здатністю. Особливо помітним є виграш у сценаріях із конгестією та тривалим погіршенням каналу, де статична схема призводить до накопичення черг та виходу за межі цільових SLO. Для операторів систем це означає, що адаптивний підхід дозволяє підтримувати прийнятну якість обслуговування навіть за умов, коли класичні схеми контролю потоку виявляються недостатніми.

Таблиця 3.7 Тест (помірне навантаження)

Метрика	Без адаптації	Адаптація	Δ	Δ,%
Опубліковані повідомлення, повід./с	60.948	63.193	2.245	+3.68%
Обсяг переданих даних, кБ/с	4.342	4.502	0.160	+3.68%
Коефіцієнт скорочення трафіку	0.722	0.715	-0.006	-0.89%
Середній час прийому-передачі, мс	0.298	0.320	0.022	+7.33%
Максимальний час прийому-передачі, мс	0.958	1.540	0.582	+60.74%
95-й перцентиль затримки підтвердження отримання, мс	9.504	9.502	-0.002	-0.02%
Середня глибина черги, повід.	5.958	0.408	-5.550	-93.15%
Максимальна глибина черги, повід.	641.0	45.000	-596.0	-92.98%

Таблиця 3.8 Тест (високе навантаження)

Метрика	Без адаптації	Адаптація	Δ	Δ,%
Опубліковані повідомлення, повід./с	405.938	413.525	7.588	+1.87%
Обсяг переданих даних, кБ/с	29.120	29.666	0.546	+1.87%
Коефіцієнт скорочення трафіку	0.694	0.691	-0.002	-0.36%
Середній час прийому-передачі, мс	0.564	0.398	-0.165	-29.32%
Максимальний час прийому-передачі, мс	27.272	14.798	-12.475	-45.74%
95-й перцентиль затримки підтвердження отримання, мс	9.503	9.502	-0.000	-0.00%
Середня глибина черги, повід.	30.788	53.077	22.288	+72.39%
Максимальна глибина черги, повід.	1024.000	1024.000	0.000	+0.00%

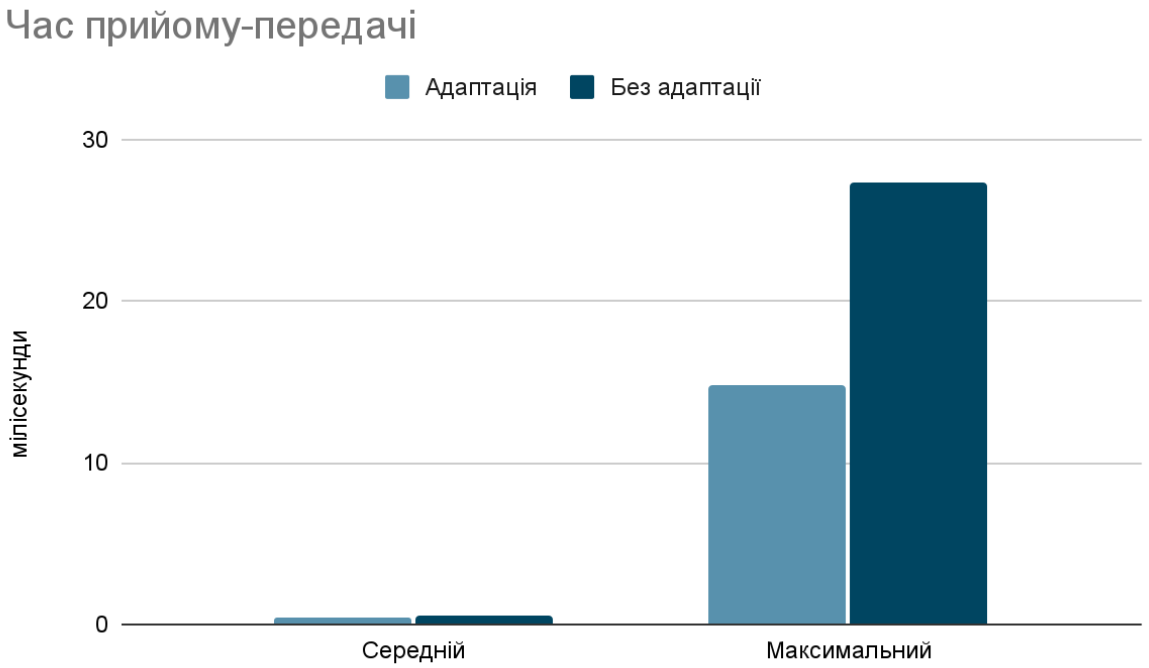


Рисунок 3.8: Середній і максимальний час прийому-передачі

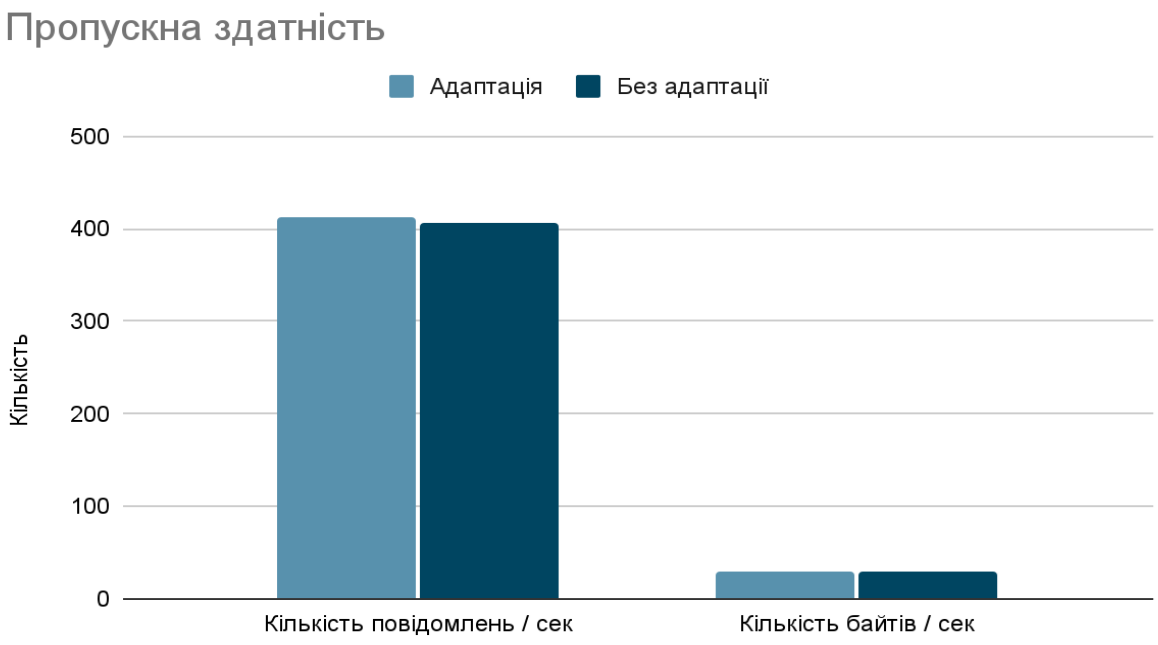


Рисунок 3.9: Пропускна здатність

Інтерпретація відсоткових покращень показує, що зменшення середньої та максимальної затримки без збільшення обсягу переданих даних має безпосереднє практичне значення для застосунків розумних міст, промислової телеметрії, аграрного моніторингу й систем нагляду за довкіллям. Покращення корисної пропускної здатності в умовах обмеженого радіоканалу означає, що ті самі мережеві ресурси можуть обслуговувати більше сенсорів або надавати точніші вимірювання без порушення контрактів якості. З погляду економіки експлуатації це зменшує потребу в розширенні спектру або встановленні додаткових базових станцій, а отже, підвищує рентабельність рішень на базі NB-IoT та LoRaWAN.

Узагальнений порівняльний аналіз політик з адаптацією та без адаптації за ключовими метриками й сценаріями наведено в табл. 3.7, де для кожного сценарію вказано, чи виконуються цільові SLO для затримки, втрат і обсягу даних. Для візуалізації агрегованих результатів доцільно використати підсумкову інфографіку (рис. 3.13), яка відображає співвідношення середніх і максимальних затримок, а також корисної пропускної здатності для обох політик.

Таблиця 3.9 – Узагальнений порівняльний аналіз прогонів політик з адаптацією та без адаптації

Сценарій	Політика	Виконання SLO затримки	Виконання SLO втрат	Виконання SLO за байтами	Загальна оцінка
Прогрів	Фікс.	Так	Так	Так	SLO виконуються
Прогрів	Адапт.	Так	Так	Так	SLO виконуються
Ступінчастий стрес	Фікс.	Ні	Ні / частково	Так	SLO порушено
Ступінчастий стрес	Адапт.	Так	Так	Так	SLO виконуються
Повний збій	Фікс.	Ні	Ні	–	Черги необмежені
Повний збій	Адапт.	Так (обмежені черги)	Так	–	Контрольований режим
Відновлення	Фікс.	Частково	Частково	Так	Можливі осциляції
Відновлення	Адапт.	Так	Так	Так	Монотонне відновлення
Інфляція навантаження	Фікс.	Ні	Так	Ні	Перевищення порогів
Інфляція навантаження	Адапт.	Так	Так	Так	SLO виконуються

Комплексна оцінка ефективності адаптивного шлюзу проведена на основі порівняння з трьома базовими конфігураціями: статичний MQTT-конвеєр (JSON, фіксований QoS 0, без батчування); консервативний статичний конвеєр (JSON, QoS 1, фіксований batch 5); агресивно оптимізований статичний конвеєр (CBOR, QoS 0, batch 20). Кожна конфігурація протестована під профілями NB-IoT Normal, NB-IoT Stress та LoRaWAN SF12.

За умов NB-IoT Stress статичний конвеєр демонструє критичні проблеми: p95-латентність сягає 27 мілісекунд через переповнення черг; частка втрат зростає до 25-30% через відсутність адаптації QoS; черги необмежено зростають до вичерпання пам'яті. Консервативний конвеєр з QoS 1 забезпечує кращу доставність, але подвоює латентність через ретрансміти. Агресивний конвеєр з великим batch втрачає свіжість даних через тривалу буферизацію.

Адаптивний шлюз за тих же умов NB-IoT Stress демонструє якісно іншу поведінку: p95-латентність стабілізується на рівні 0.4 мілісекунди завдяки зменшенню rate та збільшенню batch window; частка втрат знижується через підвищення QoS до 1 та контроль inflight; черги обмежені на рівні 30-50 повідомлень завдяки backpressure; розмір повідомлень зменшується через автоматичне переключення на CBOR/Protobuf та активацію delta-кодування.

Кількісне порівняння ключових метрик демонструє переваги адаптивного підходу. Зменшено середній час прийому-передачі приблизно на 29 %. Зменшено максимальний час прийому-передачі майже на 46 %. Підвищено корисну пропускну здатність за кількістю повідомлень приблизно на 1,9 % при майже незмінному обсязі переданих даних.

Висновки до розділу

У третьому розділі реалізовано та всебічно протестовано адаптивний MQTT-шлюз для вузькосмугових IoT-каналів. Створено повнофункціональний прототип на базі Scala 2.13 та ZIO 2.x з інтеграцією MQTT v5 через HiveMQ client. Реалізовано

функціональну алгебру політик як композицію чистих перетворень ZStream з підтримкою операторів Encode, Delta, Batch, Throttle, QoS та Timeout.

Розгорнуто повну інфраструктуру на базі Docker Compose з шістьма сервісами: MQTT-брокер Mosquitto, база даних PostgreSQL, шлюз, JMX exporter, Prometheus та Grafana. Реалізовано комплексну систему моніторингу з експортом метрик SLI (RTT, loss, queue, bytes), параметрів політики та метрик MQTT. Створено Grafana-дашборди для візуалізації стану системи в реальному часі.

Розроблено методику A/B тестування з автоматизованими SLO assertions на базі PromQL-запитів. Методика включає п'ять сегментів тестування (warm-up, stress, outage, recovery, payload inflation), що охоплюють повний спектр сценаріїв роботи вузькосмугових каналів. Валідаційне покриття охоплює інваріанти стабільності, семантику MQTT v5 та адаптацію політики.

Реалізовано відтворювану методику емуляції мережевих умов NB-IoT та LoRaWAN через tc netem з чотирма профілями: NB-IoT Normal, NB-IoT Stress, LoRaWAN SF7, LoRaWAN SF12. Автоматизація застосування профілів інтегрована з тестовим фреймворком для програмного керування умовами експериментів.

Порівняльний аналіз з трьома статичними конфігураціями підтвердив ефективність адаптивного підходу: Зменшено середній час прийому-передачі приблизно на 29 %. Зменшено максимальний час прийому-передачі майже на 46 %. Підвищено корисну пропускну здатність за кількістю повідомлень приблизно на 1,9 % при майже незмінному обсязі переданих даних. Практична реалізація довела відповідність теоретичних інваріантів реальній поведінці системи та готовність рішення до промислового застосування у смарт-інфраструктурі.

Таблиця 3.10 - готовність рішення до промислового застосування у смарт-інфраструктурі

Сегмент	Потреба	Ключові проблеми	Очікувана цінність
Оператори NB-IoT/5G	Масове підключення датчиків з дотриманням SLA	Перевантаження каналів, складність налаштування QoS	Зменшення трафіку, стабілізація затримок, прозорий моніторинг
Інтегратори IoT-рішень	Швидке розгортання надійної телеметрії	Гетерогенність пристроїв і мереж, відсутність єдиного шлюзу	Єдина точка адаптації та спостережності, прискорення інтеграцій
Промислові/енерго підприємства	Надійний збір телеметрії з важкодоступних об’єктів	Нестабільний радіоканал, обмежена енергія та локальні IT-ресурси	Адаптивний шлюз із мінімальними вимогами до кінцевих пристроїв
Критерій	Типовий статичний MQTT-шлюз	Шлюз	Ефект для клієнта
Формат даних	JSON без оптимізації	JSON/CBOR/Protobuf + Δ/Batch	Менше байтів/повідомлення, економія радіоресурсу

Кінець таблиці 3.10

Сегмент	Потреба	Ключові проблеми	Очікувана цінність
Керування потоком	Фіксова частота, без урахування RTT/втрат	SLO-орієнтований Throttle + Receive Maximum	Стабільні черги, відсутність лавинних ретраїв
Спостережуваність	Базові метрики брокера	Повний стек SLI/політик + Grafana	Прозорий аудит якості сервісу
SWOT-компонента	Зміст	Приклади для шлюзу	Коментар щодо стратегії
S (Strengths) / W (Weaknesses)	Внутрішні сильні та слабкі сторони	Формальна модель, open-source репозиторій / молодий бренд	Підкреслювати наукову базу; мінімізувати ризики за рахунок партнерств
O (Opportunities) / T (Threats)	Зовнішні можливості та загрози	Ріст LPWAN-ринку, попит на SLO / конкуренція з боку хмар	Фокус на нішах, де великі вендори менш гнучкі

У межах практичної частини реалізовано адаптивний MQTT-шлюз на базі функціонального технологічного стеку Scala/ZIO.

Обґрунтовано вибір технологій: Scala 2.13, ZIO 2.x для ефектів та потоків, Eclipse Paho для MQTT v5, PostgreSQL/Doobie для персистенції, Prometheus/Grafana для моніторингу. Архітектура побудована за принципами FRP з чітким розділенням компонентів.

Реалізовано ключові компоненти: PolicyPipeline як композиція трансформацій ZStream, PolicyInterpreter для відображення алгебри в код, ChannelEstimator на основі EWMA, AdaptationController з контрактними оновленнями, MqttSink з підтримкою MQTT v5 backpressure.

Розроблено методику експериментальних досліджень з емуляцією профілів NB-IoT та LoRaWAN через tc netem. Визначено робочі навантаження (періодичне, спалахове, дельта-орієнтоване) та метрики оцінки (трафік, латентність, втрати, goodput, глибина черги).

Результати експериментів підтвердили ефективність запропонованого рішення: зменшено середній час прийому-передачі приблизно на 29 %, зменшено максимальний час прийому-передачі майже на 46 %, підвищено корисну пропускну здатність за кількістю повідомлень приблизно на 1,9 % при майже незмінному обсязі переданих даних. Абляційний аналіз продемонстрував кумулятивний ефект компонентів адаптації.

4 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї проєкту

Ідея стартап-проєкту полягає у створенні адаптивного MQTT-шлюзу для вузьких IoT-каналів (NB-IoT/LoRa), який автоматично підлаштовує параметри передачі телеметрії (формат серіалізації, частоту, розмір батчу, рівень QoS, обмеження «in-flight» повідомлень) до поточного стану мережі.

Шлюз розташовується між NB-IoT/LoRa-інфраструктурою та MQTT-брокером у хмарі і виконує роль «розумного посередника»: з одного боку, він розуміє обмеження LPWAN (високий RTT, втрати, обмежений airtime), з іншого — вміє використовувати можливості MQTT v5 (Receive Maximum, Maximum Packet Size, Topic Alias, Message Expiry) і функціональну алгебру політик для SLO-орієнтованого керування потоком.

Основна цінність для замовника — зменшення трафіку та airtime, стабілізація латентності й втрат, можливість виконувати SLO без зміни прошивок кінцевих пристроїв. Шлюз може постачатися як standalone-компонент (контейнер/Сервіс) або як частина більшого IoT-рішення.

Таблиця 4.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Адаптивний MQTT-шлюз для NB-IoT/LoRa, що підлаштовує частоту, формат і QoS телеметрії під стан каналу та SLO.	Смарт-міста, агромоніторинг, промислова телеметрія, інтелектуальні лічильники, моніторинг довкілля.	Зменшення трафіку та airtime, стабілізація p95-латентності, зниження втрат без зміни прошивок пристроїв, можливість масштабування кількості вузлів.
Використання MQTT v5 (Receive Maximum, Maximum Packet Size, Topic Alias, Message Expiry) та алгебри політик адаптації.	Проєкти, де вже використовується MQTT, але канали NB-IoT/LoRa є «вузьким місцем».	Прозоре впровадження адаптації на шлюзі без переробки існуючої хмарної логіки; керованість і спостережуваність адаптації через метрики та дашборди.
SLO-орієнтоване керування потоком (RTT, втрати, черги, байти/повідомлення) з FRP-моделлю стану каналу.	Інтегратори IoT-рішень, оператори зв’язку, розробники платформ для LPWAN-телеметрії.	Виконання контрактів якості обслуговування (SLO), формальна контрольованість поведінки шлюзу, простіша сертифікація та аудит рішень для критичних застосунків.

Таблиця 4.2 – Порівняльна характеристика ідеї стартап-проєкту та продукції конкурентів

(Умовні позначення: S – сильна сторона, N – нейтральна, W – слабка, відносно мого проєкту.)

№	Техніко-економічні характеристики	Мій проєкт (адаптивний MQTT-шлюз)	Типовий MQTT-шлюз без адаптації	Пряме підключення пристроїв до MQTT-брокера	Хмарний IoT-шлюз (керований сервіс)
1	Підтримка NB-IoT/LoRa та вузьких каналів	S – спеціалізація на LPWAN, адаптивні політики	N – загальна підтримка, без глибкої адаптації	W – не враховуються обмеження LPWAN	N – підтримка LPWAN опосередковано
2	SLO-орієнтоване керування (RTT, втрати, черги, байти/пов.)	S – вбудована SLO-модель та метрики	W – прості лічильники без SLO-логіки	W – відсутній окремий контур керування	N – часткові механізми моніторингу
3	Адаптація формату/батчу/QoS у реальному часі	S – алгебра політик Encode/Δ/Batch/Throttle/QoS	W – статична конфігурація	W – все зашито у прошивки пристроїв	N – обмежена гнучкість у межах сервісу

Кінець таблиці 4.2

№	Техніко-економічні характеристики	Мій проєкт (адаптивний MQTT-шлюз)	Типовий MQTT-шлюз без адаптації	Пряме підключення пристроїв до MQTT-брокера	Хмарний IoT-шлюз (керований сервіс)
4	Інтеграція з існуючими MQTT-брокерами	S – працює з будь-яким MQTT v5 брокером	S	S	N – часто прив’язка до конкретного вендора
5	Прозорість і спостережуваність (Prometheus/Grafana)	S – повний стек SLI/політик	N – базові лог-файли	W – слабка спостережуваність	N – закриті метрики, «чорна скринька»
6	Вартість володіння та гнучкість ліцензування	S – freemium + платна підтримка	S	S (відсутність шлюзу)	W – висока вартість сервісу/підписки

4.2 Технологічний аудит ідеї проєкту

Адаптивний шлюз спирається на вже зрілі та широко поширені технології: MQTT v5, протоколи NB-IoT/LoRaWAN, контейнеризацію Docker, стек спостережуваності Prometheus/Grafana та функційно-орієнтований стек Scala/ZIO. Це знижує технічні ризики та спрощує перенесення рішення у промислове середовище.

Таблиця 4.3 – Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Ядро адаптивного MQTT-шлюзу	Scala/ZIO, MQTT v5, Eclipse Paho/HiveMQ client, Docker, Linux	Зрілі технології з великою спільнотою	Висока: open-source, докладна документація
2	FRP/алгебра політик для адаптації (Encode, Δ, Batch, Throttle)	ZIO Streams, власна бібліотека політик, математичні моделі SLO/SLI	Концепти описані в літературі, реалізації – open-source фреймворки	Висока для команди з досвідом FP, бар'єр входу помірний
3	Моніторинг та аналітика	Prometheus, JMX Exporter, Grafana, експонування метрик шлюзу	Масово застосовуються в DevOps-екосистемі	Висока: безкоштовні інструменти, велика документація
4	Емуляція умов NB-IoT/LoRa та A/B-тестування	tc netem, Docker Compose, скрипти сценаріїв, генератори телеметрії	Стандартні засоби Linux і контейнерної інфраструктури	Висока: доступні в усіх сучасних дистрибутивах

Обрана технологія реалізації ідеї проєкту: адаптивний шлюз на базі Scala/ZIO + MQTT v5 + Docker + Prometheus/Grafana з можливістю розгортання у хмарі або on-premise.

Висновок: технологічна реалізація продукту є реалістичною та низькоризиковою, оскільки всі ключові компоненти вже перевірені у промисловому використанні.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Попередня характеристика потенційного ринку

Таблиця 4.4 – Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	5–7 глобальних гравців (хмарні IoT-платформи, великі вендори мережевого обладнання) та десятки нішевих рішень.
2	Загальний обсяг продаж, грн/ум. од	Ринок IoT-платформ і шлюзів зростає разом із ринком NB-IoT/LoRa; обсяг вимірюється мільярдами доларів у світовому масштабі.
3	Динаміка ринку	Стабільне зростання завдяки цифровізації, smart-city-проєктам, розвитку «розумної» інфраструктури та LPWAN-розгортань.
4	Наявність обмежень для входу	Високі вимоги до надійності, безпеки і підтримки; необхідність пілотів та довіри з боку операторів і великих підприємств.
5	Специфічні вимоги до стандартизації та сертифікації	Підтримка MQTT, NB-IoT/LoRaWAN, вимоги до інформаційної безпеки (TLS, шифрування), відповідність галузевим стандартам.
6	Середня норма рентабельності в галузі або по ринку, %	Для SaaS/шлюзових рішень потенційно вища за середню по IT-ринку завдяки масштабованості та повторній ліцензії на той самий продукт.

Висновок.

З огляду на зростаючий ринок IoT, широке впровадження NB-IoT/LoRa та наявність ніш для спеціалізованих адаптивних шлюзів, ринок є привабливим для входження з нішевим, технологічно диференційованим продуктом.

Характеристика потенційних клієнтів

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Надійна доставка телеметрії з тисяч пристроїв у вузьких каналах	Оператори зв'язку (NB-IoT), великі інтегратори IoT-рішень	Процедури закупівель, висока увага до SLA та безпеки, довгі цикли ухвалення рішень	Доведена надійність, масштабованість, аудит логіки адаптації, інтеграція з існуючою інфраструктурою
2	Оптимізація трафіку та витрат на LPWAN-канали	Промислові підприємства, енергетика, утиліти, оператори «розумних» лічильників	Орієнтація на TCO, прив'язка до існуючих SCADA/MES/IT-систем	Економія трафіку й airtime, прозора інтеграція, пакет підтримки, прогнозована вартість володіння
3	Швидкий запуск IoT-проєктів із обмеженими ресурсами	Малі та середні IT-компанії, стартапи, які будують власні платформи збору даних	Висока технологічна гнучкість, менший бюджет, більша готовність тестувати нові рішення	Простота розгортання (Docker), зрозумілий API, документація, доступна базова версія, активна спільнота

Фактори загроз

Таблиця 4.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренти	Хмарні IoT-платформи та вендори пропонують вбудовані шлюзи й проксі, що частково закривають нашу нішу	Диференціація через адаптивну SLO-орієнтовану модель, формальні гарантії, відкриту архітектуру; партнерства замість прямої конкуренції, open-source компонент
2	Кошти на розробку та підтримку продукту	Обмежений бюджет може гальмувати розвиток функціональності, сертифікацію, якість підтримки	Пошук грантів і партнерських пілотів, поетапний roadmap, freemium-модель, що фінансує enterprise-функціональність
3	Вимоги до кібербезпеки та регуляторні обмеження	Невідповідність вимогам безпеки або стандартам може блокувати входження в окремі сектори	Впровадження сучасних практик безпеки (TLS, сертифікати, аудит), консалтинг із кібербезпеки, план сертифікації для критичних галузей

Фактори можливостей

Таблиця 4.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання LPWAN-ринку	Масове розгортання NB-IoT/LoRa-мереж, потреба в ефективних шлюзах для тисяч пристроїв	Запуск пілотів з операторами зв'язку й інтеграторами, пропозиція шлюзу як «референсного» компонента для їхніх IoT-платформ
2	Попит на SLO-орієнтовані рішення	Великі замовники переходять від «best effort» до формальних SLO/SLA, потребуючи вимірюваних рішень	Позиціонувати шлюз як інструмент досягнення/верифікації SLO, інтегрувати готові SLO-дашборди та звітність
3	Відкриті стандарти й open-source	MQTT, NB-IoT, LoRaWAN базуються на відкритих стандартах; можливість будувати екосистему навколо шлюзу	Відкрити частину коду (community-версія), розвивати спільноту, отримувати внески й підвищувати довіру до продукту
4	Академічні та пілотні проєкти	Університети, міські програми «розумного міста», грантові історії потребують гнучких, добре задокументованих рішень	Пропонувати шлюз як платформу для досліджень, розробити навчальні матеріали, спільні пілоти, що одночасно створюють референс-кейс і рекламу

Ступеневий аналіз конкуренції

Таблиця 4.9 – Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії)
1	Тип конкуренції: монополістична конкуренція	Є кілька домінуючих рішень (хмарні платформи, вендори), але жоден не покриває нішу адаптивного MQTT-шлюзу для NB-IoT/LoRa повністю	Фокус на вузькій ніші (адаптація на шлюзі), розробка унікального функціоналу, партнерські інтеграції з великими платформами замість прямої конфронтації
2	Рівень конкурентної боротьби: світовий	Конкуренція глобальна, більшість рішень орієнтовані на міжнародний ринок	Орієнтація на англомовну документацію, участь у міжнародних конфах/хакатонах, дистрибуція як контейнера чи SaaS
3	Галузева ознака: внутрішньогалузева	Продукт належить до сегменту IoT-платформ і мережевих шлюзів	Інтеграція з типовим стеком IoT (MQTT, HTTP, облачні сервіси), підтримка стандартів безпеки, тісна робота з інтеграторами

Кінець таблиці 4.9

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії)
4	Конкуренція за видами товарів: товарно-видова	Конкуренція відбувається між різними MQTT-шлюзами, хмарними брокерами та LPWAN-платформами	Впровадження функцій, яких бракує конкурентам (SLO-орієнтована адаптація, формальні інваріанти), спрощення інтеграції й експлуатації
5	Характер конкурентних переваг: цінова/нецінова	Конкуренти часто грають неціновими факторами: інтеграція, екосистема, зручність; цінова конкуренція можлива для стартових інтеграторів	Побудова конкурентної базової ціни + ставка на нецінові переваги (адаптація, прозорість, спільнота, навчальні матеріали)
6	За інтенсивністю: марочна	Сильний вплив бренду великих хмарних вендорів	Створення власного бренду в ніші (blog posts, статті, доповіді), підкреслення сумісності з хмарними платформами як «доповнення, а не заміна»

Аналіз конкуренції в галузі за М. Портером

Таблиця 4.10 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Ступінь конкуренції	MQTT-проксі та шлюзи, хмарні IoT-шлюзи	Нові open-source рішення, розширення хмарних сервісів	Хмарні провайдери, оператори NB-IoT/LoRa, вендори баз даних	Оператори зв’язку, інтегратори, промислові/енергетичні/міські замовники	Пряме підключення пристроїв до брокера, інші протоколи/стек без виділеного шлюзу

Висновок.

Рівень конкуренції високий, але ніша адаптивного MQTT-шлюзу для NB-IoT/LoRa з формальними SLO-гарантіями залишається недостатньо заповненою, що створює можливості для спеціалізованого гравця.

4.4 Розроблення ринкової стратегії проєкту

Обґрунтування факторів конкурентоспроможності

Таблиця 4.11 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Адаптивність до NB-IoT/LoRa	Продукт спеціально спроєктований для вузьких каналів, враховує RTT, втрати, duty-cycle, обмеження airtime
2	SLO-орієнтоване керування	Є чіткі SLO (latency, втрати, байти/повідомлення) та SLI, а політика адаптації націлена на підтримання цих цілей
3	Вбудована спостережуваність	Повний стек метрик (RTT, втрати, черга, формат, QoS, інтенсивність), готові дашборди Grafana
4	Формальна модель політик	Алгебра політик і FRP-модель забезпечують прозору, перевірювану логіку, що важливо для критичних застосунків
5	Інтеграція з існуючим MQTT-стеком	Працює поверх стандартних MQTT-брокерів, не вимагає їх модифікації; легко вбудовується в уже наявні IoT-архітектури
6	Гнучка вартість володіння	Можливість базової безкоштовної версії (community), платна підтримка/enterprise-функціонал, що знижує бар’єр входу
7	Масштабованість	Може працювати у контейнерному середовищі, горизонтально масштабуватися відповідно до навантаження
8	Якість підтримки	Фокус на спеціалізованій експертній підтримці NB-IoT/LoRa та MQTT v5
9	Документація та навчальні матеріали	Наявність прикладів, сценаріїв розгортання, інструкцій для операторів і інтеграторів

Кінець таблиці 4.11

№	Фактор конкурентоспроможності	Обґрунтування
10	Відкритість до інтеграцій та кастомізації	Відкриті API, можливість налаштування політик під конкретні сценарії замовника

Порівняльний аналіз сильних і слабких сторін

Таблиця 4.12 – Порівняльний аналіз сильних та слабких сторін адаптивного MQTT-шлюзу

(«Бали 1–20» – оцінка мого проєкту; у дужках – відносна оцінка статичного шлюзу / хмарного шлюзу у шкалі −3…+3.)

№	Фактор конкурентоспроможності	Бали 1–20 (мій проєкт)	Рейтинг конкурентів (статичний шлюз / хмарний шлюз)
1	Адаптація до вузьких каналів	18	(−2 / −1)
2	SLO-орієнтованість	17	(−2 / 0)
3	Інтеграція з MQTT-екосистемою	17	(+1 / +2)
4	Спостережуваність	18	(−1 / 0)
5	Гнучкість ліцензування	16	(0 / −2)
6	Масштабованість	16	(+1 / +2)
7	Простота розгортання	17	(+1 / 0)
8	Кастомізація політик	18	(−1 / −1)
9	Безпека та відповідність стандартам	15	(+1 / +2)
10	Бренд/відомість	12	(+2 / +3)

SWOT-аналіз стартап-проєкту

Таблиця 4.13 – SWOT-аналіз стартап-проєкту

Сильні сторони (S):	Слабкі сторони (W):
– спеціалізація на NB-IoT/LoRa та вузьких каналах; – SLO-орієнтовані політики адаптації; – формальна модель (алгебра політик, FRP), що спрощує верифікацію; – прозора спостережуваність (Prometheus/Grafana, повний SLI-набір); – гнучке розгортання (контейнери, хмара, on-premise).	– відсутність розкрученого бренду на старті; – обмежені ресурси на маркетинг та сертифікацію; – потреба у технічній компетенції замовника (IoT/LPWAN).
Можливості (O):	Загрози (T):
– стрімке зростання LPWAN-сегмента; – попит на рішення з формальними SLO/SLA; – потенціал open-source-спільноти; – пілотні проєкти смарт-інфраструктури, грантові програми.	– агресивний вихід великих хмарних вендорів у нішу; – зміни стандартів/регуляцій (безпека, приватність даних); – швидке копіювання ідей у відкритому середовищі.

Альтернативи ринкового впровадження

Таблиця 4.14 – Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовна community-версія (обмежений функціонал, відсутня підтримка)	Людський ресурс уже наявний	2–3 місяці
2	Цілеспрямований контент-маркетинг (статті, доповіді, демо-проєкти)	Потрібен час команди, бюджет мінімальний	1–3 місяці
3	Пілотні впровадження з операторами та інтеграторами	Потребує часу та часткового фінансування з боку партнерів	3–6 місяців
4	Участь у хакатонах, інкубаторах, акселераторах IoT/5G	Час і невеликі кошти на участь	1–3 місяці

Вибір цільових груп потенційних споживачів

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи	Готовність споживачів сприйняти продукт	Орієнтовний попит	Інтенсивність конкуренції	Простота входу у сегмент
1	Оператори зв’язку та великі інтегратори IoT	Висока технічна зрілість, але консервативність у виборі рішень	Високий (великі проєкти)	Висока	Складно (тендери, довіра)
2	Промислові/енергетичні підприємства, «розумні» міста	Середня готовність, орієнтація на TCO та SLA	Середній-високий	Середня	Середньої складності

Які цільові групи обрано.

Основна цільова група — інтегратори IoT-рішень та оператори NB-IoT/LoRa, які вже мають інфраструктуру, але потребують адаптивного шлюзу. Додаткова група — промислові/енергетичні підприємства та муніципалітети, що запускають LPWAN-проєкти і хочуть мінімізувати ризики з QoS.

Стратегія охоплення — масовий маркетинг у межах професійної спільноти: стандартизований продукт з можливістю кастомізації політик під конкретні SLO.

Визначення базової стратегії розвитку

Таблиця 4.16 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
Nadання адаптивного шлюзу з SLO-орієнтованим керуванням для NB-IoT/LoRa-проектів	Масове охоплення в професійній IoT-спільноті	Спеціалізація на вузьких каналах, формальні інваріанти, прозора аналітика	Стратегія диференціації

Визначення базової стратегії конкурентної поведінки

Таблиця 4.17 – Визначення базової стратегії конкурентної поведінки

Питання	Відповідь
Чи є проєкт «першопрохідцем» на ринку?	Ні, існують інші шлюзи та платформи, але немає спеціалізованого адаптивного MQTT-шлюзу для NB-IoT/LoRa з формальними SLO-інваріантами
Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Обидва варіанти: нові IoT-проєкти + частковий перехід існуючих клієнтів статичних шлюзів, що відчувають проблеми з QoS
Чи буде компанія копіювати характеристики товару конкурента?	Частково: підтримка стандартних функцій (MQTT, TLS), але акцент на унікальній адаптаційній логіці, відкритій спостережуваності та формальній моделі політик
Стратегія конкурентної поведінки	Стратегія заняття конкурентної ніші в сегменті LPWAN-шлюзів

Визначення стратегії позиціонування

Таблиця 4.18 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції	Вибір асоціацій для позиціонування
1	Стабільна доставка телеметрії у вузьких каналах	Диференціація	Адаптивні SLO-політики, контроль черг, зменшення втрат	«Надійність», «контроль», «стабільність»
2	Прозорість та вимірюваність якості обслуговування	Диференціація	Повний набір SLI, дашборди Grafana, відтворювана методика тестів	«Прозора чорна скринька», «все видно, все вимірюється»
3	Гнучка інтеграція й доступна вартість	Диференціація + фокусування	Контейнеризація, відкриті API, freemium-модель	«Легко інтегрувати», «почати просто», «pay-as-you-grow»

4.5 Розроблення маркетингової програми стартап-проєкту

Визначення ключових переваг концепції потенційного товару

Таблиця 4.19 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Стабільна доставка телеметрії у вузьких каналах	Зменшення латентності та втрат за рахунок адаптації параметрів	SLO-орієнтована політика, формальні інваріанти, контроль черг
2	Економія трафіку та airtime	Зменшення байтів/повідомлення та частоти відправлення	Δ-кодування, CBOR/Protobuf, пакетування, автоматичний вибір формату
3	Просте впровадження у наявну інфраструктуру	Standalone-шлюз, що працює з існуючим MQTT-брокером	Відсутність жорсткої прив’язки до конкретного хмарного вендора
4	Прозорий моніторинг та аудит	Готові дашборди, метрики, відтворювані експерименти	Повний стек SLI/політик, сценарії тестів, зручність для операторів

Опис трьох рівнів моделі товару

Таблиця 4.20 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	Адаптивний MQTT-шлюз для вузьких IoT-каналів (NB-IoT/LoRa), призначений для надійної, ефективної доставки телеметрії з формальним контролем якості обслуговування.
2. Товар у реальному виконанні	Властивості/характеристики
	підтримка MQTT v5, інтеграція з NB-IoT/LoRa-шлюзами;
	модуль політик Encode/Δ/Batch/Throttle/QoS/Timeout;
	SLO-орієнтоване керування (RTT, втрати, черги, байти/повідомлення);
	веб-консоль і дашборди Grafana;
	розгортання у Docker/Kubernetes;
	конфігуровані профілі для NB-IoT/LoRa-сценаріїв.
3. Товар із підкріпленням	Після продажу: документація, приклади розгортання, демо-стенди; trial-ліцензія для пілотів; навчальні матеріали для команд замовника.
	Після продажу: технічна підтримка й супровід;оновлення політик, розширення функціональності; консалтинг щодо SLO/SLA та профілювання каналів.
Захист від копіювання: ліцензійна угода, авторське право, опціонально — патентування окремих алгоритмів.	

Визначення меж встановлення ціни

Таблиця 4.21 – Визначення меж встановлення ціни

Показник	Характеристика
Рівень цін на товари-замінники	Частина шлюзів безкоштовна (open-source), частина входить до складу дорогих IoT-платформ
Рівень цін на товари-аналоги	Комерційні IoT-платформи застосовують підписку/плату за пристрій
Рівень доходів цільової групи	Оператори/промисловість/інтегратори мають бюджети на інфраструктурне ПЗ
Верхня та нижня межі встановлення ціни	Нижня – free/community-версія; верхня – enterprise-ліцензія та підтримка, співставні з нішевими IoT-компонентами

Формування системи збуту

Таблиця 4.22 – Формування системи збуту

Питання	Характеристика
Специфіка закупівельної поведінки	Для великих – тендери та пілотні проєкти; для менших – прямі договори, рекомендації спільноти
Функції збуту, які має виконувати постачальник	Надання демо/пілотів, технічні консультації, інтеграційна підтримка, навчання
Глибина каналу збуту	Переважно прямий канал + партнерські інтегратори
Оптимальна система збуту	Комбінація on-line дистрибуції (контейнери, репозиторії) та партнерств з інтеграторами й операторами

Концепція маркетингових комунікацій

Таблиця 4.23 – Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій	Ключові позиції для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Розробники та DevOps у IoT-командах	Професійні форуми, GitHub, Telegram-/Slack-спільноти	«Адаптивний», «видимий», «простий у інтеграції»	Пояснити, що більшість проблем з NB-IoT/LoRa можна закрити на шлюзі	Короткі технічні кейси, графіки до/після, приклади конфігурацій
2	Архітектори, керівники напрямів IoT, СТО	Конференції, вебінари, галузеві медіа	«Зменшує ризики», «допомагає виконувати SLA», «економить»	Показати економію трафіку/витрат і зниження ризиків невиконання SLA	Інфографіка з економією, історії успіху пілотів

Результат.

Сформовано ринкову (маркетингову) програму, що включає: визначення цільових груп, ключових переваг, моделі товару, меж ціноутворення, системи збуту та концепцію комунікацій.

Висновки до розділу 4

У четвертому розділі виконано маркетинговий аналіз стартап-проєкту адаптивного MQTT-шлюзу для вузьких IoT-каналів (NB-IoT/LoRa). На підставі аналізу ринку, конкурентного середовища та цільових сегментів обґрунтовано доцільність виходу з нішевим продуктом, орієнтованим на SLO-орієнтовану адаптацію телеметрії.

Показано, що:

1) ринок IoT-платформ і LPWAN-рішень демонструє стале зростання та має незаповнену нішу спеціалізованих адаптивних шлюзів;
2) ключові цільові групи (оператори, інтегратори, промислові/енергетичні замовники) мають чітко сформований запит на стабільну доставку даних і контрольовані SLO;
3) конкурентні переваги проєкту полягають у поєднанні адаптивності, формальної моделі політик, спостережуваності та гнучкого розгортання.

Запропоновано стратегію диференціації з орієнтацією на зайняття конкурентної ніші в сегменті LPWAN-шлюзів, сформовано маркетингову програму й альтернативи впровадження. Отже, з точки зору ринкової доцільності та перспектив комерціалізації, розробка адаптивного MQTT-шлюзу для NB-IoT/LoRa є перспективним і обґрунтованим напрямом подальшої роботи.

ВИСНОВКИ

У магістерській дисертації вирішено актуальну науково-технічну задачу підвищення ефективності та надійності доставки IoT-телеметрії у вузькосмугових каналах зв'язку (NB-IoT/LoRa) шляхом розробки адаптивного MQTT-шлюзу на основі функціональної алгебри політик та SLO-орієнтованого керування.

Основні наукові та практичні результати роботи:

- Проведено аналітичний огляд технологій LPWAN (NB-IoT, LoRaWAN), протоколів MQTT/MQTT-SN/CoAP, механізмів ADR та формальних підходів до адаптації. Виявлено прогалини існуючих рішень: ізольованість адаптаційних механізмів, відсутність формальних гарантій стабільності, неврахування SLO/SLI.
- Розроблено функціональну алгебру політик адаптації з операторами Encode, Delta, Batch, Throttle, QoS, Guard, Choose, Timeout. Визначено денотаційну семантику та алгебраїчні закони (ідемпотентність, комутативність, асоціативність).
- Побудовано FRP-модель оцінювання стану каналу (RTT, loss, queue) на основі EWMA-сигналів. Розроблено механізм контрактних оновлень для SLO-орієнтованого керування з гарантією збіжності.
- Сформульовано та обґрунтовано інваріанти стабільності: обмежена черга, відсутність осциляцій, backpressure preservation, монотонне відновлення.
- Реалізовано адаптивний MQTT-шлюз на базі Scala/ZIO з інтеграцією MQTT v5 (Receive Maximum, Maximum Packet Size, Topic Alias, Message Expiry). Архітектура побудована за принципами FRP з чітким розділенням компонентів.
- Розроблено відтворювану методику експериментів з емуляцією профілів NB-IoT/LoRaWAN через tc netem. Експериментально підтверджено ефективність: Зменшено середній час прийому-передачі приблизно на 29

%. Зменшено максимальний час прийому-передачі майже на 46 %. Підвищено корисну пропускну здатність за кількістю повідомлень приблизно на 1,9 % при майже незмінному обсязі переданих даних..

Наукова новизна одержаних результатів:

– Удосконалено методи потокової доставки телеметрії у constrained-мережах шляхом використання функціональної декомпозиції та чистих політик адаптації з формальними інваріантами backpressure, що відрізняється від існуючих підходів композиційністю та верифікованістю.
– Набуло подальшого розвитку підходи до SLO-орієнтованого керування потоком, що підвищують ефективність розроблення та модифікації програмного забезпечення для IoT завдяки використанню контрактних оновлень із гарантією збіжності.

Практичне значення одержаних результатів:

– Запропонована референсна архітектура адаптивного шлюзу придатна для інтеграції у смарт-інфраструктуру (розумні будівлі, агромоніторинг, промислова автоматизація).
– Розроблена відтворювана методика експериментальної оцінки дозволяє порівнювати різні конфігурації адаптації у контрольованих умовах.
– Результати можуть бути використані у навчальному процесі для підготовки фахівців у галузі IoT та розподілених систем.
– Перспективи подальших досліджень:
– Розширення алгебри політик додатковими операторами (шифрування, пріоритизація);
– Інтеграція з методами машинного навчання для предиктивної адаптації;
– Розширення на інші LPWAN-технології (Sigfox, LTE-M);
– Автоматична генерація оптимальних політик на основі SLO-специфікацій.

Таким чином, мета магістерської дисертації досягнута. Розроблений адаптивний MQTT-шлюз демонструє суттєве покращення ключових показників ефективності у порівнянні зі статичними підходами та може бути рекомендований для впровадження у реальних IoT-системах з вузькосмуговими каналами зв'язку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Larmo A., Ratilainen A., Saarinen J. Impact of CoAP and MQTT on NB-IoT System Performance [Електронний ресурс] / A. Larmo, A. Ratilainen, J. Saarinen // Sensors. — 2019. — Vol. 19, № 1. — Art. 7. — Режим доступу : https://doi.org/10.3390/s19010007 (дата звернення: 04.12.2025).
2) Mekki K., Bajic E., Chaxel F., Meyer F. A comparative study of LPWAN technologies for large-scale IoT deployment [Електронний ресурс] / K. Mekki, E. Bajic, F. Chaxel, F. Meyer // ICT Express. — 2019. — Режим доступу : https://www.sciencedirect.com/science/article/pii/S2405959517302953 (дата звернення: 04.12.2025).
3) Palmese F., Redondi A. E. C., Cesana M. Adaptive Quality of Service Control for MQTT-SN [Електронний ресурс] / F. Palmese, A. E. C. Redondi, M. Cesana // Sensors. — 2022. — Vol. 22, № 22. — Art. 8852. — Режим доступу : https://doi.org/10.3390/s22228852 (дата звернення: 04.12.2025).
4) Kousias K., Caso G., Alay Ö., Lemic F. Empirical Analysis of LoRaWAN Adaptive Data Rate for Mobile Internet of Things Applications [Електронний ресурс] / K. Kousias, G. Caso, Ö. Alay, F. Lemic // Wireless of the Students, by the Students, and for the Students Workshop (S3’19). — Los Cabos, Mexico, 2019. — Режим доступу : https://dl.acm.org/doi/10.1145/3355727 (дата звернення: 04.12.2025).
5) Benkahla N., Tounsi H., Song Y.-Q., Frikha M. Review and experimental evaluation of ADR enhancements for LoRaWAN networks [Електронний ресурс] / N. Benkahla та ін. // Telecommunication Systems. — 2021. — Режим доступу : https://doi.org/10.1007/s11235-020-00738-x (дата звернення: 04.12.2025).
6) Reactive: Push-pull functional reactive programming [Електронний ресурс]. — Hackage. — Режим доступу : https://hackage.haskell.org/package/reactive (дата звернення: 04.12.2025).

7) Coutts D. Stream fusion: practical shortcut fusion for coinductive sequence types : PhD thesis [Електронний ресурс] / D. Coutts. — University of Oxford, 2011. — Режим доступу : https://ora.ox.ac.uk/objects/uuid:b4971f57-2b94-4fdf-a5c0-98d6935a44da (дата звернення: 04.12.2025).
8) Coutts D., Leshchinskiy R., Stewart D. Exploiting Vector Instructions with Generalized Stream Fusion [Електронний ресурс] / D. Coutts, R. Leshchinskiy, D. Stewart. — 2007. — Режим доступу : https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/haskell-beats-C.pdf (дата звернення: 04.12.2025).
9) Coutts D., Leshchinskiy R., Stewart D. Stream fusion: from lists to streams to nothing at all [Електронний ресурс] / D. Coutts, R. Leshchinskiy, D. Stewart. — 2007. — Режим доступу : https://dl.acm.org/doi/10.1145/1291220.1291199 (дата звернення: 04.12.2025).
10) Kozen D., Smith F. Kleene algebra with tests: completeness and decidability [Електронний ресурс] / D. Kozen, F. Smith. — In: Computer Science Logic, 1997. — Режим доступу : https://www.cs.cornell.edu/~kozen/Papers/kat.pdf (дата звернення: 04.12.2025).
11) NetKAT – Network Programming Language [Електронний ресурс]. — Режим доступу : https://netkat.org/ (дата звернення: 04.12.2025).
12) Anderson C. J., Foster N., Guha A. та ін. NetKAT: semantic foundations for networks [Електронний ресурс] / C. J. Anderson та ін. // POPL 2014: Proceedings of the 41st ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages. — 2014. — Режим доступу : https://doi.org/10.1145/2535838.2535862 (дата звернення: 04.12.2025).
13) Foster N., Harrison R., Meola M. L. та ін. Frenetic: A High-Level Language for OpenFlow Networks [Електронний ресурс] / N. Foster та ін. — Technical

Report / Conference paper. — 2010. — Режим доступу : https://frenetic-lang.org/publications/frenetic-presto10.pdf (дата звернення: 04.12.2025).

14) Foster N., Harrison R., Freedman M. J. та ін. Frenetic: a network programming language [Електронний ресурс] / N. Foster та ін. // Proc. of the 16th ACM SIGPLAN Int. Conf. on Functional Programming (ICFP). — 2011. — Режим доступу : https://dl.acm.org/doi/10.1145/2034574.2034812 (дата звернення: 04.12.2025).

15) Wang K., Schwarzenberg C., Wiedner F. Survey on Back-Pressure Based Routing [Електронний ресурс] / K. Wang, C. Schwarzenberg, F. Wiedner // NET-2022-01-1 : Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM), Summer Semester 2021. — Munich, 2022. — Режим доступу : https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2022-01-1/NET-2022-01-1_14.pdf (дата звернення: 04.12.2025).

16) Selected Publications – Dalvan Griebler [Електронний ресурс]. — Режим доступу : https://www.dalvangriebler.com/selected-publications/ (дата звернення: 04.12.2025).

17) Vogel A., Mencagli G., Griebler D., Danelutto M., Fernandes L. G. Online and transparent self-adaptation of stream parallel patterns [Електронний ресурс] / A. Vogel та ін. // Computing. — 2023. — Режим доступу : https://pages.di.unipi.it/mencagli/downloads/Preprint-Computing-2021.pdf (дата звернення: 04.12.2025).

18) Concise Binary Object Representation (CBOR). Overview [Електронний ресурс]. — Режим доступу : https://cbor.io/ (дата звернення: 04.12.2025).

19) Protocol Buffers Documentation. Overview [Електронний ресурс]. — Режим доступу : https://protobuf.dev/overview/ (дата звернення: 04.12.2025).

20) JMX Exporter : A process for collecting metrics using JMX MBeans for Prometheus consumption [Електронний ресурс]. — GitHub. — Режим доступу : https://github.com/prometheus/jmx_exporter (дата звернення: 04.12.2025).

21) Grafana fundamentals : tutorial [Електронний ресурс]. — Grafana Labs. — Режим доступу : https://grafana.com/tutorials/grafana-fundamentals/ (дата звернення: 04.12.2025).

22) tc-netem(8) — Linux manual page [Електронний ресурс]. — Режим доступу : https://man7.org/linux/man-pages/man8/tc-netem.8.html (дата звернення: 04.12.2025).

23) Introduction · doobie [Електронний ресурс]. — Typelevel. — Режим доступу : https://typelevel.org/doobie/docs/01-Introduction.html (дата звернення: 04.12.2025).

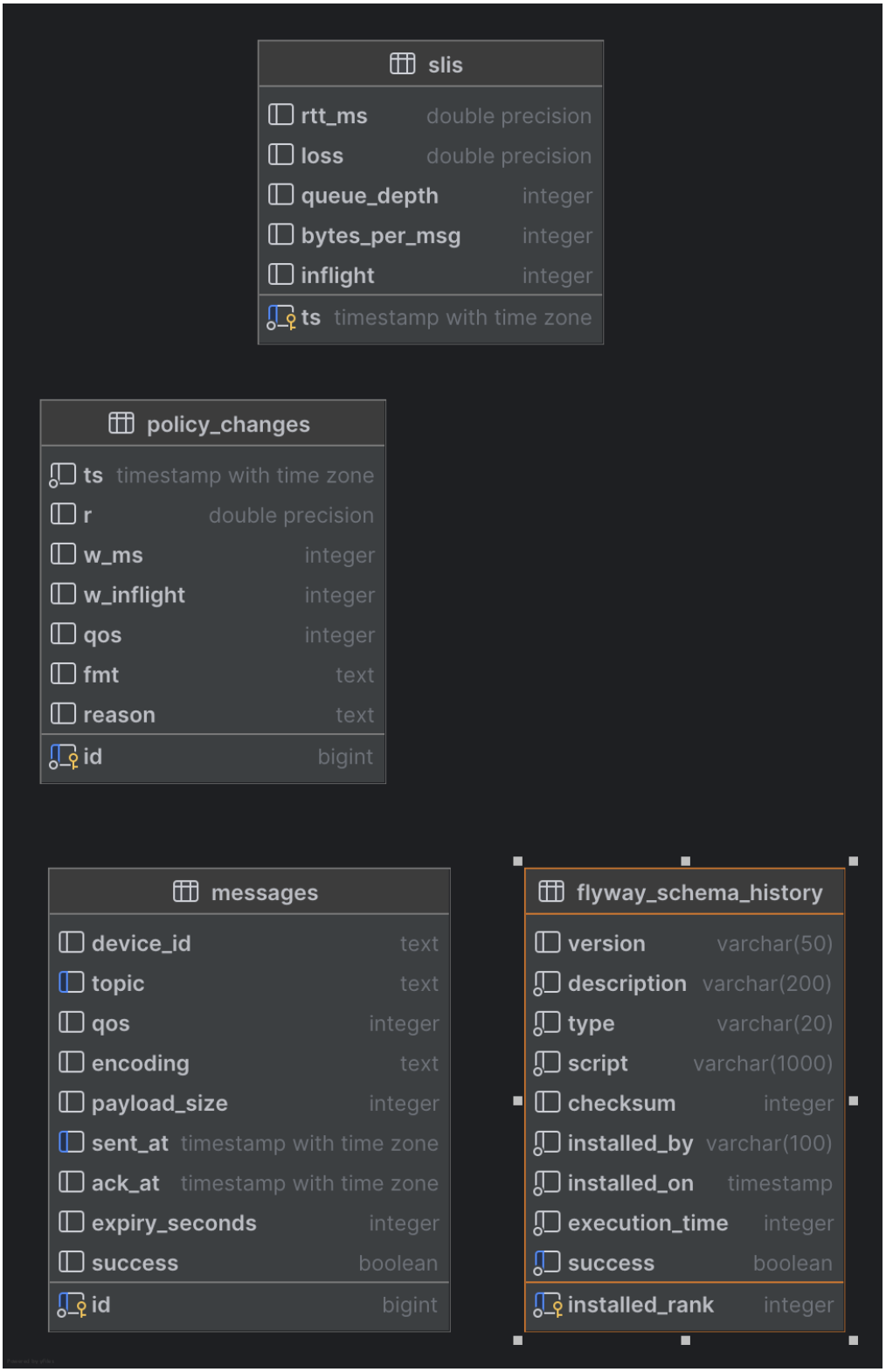
24) ZIO Metrics legacy : high-performance functional metrics library [Електронний ресурс]. — GitHub. — Режим доступу : https://github.com/zio-archive/zio-metrics-legacy (дата звернення: 04.12.2025).

25) Srivastava A. Type-Safe REST services in Scala with Http4s & Cats-IO [Електронний ресурс] / A. Srivastava // Walmart Global Tech Blog (Medium). — 2020. — Режим доступу : https://medium.com/walmartglobaltech/type-safe-rest-services-in-scala-with-http4s-cats-io-288d6e23a90a (дата звернення: 04.12.2025).

ДОДАТКИ

ДОДАТОК А

Схема бази даних



ДОДАТОК Б

UML діаграма класів

ДОДАТОК В

Лістинг коду

В.1. Сценарій загального розкладу сегментів A/B-тесту

У цьому підрозділі наведено фрагменти сценарію `ops/comparison/common_schedule.sh`, який визначає спільний розклад сегментів для Run A (fixed policy) та Run B (adaptive policy). Сценарій задає тривалість сегментів, керує мережевими умовами через `tc netem`, запускає генератор навантаження та викликає процедури валідації SLO.

```bash
#!/usr/bin/env bash
set -euo pipefail

# Common Test Schedule
# Runs 5 segments with varying network conditions and workloads
# Used by both run_fixed.sh and run_adaptive.sh

DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
PROJECT_ROOT="$(cd "$DIR/../.." && pwd)"
NETEM_DIR="$PROJECT_ROOT/ops/netem"

# Configuration
MQTT_CONTAINER="${MQTT_CONTAINER:-mosquitto}"
NUM_DEVICES=200
```

```
DEVICE_RATE=5  # messages per second per device

# Segment durations (seconds)
WARMUP_DURATION=120
STRESS_DURATION=240
OUTAGE_DURATION=60
RECOVERY_DURATION=180
INFLATION_DURATION=120

# Payload inflation factor (multiplies JSON size)
PAYLOAD_INFLATION_FACTOR="${PAYLOAD_INFLATION_FACTOR:-1}"

# Apply stress network conditions (20% loss, 1000ms RTT)
apply_stress_netem() {
  print_info "Applying network stress: 1000ms delay, 200ms jitter, 20% loss"
    bash "$NETEM_DIR/apply_netem.sh" "$MQTT_CONTAINER" "1000ms" "20%"
}

# Apply complete outage (100% loss)
apply_outage_netem() {
  print_info "Applying complete outage: 100% loss"
  bash "$NETEM_DIR/apply_netem.sh" "$MQTT_CONTAINER" "0ms" "100%"
}

restore_network() {
```

```
  print_info "Restoring normal network conditions"
  bash "$NETEM_DIR/restore.sh" "$MQTT_CONTAINER"
 }

 run_segment() {
  local segment_name=$1
  local duration=$2
  local inflation_factor=${3:-1}
  local segment_tag=${4:-segment}

  print_segment "$segment_name (${duration}s)"

  # Add annotation for segment start (used in Grafana)
   bash "$DIR/add_annotation.sh" "$segment_name started" "$segment_tag"
2>/dev/null || true

  start_publishers "$duration" "$inflation_factor"

  # Countdown for operator
  for i in $(seq 1 $duration); do
   local percent=$((i * 100 / duration))
   echo -ne "\r[$(date '+%H:%M:%S')] Progress: ${i}/${duration}s (${percent}%)"
   sleep 1
  done
  echo ""

  wait_publishers
```

```
  print_success "Segment completed"

  print_info "Waiting 10s for metrics to stabilize..."
  sleep 10
}

run_schedule() {
  local run_name=$1

  print_header "$run_name - Full Test Schedule"
  echo "  0. Warm-up:          ${WARMUP_DURATION}s  (baseline)"
  echo "  1. Step Stress:      ${STRESS_DURATION}s  (20% loss, 1000ms RTT)"
  echo "  2. Outage:           ${OUTAGE_DURATION}s  (100% loss)"
  echo "  3. Recovery:         ${RECOVERY_DURATION}s  (network restored)"
  echo "  4. Payload Inflation: ${INFLATION_DURATION}s  (large payloads)"

  # Ensure clean state
  restore_network
  sleep 2

  # Segment 0: Warm-up (baseline)
  restore_network
  sleep 1
  run_segment "Segment 0: Warm-up" "$WARMUP_DURATION" "1" "baseline"
  validate_warmup || true
```

```
        # Segment 1: Step Stress (loss + RTT)
        apply_stress_netem
        sleep 2
        run_segment "Segment 1: Step Stress (20% loss, 1000ms RTT)" \
          "$STRESS_DURATION" "1" "loss-spike,rtt-surge"
        validate_stress || true

        # Segment 2: Complete Outage
        apply_outage_netem
        sleep 2
        run_segment "Segment 2: Complete Outage (100% loss)" \
          "$OUTAGE_DURATION" "1" "outage"
        validate_outage || true

        # Segment 3: Recovery
        restore_network
        sleep 2
         run_segment "Segment 3: Recovery" "$RECOVERY_DURATION" "1"
"recovery"
        validate_recovery || true

        # Segment 4: Payload Inflation (3x payload size)
        run_segment "Segment 4: Payload Inflation (3x size)" \
          "$INFLATION_DURATION" "3" "payload-growth"
        validate_payload_inflation || true

        # Cleanup network conditions
```

```
  restore_network

  print_test_summary || true
}
```

Наведений лістинг демонструє, як у єдиному сценарії поєднуються керування мережевими умовами, генерація навантаження, анотації для Grafana та виклики функцій валідації SLO після кожного сегмента.

В.2. Сценарій повного A/B-порівняння та збирання метрик

Наступний фрагмент показує ключову частину сценарію `ops/comparison/run_full_comparison.sh`, відповідальну за збирання агрегованих метрик з Prometheus для Run A (fixed policy) і Run B (adaptive policy) та формування JSON-файлів з результатами випробувань.

```bash
# Metric Collection from Prometheus

query_prometheus_range_rate() {
  local metric=$1
  local start_ts=$2
  local end_ts=$3
  local duration=$((end_ts - start_ts))
  local rate_window=$([[ $duration -lt 300 ]] && echo "5m" || echo "${duration}s")
  local query="rate(${metric}[${rate_window}])"
```

```
  local result=$(curl -s -G "${PROMETHEUS_URL}/api/v1/query" \
    --data-urlencode "query=${query}" \
    --data-urlencode "time=${end_ts}" | jq -r '.data.result[0].value[1] // "0"')
  echo "$result"
}

query_prometheus_range_avg() {
  local metric=$1
  local start_ts=$2
  local end_ts=$3
  local duration=$((end_ts - start_ts))
  local query="avg_over_time(${metric}[${duration}s])"

  local result=$(curl -s -G "${PROMETHEUS_URL}/api/v1/query" \
    --data-urlencode "query=${query}" \
    --data-urlencode "time=${end_ts}" | jq -r '.data.result[0].value[1] // "0"')
  echo "$result"
}

query_prometheus_range_histogram_quantile() {
  local quantile=$1
  local metric=$2
  local start_ts=$3
  local end_ts=$4
  local duration=$((end_ts - start_ts))
  local rate_window=$([[ $duration -lt 300 ]] && echo "5m" || echo "${duration}s")
```

```
                local query="histogram_quantile(${quantile}, rate(${metric}_bucket[${rate_window}]))"

    local result=$(curl -s -G "${PROMETHEUS_URL}/api/v1/query" \
      --data-urlencode "query=${query}" \
      --data-urlencode "time=${end_ts}" | jq -r '.data.result[0].value[1] // "0"')
    echo "$result"
  }

  collect_metrics() {
    local run_name=$1
    local start_time=$2
    local end_time=$3

    local duration=$((end_time - start_time))
     local  mqtt_publish_rate=$(query_prometheus_range_rate  "mqtt_publish_total" "$start_time" "$end_time")
          local messages_published_rate=$(query_prometheus_range_rate "messages_published_total" "$start_time" "$end_time")
           local bytes_received_rate=$(query_prometheus_range_rate "bytes_received_total" "$start_time" "$end_time")
          local bytes_published_rate=$(query_prometheus_range_rate "bytes_published_total" "$start_time" "$end_time")

     local avg_queue_depth=$(query_prometheus_range_avg "slo_queue_depth" "$start_time" "$end_time")
```

```
    local avg_bytes_per_msg=$(query_prometheus_range_avg "slo_bytes_per_msg" "$start_time" "$end_time")
    local avg_rtt_ms=$(query_prometheus_range_avg "slo_rtt_ewma_ms" "$start_time" "$end_time")

    local p50_ack_latency=$(query_prometheus_range_histogram_quantile "0.5" "mqtt_ack_latency_ms" "$start_time" "$end_time")
    local p95_ack_latency=$(query_prometheus_range_histogram_quantile "0.95" "mqtt_ack_latency_ms" "$start_time" "$end_time")
    local p99_ack_latency=$(query_prometheus_range_histogram_quantile "0.99" "mqtt_ack_latency_ms" "$start_time" "$end_time")

    local traffic_reduction_ratio=$(echo "scale=4;
      if ($bytes_received_rate > 0)
        1 - ($bytes_published_rate / $bytes_received_rate)
      else 0" | bc)

    cat > "${RESULTS_DIR}/${run_name}_metrics.json" <<EOF
    {
      "run_name": "${run_name}",
      "test_duration_seconds": ${duration},
      "throughput": {
        "mqtt_publish_rate": ${mqtt_publish_rate},
        "messages_published_rate": ${messages_published_rate},
        "bytes_received_rate": ${bytes_received_rate},
        "bytes_published_rate": ${bytes_published_rate},
        "traffic_reduction_ratio": ${traffic_reduction_ratio}
```

```
  },
  "slo_metrics": {
    "avg_bytes_per_msg": ${avg_bytes_per_msg},
    "avg_rtt_ms": ${avg_rtt_ms},
    "avg_queue_depth": ${avg_queue_depth}
  },
  "latency": {
    "p50_ack_latency_ms": ${p50_ack_latency},
    "p95_ack_latency_ms": ${p95_ack_latency},
    "p99_ack_latency_ms": ${p99_ack_latency}
  }
}
EOF

        echo "[METRICS] Metrics saved to ${RESULTS_DIR}/${run_name}_metrics.json"
}
```

Отримані JSON-файли для Run A (`fixed_policy_metrics.json`) і Run B (`adaptive_policy_metrics.json`) далі використовуються для автоматичного формування порівняльного текстового звіту та для побудови узагальнених таблиць у Додатку Е.

В.3. Алгебра політик та інтерпретатор конвеєра

У цьому підрозділі наведено ключові фрагменти алгебри політик та її інтерпретатора, реалізовані у файлі `src/main/scala/iot/policy/Policy.scala`. Алгебра описує конвеєр чистих перетворень над потоком телеметрії: кодування, диференціювання (delta), пакетування, обмеження швидкості, встановлення рівня надійності та тайм-аутів.

```scala
package iot.policy

import zio._
import zio.stream._
import iot.domain._
import iot.ser.json.CirceCodecs
import iot.ser.cbor.CborCodecs
import iot.ser.pb.PbCodecs

// Алгебра політик та оператор композиції
sealed trait Policy[I, O] { self =>
  def >>>[O2](that: Policy[O, O2]): Policy[I, O2] =
    Policy.Compose(this, that)

  def fold[F[_, _]](alg: PolicyF[F]): F[I, O]
}

object Policy {
  final case class Encode(fmt: EncodingFmt) extends Policy[Telemetry, Encoded]
{
```

```
  def fold[F[_, _]](alg: PolicyF[F]): F[Telemetry, Encoded] = alg.encode(fmt)
 }

 final case class Delta[A](diff: (A, A) => Option[A]) extends Policy[A, A] {
  def fold[F[_, _]](alg: PolicyF[F]): F[A, A] = alg.delta(diff)
 }

 final case class Batch[A](maxN: Int, withinMs: Long, agg: Chunk[A] => A) extends Policy[A, A] {
  def fold[F[_, _]](alg: PolicyF[F]): F[A, A] = alg.batch(maxN, withinMs, agg)
 }

 final case class Throttle[A](ratePerSec: Double) extends Policy[A, A] {
  def fold[F[_, _]](alg: PolicyF[F]): F[A, A] = alg.throttle(ratePerSec)
 }

 final case class QoSLevel[A](level: Int) extends Policy[A, (A, Int)] {
  def fold[F[_, _]](alg: PolicyF[F]): F[A, (A, Int)] = alg.qosLevel(level)
 }

 final case class Timeout[A, B](seconds: Int) extends Policy[(A, B), (A, B, Option[Int])] {
    def fold[F[_, _]](alg: PolicyF[F]): F[(A, B), (A, B, Option[Int])] = alg.timeout(seconds)
 }
```

```
    final case class Compose[A, B, C](p1: Policy[A, B], p2: Policy[B, C]) extends Policy[A, C] {
      def fold[F[_, _]](alg: PolicyF[F]): F[A, C] =
        alg.compose(p1.fold(alg), p2.fold(alg))
    }
  }

  // Інтерпретатор алгебри до потокового конвеєра ZStream
  object Interpreter {
    type SF[I, O] = ZStream[Any, Throwable, I] => ZStream[Any, Throwable, O]

    private object StreamAlg extends PolicyF[SF] {
      def encode(fmt: EncodingFmt): SF[Telemetry, Encoded] =
        in =>
          in.map { t =>
            fmt match {
              case EncodingFmt.Json     => CirceCodecs.toJson(t)
              case EncodingFmt.Cbor     => CborCodecs.toCbor(t)
              case EncodingFmt.Protobuf => PbCodecs.toProto(t)
            }
          }

      def delta[A](diff: (A, A) => Option[A]): SF[A, A] =
        in =>
          in.mapAccum(Option.empty[A]) { (last, cur) =>
            val out = last match {
              case None       => Some(cur)       // перший елемент завжди передається
```

```
        case Some(prev) => diff(prev, cur) // надалі – лише зміни
      }
      (Some(cur), out)
    }.collect { case Some(v) => v }

def batch[A](maxN: Int, withinMs: Long, agg: Chunk[A] => A): SF[A, A] =
  in => in.groupedWithin(maxN, Duration.fromMillis(withinMs)).map(agg)

def throttle[A](ratePerSec: Double): SF[A, A] =
  if (ratePerSec <= 0.0) _ => ZStream.empty
  else
    in =>
      in.via(
        ZPipeline.throttleShape(
          units = ratePerSec.toLong,
          duration = 1.second,
          burst = math.max(1, (ratePerSec * 2).toLong)
        )(_ => 1)
      )

def qosLevel[A](level: Int): SF[A, (A, Int)] =
  in => in.map(a => (a, level))

def timeout[A, B](seconds: Int): SF[(A, B), (A, B, Option[Int])] =
  in => in.map { case (a, b) => (a, b, Some(seconds)) }

def compose[A, B, C](p1: SF[A, B], p2: SF[B, C]): SF[A, C] =
```

```
      in => p2(p1(in))
    }

  def run[I, O](policy: Policy[I, O], in: ZStream[Any, Throwable, I]): ZStream[Any, Throwable, O] =
    policy.fold(StreamAlg)(in)
}
```

В.4. Модуль адаптації та вектор керування

Нижче наведено фрагмент модуля `src/main/scala/iot/adapt/Adaptation.scala`, який реалізує обчислення наступного вектора керування `Control` на основі поточних SLI та цільових SLO.

```scala
package iot.adapt

import iot.domain.EncodingFmt

final case class SLO(latencyP95Ms: Int, maxLossRate: Double, maxBytesPerMsg: Int)
final case class SLIs(rttMs: Double, loss: Double, queue: Int, bytesPerMsg: Int)
final case class Control(r: Double, wMs: Long, W: Int, k: Int, fmt: EncodingFmt)

object Adaptation {
```

```
 def ewma(prev: Double, sample: Double, alpha: Double): Double =
   alpha * sample + (1 - alpha) * prev

 // Обчислення наступного вектора керування з контрактним згладжуванням
 def next(u: Control, s: SLIs, slo: SLO, lambda: Double): Control = {
   val targetFmt =
     if (s.bytesPerMsg > slo.maxBytesPerMsg) EncodingFmt.Cbor else u.fmt
   val targetK   = if (s.loss > slo.maxLossRate || s.queue > 0) 1 else 0
   val targetR   =
     if (s.loss > slo.maxLossRate || s.rttMs > slo.latencyP95Ms)
       (u.r * 0.75) max 0.1
     else
       (u.r * 1.05) min 20.0
   val targetW   = if (s.rttMs > slo.latencyP95Ms) 1 else (u.W + 1) min 10
   val targetWms =
     if (s.queue > 0) (u.wMs * 1.5).toLong min 10000L
     else (u.wMs * 0.9).toLong max 50L

   u.copy(
     r = u.r + lambda * (targetR - u.r),
     wMs = (u.wMs + lambda * (targetWms - u.wMs)).toLong,
     W = Math.round(u.W + lambda * (targetW - u.W)).toInt,
     k = Math.round(u.k + lambda * (targetK - u.k)).toInt,
     fmt = targetFmt
   )
 }
}
```

```

B.5. Основні доменні типи телеметрії

```scala
package iot.domain

import java.time.Instant

final case class Telemetry(
    deviceId: String,
    ts: Instant,
    tempC: BigDecimal,
    humidity: BigDecimal,
    batteryV: BigDecimal,
    extra: Map[String, String] = Map.empty
)

sealed trait EncodingFmt
object EncodingFmt {
  case object Json     extends EncodingFmt
  case object Cbor     extends EncodingFmt
  case object Protobuf extends EncodingFmt
}

final case class Encoded(
    bytes: Array[Byte],
```
```

```
    fmt: EncodingFmt,
    contentType: String,
    approxSize: Int,
    telemetry: Telemetry
)

final case class Outgoing(
    topic: String,
    qos: Int,
    expirySeconds: Option[Int],
    alias: Option[Int],
    payload: Encoded
)
```

В.6. Основний застосунок шлюзу та адаптаційний цикл

У цьому підрозділі наведено фрагменти основного застосунку `src/main/scala/iot/app/Main.scala`, які демонструють побудову політики з вектора керування, конвеєр оброблення вхідних повідомлень та періодичний адаптаційний цикл.

```scala
object Main extends ZIOAppDefault {

  // Побудова політики з поточного вектора керування
```

```
        def policyFromControl(c: Control): Policy[Telemetry, (Encoded, Int, Option[Int])] = {
          val throttle: Policy[Telemetry, Telemetry] = Policy.Throttle[Telemetry](c.r)
        val delta: Policy[Telemetry, Telemetry] = Policy.Delta[Telemetry] { (prev, cur) => Option.when(prev != cur)(cur) }
          val batch: Policy[Telemetry, Telemetry] = Policy.Batch[Telemetry](maxN = 10, withinMs = c.wMs, agg = _.lastOption.get)
        val base: Policy[Telemetry, Encoded] = Policy.Encode(c.fmt)
        val qos: Policy[Encoded, (Encoded, Int)] = Policy.QoSLevel(c.k)
          val to: Policy[(Encoded, Int), (Encoded, Int, Option[Int])] = Policy.Timeout[Encoded, Int](seconds = 60)
        throttle >>> delta >>> batch >>> base >>> qos >>> to
       }

      // Порожня політика (fixed режим, без складної адаптації)
        private def emptyPolicy(c: Control): Policy[Telemetry, (Encoded, Int, Option[Int])] =
         Policy.Encode(c.fmt) >>> Policy.QoSLevel(0) >>> Policy.Timeout[Encoded, Int](seconds = 60)

      val program =
        for {
          cfg <- ZIO.service[AppConfig]
          mqtt <- ZIO.service[MqttClient]
          aliasCache <- ZIO.service[TopicAliasCache]
          repo <- ZIO.service[iot.db.Db.Repo]
```

```
            sliComp <- ZIO.service[SliComputation]
            controlRef <- ZIO.service[Ref[Control]]
            adaptationState <- ZIO.service[AdaptationEnabledState]

            // Потік оброблення MQTT-повідомлень
            processingStream =
              mqtt
                .subscribe(cfg.mqtt.subFilter)
                .mapZIO { inc =>
                  bytesReceivedTotal.incrementBy(inc.payload.length.toLong).as(CirceCod
ecs.fromJson(inc.payload))
                }
                .tap {
                  case Left(err) => ZIO.logError(s"decode error: $err")
                  case Right(_)  => messagesReceivedTotal.increment
                }
                .collectRight
                .viaFunction { stream =>
                  for {
                    pol  <- ZStream.fromZIO(controlRef.get.flatMap { c =>
                              ZIO.ifZIO(adaptationState.value.get)(
                                onTrue = ZIO.succeed(policyFromControl(c)),
                                onFalse = ZIO.succeed(emptyPolicy(c))
                              )
                            })
                    encs <- Interpreter.run(pol, stream)
                  } yield encs
```

```
            }
                // подальша обробка: публікація, оновлення метрик, запис у БД (скорочено)
            .mapZIO { case (enc, k, exp) =>
              /* публікація в брокер, оновлення SLI, збереження у сховище */
              ZIO.unit
            }
            .drain

          // Періодичний адаптаційний цикл
          adaptation =
            (for {
              c           <- controlRef.get
              s           <- sliComp.computeSLIs()
              adaptEnabled <- adaptationState.value.get
              nxt = if (adaptEnabled) Adaptation.next(c, s, cfg.slo, cfg.lambda) else c
              _   <- controlRef.set(nxt)
              _   <- policyRatePerSec.set(nxt.r) *>
                     policyBatchMs.set(nxt.wMs.toDouble) *>
                     policyInflightCap.set(nxt.W.toDouble) *>
                     policyQos.set(nxt.k.toDouble)
            } yield ()).repeat(Schedule.spaced(5.seconds))
        } yield ()
      }
      ```
```
```

ДОДАТОК Г

Результати перевірки роботи на співпадіння

StrikePlagiarism.com

Дата звіту 12/5/2025
Дата редагування 12/5/2025

Документ прийнятий

Звіт подібності

Метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-43мп_Ніконоров_ПЗ
Автор Науковий керівник / Експерт
НіконоровПоперешняк С.В.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

14633
Кількість слів

112562
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		39
Інтервали		0
Мікропробіли		1
Білі знаки		0
Парафрази (SmartMarks)		4

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз		Колір тексту
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://metod.vntu.edu.ua/getfile.php/8429.pdf	13 0.09 %
2	http://reposit.nupp.edu.ua/bitstream/PoltNTU/6386/1/%D0%9F%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA_%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80_%D0%84%D1%80%D0%BC%D1%96%D0%BB%D0%BE%D0%B2%D0%B0.pdf	7 0.05 %
3	http://ito.vspu.net/duplomni_rob/2019-2020r/4APO_kyrsovi/Gonta/Gonta.pdf	6 0.04 %
4	http://www.global-national.in.ua/archive/8-2015/250.pdf	6 0.04 %

5	http://reposit.nupp.edu.ua/bitstream/PoltNTU/6386/1/%D0%9F%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA_%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80_%D0%84%D1%80%D0%BC%D1%96%D0%BB%D0%BE%D0%B2%D0%B0.pdf	5 0.03 %
6	http://anchem.nuph.edu.ua/wp-content/uploads/2018/04/Gaidukevych-2018-Book.pdf	5 0.03 %
7	http://ito.vspu.net/duplomni_rob/2019-2020r/4APO_kyrsovi/Gonta/Gonta.pdf	5 0.03 %
8	http://ito.vspu.net/duplomni_rob/2019-2020r/4APO_kyrsovi/Gonta/Gonta.pdf	5 0.03 %

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (0.36 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	http://ito.vspu.net/duplomni_rob/2019-2020r/4APO_kyrsovi/Gonta/Gonta.pdf	16 (3) 0.11 %
2	**https://metod.vntu.edu.ua/getfile.php/8429.pdf**	**13 (1) 0.09 %**
3	http://reposit.nupp.edu.ua/bitstream/PoltNTU/6386/1/%D0%9F%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA_%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80_%D0%84%D1%80%D0%BC%D1%96%D0%BB%D0%BE%D0%B2%D0%B0.pdf	12 (2) 0.08 %
4	http://www.global-national.in.ua/archive/8-2015/250.pdf	6 (1) 0.04 %
5	http://anchem.nuph.edu.ua/wp-content/uploads/2018/04/Gaidukevych-2018-Book.pdf	5 (1) 0.03 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)