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

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Архітектура IoT на основі контролю передачі даних та туманних обчислень»

КвРКІП.700362.19.01.28 ПЗ

Виконав: студент 2 курсу, група КІ2м-23-1 Валентин МОСІЙЧУК
Підпис Ім'я, прізвище

Керівник: д.т.н., професор Сергій ЛИСЕНКО
Науковий ступінь, вчене звання Підпис Ім'я, прізвище

До захисту допускаю:
Зав. кафедри КІІС, доктор філософії, доцент
Ольга ПАВЛОВА
06 05 2025 р.

Хмельницький, 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-НАУКОВА ПРОГРАМА «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 01 " 09 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Валентину МОСІЙЧУКУ

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Архітектура IoT на основі контролю передачі даних та туманних обчислень

Керівник проекту (роботи) Сергій ЛИСЕНКО, д.т.н., професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 08.01.2025 №8

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

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

Аналіз технологій та архітектур туманних IoT-систем

Контроль передачі даних у IoT-ФОГ середовищах

Інтелектуальне управління обчисленнями та ресурсами

Проєктування, реалізація та перевірка IoT-ФОГ архітектури

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

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сергій ЛИСЕНКО, професор кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «_01_»_____09_____2024р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	01.09.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об`єкта та предмета дослідження	01.10.2024	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	01.11.2024	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.12.2024	виконано
5	Робота над науковою статтею	01.02.2025	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2025	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	01.04.2025	виконано
8	Оформлення пояснювальної записки згідно вимог	18.04.2025	виконано
9	Попередній захист ДРМ	29.04.2025	виконано
10	Захист ДРМ на засіданні ЕК	До 15.05.2025	

Студент _____ Підпис Валентин МОСІЙЧУК Ім'я, прізвище

Керівник роботи _____ Підпис Сергій ЛИСЕНКО Ім'я, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: «Архітектура IoT на основі контролю передачі даних та туманних обчислень»

Автор роботи: Валентин МОСІЙЧУК, студент групи КІ2м-23-1

Керівник роботи: Сергій ЛИСЕНКО, д.т.н., професор

Пояснювальна записка: 83 с., 18 рис., 6 табл. 1 матр., 3 дод., 90 джерел.

ІНТЕРНЕТ РЕЧЕЙ (IOT), ТУМАННІ ОБЧИСЛЕННЯ, ХМАРНІ ОБЧИСЛЕННЯ, АРХІТЕКТУРА IOT, МАРШРУТИЗАЦІЯ ПОВІДОМЛЕНЬ, FOG-ШЛЮЗ, CHIRPSTACK, LORAWAN.

Об’єктом дослідження є інфраструктура туманних обчислень у мережах Інтернету речей (IoT), зокрема способи обробки, маршрутизації та зберігання даних, які передаються від пристроїв-кінцевих вузлів через шлюзи до хмарних або локальних обчислювальних вузлів.

Предметом дослідження є методи оптимізації архітектури IoT-мережі з використанням туманних обчислень, контроль маршрутизації повідомлень, забезпечення конфіденційності, зменшення часу обробки повідомлень і підвищення надійності системи при втраті зв'язку з хмарою.

Метою кваліфікаційної роботи є зменшення часу оборобки повідомлень шляхом розроблення удосконаленої архітектури IoT-систем із використанням туманних обчислень.

Для розв’язання поставлених задач використовувалися методи агрегації, пріоритезації, адаптивної маршрутизації та компресії даних

Наукова новизна отриманих результатів:

1. Набула подальшого архітектура IoT-систем із використанням туманних обчислень, яка на відміну від відомих для оптимізація часу оборобки повідомлень застосовує туманні обчислення.

2. Набули подальшого розвитку програмно-технічні засоби удосконаленої архітектури IoT-систем із використанням туманних обчислень.На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення.

Практична значимість отриманих результатів полягає у розробленій архітектурі IoT-систем із використанням туманних обчислень.

У першому розділі описано сучасні підходи до побудови IoT-архітектур, обґрунтовано вибір fog computing як більш ефективної моделі для обробки даних у реальному часі.

У другому розділі проаналізовано методи оптимізації та схеми маршрутизації даних, що можуть бути використані в туманних IoT-мережах.

У третьому розділі запропоновано та описано детальну архітектуру системи на основі fog computing з підтримкою конфіденційності, маршрутизації, автономності.

У четвермоу розділі реалізовано програмний прототип системи з використанням реального обладнання та open-source платформ. А також проведено експериментальну перевірку – порівняння з TTN/ChirpStack, тести на затримку, обробку приватних даних і роботу в офлайн-режимі. Підтверджено переваги запропонованого рішення.

У висновках підведено підсумки досягнутих результатів та окреслено напрями подальших досліджень.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IoT – інтернет речей

ШІ – штучний інтелект

ГНМ – глибока нейронна мережа

ГНП – глибоке навчання з підкріпленням

ІТ – інформаційні технології

МН – машинне навчання

MEC – Multi-Access Edge Computing (граничні обчислення з множинним доступом)

НП – навчання з підкріпленням

ОС – операційна система

ЯО – якість обслуговування

СШІ – самоусвідомлений штучний інтелект

ВСТУП

Сучасний світ активно рухається у напрямку цифрової трансформації, що призводить до широкого впровадження Інтернету речей (IoT) у різних сферах людської діяльності. Смарт-пристрої та системи, які здатні збирати, обробляти та передавати дані, забезпечують підвищену ефективність у таких галузях, як промисловість, медицина, транспорт, розумні міста тощо. Разом з тим, стрімке зростання кількості IoT-пристроїв створює нові виклики, пов’язані з обробкою та передачею даних у реальному часі. Традиційні хмарні обчислення не завжди здатні забезпечити необхідний рівень продуктивності через високу затримку передачі даних та обмежену пропускну здатність мережі. У цьому контексті на перший план виходять туманні обчислення (Fog Computing), які дозволяють здійснювати обробку інформації ближче до її джерела, мінімізуючи затримки та знижуючи навантаження на мережу.

Туманні обчислення є проміжним рівнем між пристроями IoT та хмарними сервісами, забезпечуючи розподілену обробку даних на рівні локальних вузлів. Це дозволяє не тільки зменшити час реакції системи, але й підвищити рівень безпеки, оскільки частина даних може оброблятися безпосередньо на периферійних пристроях без необхідності передачі в хмару. Впровадження цієї технології стає особливо актуальним у критичних додатках, де навіть незначна затримка може мати серйозні наслідки, наприклад, у розумному транспорті, промислових системах або медичних моніторингових рішеннях.

У цьому дослідженні розглядається архітектура IoT-систем на основі туманних обчислень, а також механізми контролю передачі даних для оптимізації їх використання. Особлива увага приділяється питанням фільтрації, агрегації та пріоритизації трафіку, що дозволяє мінімізувати затримки та підвищити ефективність обробки даних. Також розглядаються аспекти безпеки та конфіденційності, що є ключовими при розробці сучасних IoT-рішень.

Мета роботи полягає у зменшенні часу оборобки повідомлень шляхом розроблення удосконаленої архітектури IoT-систем із використанням туманних обчислень.

Поставлена мета досягається розв’язанням таких основних задач:

– дослідити архітектуру IoT туманних обчислень на основі контролю передачі даних та туманних обчислень;

– проаналізувати сучасні архітектури IoT туманних обчислень;

– дослідити методи оптимізації передачі даних у розподілених системах;

– удосконалити архітектуру IoT туманних обчислень на основі контролю передачі даних та туманних обчислень;

– реалізувати засоби удосконаленої архітектури IoT-систем із використанням туманних обчислень.

Об’єктом дослідження є процес оптимізація функціонування IoT-систем.

Предметом дослідження є удосконаленої архітектури IoT-систем із використанням туманних обчислень.

Наукова новизна отриманих результатів:

1. Набула подальшого архітектура IoT-систем із використанням туманних обчислень, яка на відміну від відомих для оптимізація часу оборобки повідомлень застосовує туманні обчислення.

2. Набули подальшого розвитку програмно-технічні засоби удосконаленої архітектури IoT-систем із використанням туманних обчислень.

Практична цінність отриманих результатів. В результаті виконаного наукового дослідження було розроблено архітектури IoT-систем із використанням туманних обчислень.

Для розв’язання поставлених задач використовуються основні положення теорії комп’ютерних мереж та систем, системного аналізу, моделювання, методів аналізу даних, теорії математичної статистики, теорії дискретної математики, теорії.

За темою кваліфікаційної роботи магістра опубліковані тези у матеріалах XXVI Міжнародної студентської науково-технічної конференції «Перспективні мережі та комп’ютерні технології» ПерСиК 2025, 17 квітня 2025 р., Харків, Україна.

1 АНАЛІЗ ТЕХНОЛОГІЙ ТА АРХІТЕКТУР ТУМАННИХ IoT-СИСТЕМ

1.1 Основні концепції туманних обчислень та IoT

Сучасний цифровий світ зазнає значного впливу зі сторони інноваційних технологій, серед яких чільне місце займають Інтернет речей (Internet of Things, IoT) та туманні обчислення (Fog Computing) [1]. Ці два напрями взаємодоповнюють один одного та створюють фундамент для побудови високоефективних, адаптивних і масштабованих систем нового покоління, здатних забезпечити обробку великих обсягів даних в умовах реального часу [2].

Інтернет речей – це концепція, яка базується на ідеї з'єднання фізичних об’єктів у мережу з метою автоматизованого збору, обміну, аналізу та обробки даних [3]. IoT охоплює широкий спектр пристроїв – від простих датчиків температури до складних промислових машин, оснащених численними сенсорами [4]. Головна ідея полягає в тому, що ці пристрої можуть самостійно взаємодіяти між собою, з користувачами, а також із центральними інформаційними системами, що дозволяє створювати середовище, яке постійно оновлюється та реагує на зміни в режимі реального часу [5]. Однією з ключових особливостей IoT є децентралізований характер отримання даних, оскільки пристрої можуть бути розподілені по всій території, охоплюючи як локальні, так і глобальні мережі [6]. Завдяки цьому забезпечується глибока інтеграція цифрового простору з фізичним середовищем [7].

Втім, традиційна хмарна інфраструктура, яка широко застосовується для обробки IoT-даних, не завжди може забезпечити необхідну продуктивність у випадках, коли потрібна низька затримка, висока надійність або обмежена пропускна здатність мережі [8]. У відповідь на ці виклики з’являється концепція туманних обчислень [9]. Fog computing – це обчислювальна парадигма, яка передбачає розміщення ресурсів обробки, зберігання та управління ближче до джерел даних, тобто між крайовими пристроями (edge devices) та хмарою [10]. Інакше кажучи, fog computing виступає як проміжна обчислювальна ланка, яка

дозволяє частково або повністю перенести обчислювальні процеси з центру (хмари) на периферію мережі [11].

Така модель обробки має низку переваг. По-перше, вона значно знижує затримки у передачі даних, що критично важливо для таких застосувань, як автономне управління транспортом, промислове автоматизоване виробництво або системи моніторингу здоров’я [12]. По-друге, fog computing зменшує навантаження на центральні сервери та хмару, оскільки первинна обробка й фільтрація даних виконується локально, і до центру надсилаються лише необхідні, агреговані або вже оброблені дані [13]. По-третє, туманні обчислення підвищують рівень безпеки та конфіденційності, оскільки чутливі дані можуть не покидати локальне середовище обробки, що особливо актуально у фінансових та медичних системах [14].

Важливою характеристикою fog computing є підтримка розподілених обчислень, що дозволяє створювати масштабовані системи із динамічним розподілом навантаження [15]. Наприклад, у складних розумних містах, де працює безліч IoT-пристроїв – від камер спостереження до сенсорів якості повітря – саме туманні обчислення дозволяють об'єднати локальні вузли в єдину систему, яка реагує на події в режимі реального часу та здатна до автономного прийняття рішень [16]. Окрім того, fog-вузли можуть виконувати функції локального зберігання даних, а також реалізовувати алгоритми машинного навчання та штучного інтелекту для виявлення аномалій, прогнозування подій чи оптимізації управління [17].

Додатково до базових характеристик, важливою рисою Інтернету речей є його здатність генерувати величезні обсяги різнорідних даних – від простих показників сенсорів до складних потоків відео або телеметричних даних з рухомих об’єктів [18]. Ці дані часто є неструктурованими, мають високу швидкість надходження та потребують попередньої обробки вже на рівні пристроїв, що їх генерують [19]. У зв'язку з цим дедалі більше досліджень спрямовані на розробку систем, здатних до локального аналізу та автоматичної фільтрації інформації перед її передачею в хмарну інфраструктуру або інші мережеві вузли [20]. Саме fog

computing забезпечує реалізацію таких функцій, виступаючи не лише як засіб розвантаження мережі, а й як компонент попередньої аналітики даних [21].

Одним із важливих елементів fog-архітектури є її здатність підтримувати гетерогенність – тобто сумісність з різними типами пристроїв, протоколів і сервісів [22]. Завдяки цьому fog computing може ефективно інтегруватися в наявну інфраструктуру підприємств, міст або транспортних систем, не вимагаючи її повної перебудови [23]. Вона також дозволяє використовувати як потужні сервери на периферії, так і мікроконтролери або вбудовані обчислювальні пристрої, що істотно знижує вартість розгортання IoT-рішень [24].

Ще одним важливим аспектом є мобільність та гнучкість fog-середовища [25]. Оскільки багато IoT-пристроїв функціонують у динамічних умовах – наприклад, у транспорті, сільському господарстві або на будівельних майданчиках – туманні обчислення дозволяють створювати тимчасові або мобільні вузли обробки, які автоматично адаптуються до змін топології мережі [26]. Це відкриває нові перспективи для розгортання обчислювальних ресурсів у віддалених або обмежених умовах, де використання класичної хмари є технічно або економічно недоцільним [27].

З погляду безпеки та стійкості до відмов, fog computing демонструє вищу ефективність у порівнянні з централізованими системами [28]. Децентралізація обробки дозволяє уникнути повної зупинки системи у разі виходу з ладу одного з вузлів, адже інші елементи мережі можуть продовжити функціонування незалежно [29]. Крім того, за рахунок шифрування даних на місці збору та контролю доступу на локальному рівні забезпечується вищий ступінь захисту конфіденційної інформації [30].

У науковому та прикладному аспектах поєднання IoT з fog computing також створює потужний потенціал для застосування методів штучного інтелекту безпосередньо на краю мережі (edge AI) [31]. Це дозволяє розгортати інтелектуальні сервіси, які можуть самостійно приймати рішення на основі локального аналізу даних, виявляти закономірності або аномалії, прогнозувати

стан системи та навіть запускати автоматичні реакції без необхідності звернення до віддалених серверів [32].

Підсумовуючи, концепції Інтернету речей та туманних обчислень формують технічну основу для створення дійсно автономних, адаптивних і масштабованих кіберфізичних систем [33]. Вони поєднують в собі розподілену інфраструктуру, інтелектуальну обробку даних, мобільність, безпеку та здатність до самонавчання, що робить їх ключовими компонентами цифрової трансформації практично в усіх сферах людської діяльності – від міського середовища до індустрії 4.0 та цифрового здоров'я [34].

1.2 Сучасні технології бездротового зв'язку та архітектурні моделі

Сучасні технології бездротового зв'язку в IoT можна умовно поділити на дві великі категорії: короткодистанційні технології (як-от Wi-Fi, Bluetooth, Zigbee) та технології з широким покриттям і низькою потужністю (LPWAN), такі як LoRaWAN, Sigfox і NB-IoT. Кожна з них орієнтована на різні сценарії використання [35].

Wi-Fi (Wireless Fidelity) – це технологія бездротового зв'язку, яка базується на стандартах IEEE 802.11 [36]. Вона є однією з найбільш розповсюджених мережевих технологій у світі, використовується для організації локальних бездротових мереж (WLAN) і забезпечує високошвидкісний доступ до Інтернету або локальних ресурсів. У контексті IoT Wi-Fi відіграє важливу роль, зокрема в додатках, де потрібна висока пропускна здатність, швидка передача даних і наявна постійна електроживлення [37].

Wi-Fi найчастіше використовується в середовищах із обмеженим простором – таких як розумні будинки, офіси, лабораторії та освітні заклади, де пристрої перебувають у межах покриття однієї або кількох точок доступу (Access Point, AP) [38]. У цих умовах перевагами Wi-Fi є простота інтеграції з існуючими мережами, висока швидкість передачі даних (до декількох гігабітів на секунду в новіших

версіях, таких як Wi-Fi 6/6E) та наявність стандартних протоколів безпеки (WPA3 тощо) [39].

Однак незважаючи на свою популярність, Wi-Fi має низку обмежень, які ускладнюють його використання в багатьох сценаріях IoT [40]. Найбільш вагомими недоліками є високе енергоспоживання і обмежена дальність дії. Традиційні IoT-пристрої часто функціонують на батарейному живленні, тому для них критично важлива економія енергії. Звичайна Wi-Fi передача вимагає набагато більше енергії, ніж, скажімо, BLE або LoRaWAN, що знижує термін автономної роботи пристрою з декількох років до декількох днів або тижнів [41]. Це робить Wi-Fi неідеальним вибором для пристроїв, розташованих у важкодоступних місцях або тих, що повинні працювати довготривало без втручання людини.

Щодо радіуса дії, Wi-Fi має приблизно 30–50 метрів в середині приміщень (через перешкоди та загасання сигналу) і до 100 метрів на відкритому просторі [42]. Цього достатньо для більшості кімнатних або офісних середовищ, але недостатньо для покриття великих площ, таких як промислові зони, сільське господарство або міська інфраструктура.

Незважаючи на ці обмеження, Wi-Fi залишається цінним інструментом у вузькоспеціалізованих IoT-сценаріях, особливо там, де:

- потрібна велика пропускна здатність для передавання зображень або відео (наприклад, IP-камери спостереження, пристрої відеоаналітики) [43];
- є джерело постійного живлення (наприклад, побутова техніка, розумні телевізори, роутери);
- існує необхідність у швидкій та прямій інтеграції з хмарними сервісами;
- застосовується локальна обробка даних з мінімальними затримками [44].

Також Wi-Fi підтримує захищені з’єднання, що особливо важливо для забезпечення конфіденційності та цілісності даних. Новітні стандарти, як-от Wi-Fi 6 (802.11ax) і Wi-Fi 6E, додають покращення в енергоефективності, управлінні навантаженням у мережі (особливо у випадку великої кількості підключених

пристроїв) і зменшенні затримок [45]. Це відкриває нові можливості для IoT-додатків, які раніше не могли ефективно працювати з Wi-Fi.

У перспективі Wi-Fi залишиться важливою частиною IoT-екосистеми, особливо у поєднанні з іншими технологіями. Наприклад, у гібридних рішеннях пристрої можуть використовувати Wi-Fi для комунікації на коротких відстанях і водночас мати резервний канал на основі LPWAN для передачі даних у разі втрати локальної мережі [46].

Bluetooth Low Energy (BLE) є спеціалізованою версією класичної технології Bluetooth, розробленою для бездротового з’єднання пристроїв на короткі відстані з низьким енергоспоживанням [47]. Вона оптимізована для підключення пристроїв, що працюють від батарей, до мережі, забезпечуючи мінімальні вимоги до енергоспоживання. BLE ідеально підходить для IoT, оскільки дозволяє з'єднувати пристрої, такі як фітнес-трекери, медичні пристрої, датчики, розумні годинники та інші пристрої, що потребують бездротової комунікації з низьким енергоспоживанням [48].

Однією з ключових характеристик BLE є низьке енергоспоживання. Завдяки спеціальному режиму сну, пристрої можуть перебувати в сплячому стані, прокидаючись лише для коротких періодів для передачі даних. Це дозволяє значно подовжити термін служби батарей, що особливо важливо для пристроїв, які працюють автономно без частих підзарядок або заміни батарей [49].

BLE оптимізовано для роботи на коротких відстанях, зазвичай до 100 метрів на відкритому просторі, але цей радіус може бути меншим через перешкоди в середовищі. Це робить BLE ідеальним для пристроїв, які працюють в обмежених просторах, таких як будинки, офіси або локальні сенсорні мережі [50]. Хоча BLE забезпечує низьку пропускну здатність порівняно з іншими технологіями, такими як Wi-Fi або LTE, він здатен передавати дані, необхідні для IoT-програм, де вимоги до швидкості передачі не є критичними [51].

BLE також підтримує підключення до кількох пристроїв одночасно, що дає можливість ефективно організовувати мережі, де центральний пристрій, наприклад, смартфон або шлюз, може з'єднуватися з десятками або сотнями

сенсорів та інших IoT-пристроїв [52]. Це дозволяє створювати широкі системи автоматизації та моніторингу без потреби в складних інфраструктурах.

Технологія активно використовується в різних сферах, таких як медичні пристрої, де передача даних в реальному часі є важливою. Прилади, як-от монітори серцевого ритму, глюкометри, фітнес-трекери та інші, використовують BLE для бездротової передачі даних. Ці пристрої можуть передавати зібрані показники на мобільні додатки або до медичних центрів для подальшої обробки [53].

BLE також широко застосовується в розумних будинках та офісах, де забезпечується управління освітленням, дверними замками, системами безпеки, а також відстеженням місцезнаходження людей і предметів. Важливою перевагою BLE є те, що він не потребує великої потужності для реалізації таких рішень, що робить його привабливим для використання в системах, що потребують тривалого функціонування на батарейках [54].

Хоча Bluetooth Low Energy має багато переваг, він має і свої обмеження. Пропускна здатність BLE є обмеженою, що робить його менш придатним для застосувань, де необхідна висока швидкість передачі великих обсягів даних. Крім того, дальність зв’язку обмежена, і вона не підходить для довгих комунікацій на великі відстані. Проте зважаючи на те, що BLE спеціально розроблений для енергозбереження та роботи з малими пристроями, його обмежена пропускна здатність є прийнятною для більшості IoT-застосувань [55].

LPWAN (Low Power Wide Area Network) – це категорія бездротових технологій, призначених для забезпечення низькошвидкісної передачі даних на великі відстані з низьким енергоспоживанням [56]. LPWAN є ідеальним вибором для IoT-систем, де пристрої повинні працювати автономно на батарейках протягом тривалого часу і передавати лише невеликі обсяги даних. Ці мережі зазвичай використовуються для з'єднання сенсорів, моніторингу, контролю та інших IoT-пристроїв в умовах, коли інші технології, такі як Wi-Fi або мобільні мережі, не підходять через високе енергоспоживання чи обмежену дальність зв'язку [57].

Завдяки великому радіусу дії, LPWAN дозволяє забезпечувати з’єднання між пристроями, розміщеними на великій відстані, навіть у віддалених місцях, де немає

доступу до традиційних мереж зв’язку [58]. Важливою характеристикою є також здатність працювати в умовах міського середовища з високим рівнем перешкод, таких як стіни будівель чи інші структури, які можуть ускладнити зв'язок для інших технологій [59]. Ці характеристики роблять LPWAN дуже популярним для таких застосувань, як моніторинг навколишнього середовища, сільське господарство, системи відстеження, логістика та енергетика [60].

LoRaWAN (Long Range Wide Area Network) є однією з найпоширеніших технологій в категорії LPWAN і спеціалізується на передаванні даних на великі відстані з низьким енергоспоживанням [61]. LoRaWAN використовує радіочастотний спектр без ліцензії, що дозволяє організовувати мережі без необхідності ліцензування радіочастот, що знижує витрати на інфраструктуру [62]. LoRaWAN має можливість передавати дані на відстань до кількох десятків кілометрів в умовах відкритого простору, а також до кількох кілометрів в міських умовах [63]. Це робить технологію дуже зручною для IoT-програм, де необхідно забезпечити покриття великої території або зв'язок з віддаленими пристроями.

LoRaWAN підтримує великі мережі пристроїв з обмеженими вимогами до пропускної здатності, тому зазвичай використовується для передачі невеликих обсягів даних, таких як показники датчиків, що вимірюють температуру, вологість, рівень забруднення тощо [64]. Технологія LoRaWAN оптимізована для роботи на батарейках, що дозволяє пристроям працювати автономно протягом тривалого часу – від кількох місяців до кількох років, в залежності від налаштувань і інтервалів між передачею даних [65]. Технологія також підтримує мережі з низьким енергоспоживанням, що робить її дуже привабливою для застосувань в розумних містах, сільському господарстві, логістиці та інших сферах IoT [66].

NB-IoT (Narrowband IoT) є ще однією важливою технологією в категорії LPWAN, що пропонує рішення для бездротової передачі даних на великі відстані через мобільні мережі [67]. NB-IoT використовує частотний спектр в діапазоні вузької смуги для забезпечення ефективної передачі даних із низьким енергоспоживанням, що робить його ідеальним для IoT-пристроїв, які повинні працювати в умовах обмежених ресурсів, таких як датчики і монітори [68]. Це

технологія, розроблена з урахуванням потреб великої кількості пристроїв в мережах мобільного зв’язку.

Однією з головних переваг NB-IoT є те, що вона працює на вже існуючій мобільній інфраструктурі, що дозволяє швидко і економічно забезпечити підключення для великої кількості IoT-пристроїв без необхідності створювати окрему мережу [69]. Технологія підтримує великий діапазон зв’язку (від кількох кілометрів в міських умовах до десятків кілометрів в умовах відкритого простору) і здатна передавати дані навіть у важкодоступних місцях, таких як підземні приміщення або глибокі будівлі [70]. Як і LoRaWAN, NB-IoT також оптимізований для низького енергоспоживання, що дозволяє IoT-пристроям працювати на батарейках протягом тривалого часу [71].

Однак, на відміну від LoRaWAN, NB-IoT використовує вже існуючу мобільну інфраструктуру, тому його застосування часто залежить від наявності покриття в конкретному регіоні і від того, які оператори мобільного зв'язку підтримують цю технологію [72]. NB-IoT здатен передавати більш високі обсяги даних порівняно з іншими LPWAN технологіями, що дозволяє його використовувати не тільки для передачі невеликих обсягів даних, але і для деяких більш вимогливих застосувань, таких як моніторинг і управління в реальному часі [73].

Загалом, LPWAN технології, включаючи LoRaWAN і NB-IoT, дозволяють створювати потужні мережі IoT, що забезпечують передачу даних на великі відстані з низьким енергоспоживанням. Вибір між цими технологіями залежить від конкретних вимог до покриття, швидкості передачі даних, енергоспоживання та вартості, що робить їх дуже різноманітними для використання в численних IoT додатках [74].

Архітектура туманних обчислень у середовищі Інтернету речей (IoT) побудована на багаторівневому принципі, що забезпечує ефективну організацію обробки, передачі й зберігання даних [75]. Її структура є гнучкою і масштабованою, дозволяючи реалізовувати як централізовані, так і децентралізовані схеми обчислень, орієнтовані на конкретні сценарії застосування. В основі архітектури

лежить поділ інфраструктури на три ключові рівні: крайовий (edge), туманний (fog) і хмарний (cloud).

Крайовий рівень (Edge Level) є фундаментальним шаром у багаторівневій архітектурі туманних обчислень. Він розташовується безпосередньо поруч із фізичним середовищем, у якому розгортається система Інтернету речей, і забезпечує первинну взаємодію з об'єктами навколишнього світу [76]. Це найближчий до джерела даних рівень, де відбувається захоплення, обробка й передача інформації, що надходить від сенсорів, камер, RFID-міток, мікроконтролерів, портативних пристроїв, промислових машин тощо.

Основною характеристикою крайового рівня є його низька затримка – завдяки розміщенню максимально близько до джерел даних він здатен забезпечити майже миттєву реакцію системи на зовнішні події. Це особливо важливо в критичних сценаріях, де затримки в обробці можуть призвести до серйозних наслідків: автономне керування транспортом, медичний моніторинг пацієнтів, контроль промислового обладнання, системи безпеки тощо [77].

Крайовий рівень представлений великою кількістю пристроїв з різною обчислювальною потужністю. У найпростішому випадку це можуть бути сенсорні вузли, які лише збирають дані та передають їх вище – на туманний чи хмарний рівень. Проте із розвитком edge computing дедалі більше пристроїв здатні виконувати базову попередню обробку інформації локально: фільтрацію, перетворення, аналіз на основі простих евристик або моделей машинного навчання [78].

Це дозволяє зменшити навантаження на мережу, уникати надмірної передачі «сирих» даних і підвищити енергоефективність усієї системи [79]. Наприклад, камера відеоспостереження може виявляти рух і передавати вищим рівням тільки фрагменти, де є активність, замість того щоб транслювати відео потік цілодобово.

Окрім обробки, крайові пристрої часто виконують функції локального контролю та управління – наприклад, запуск актуаторів у відповідь на певні тригери, регулювання температури, увімкнення сигналізації або зупинку

механізмів при виявленні небезпеки. У такий спосіб крайовий рівень може діяти автономно, не очікуючи команди від хмари чи fog-ноди.

У технічному аспекті пристрої edge-рівня мають низьку споживчу потужність, можуть працювати від батарей або відновлюваних джерел енергії, бути обладнані бездротовими модулями зв'язку (Wi-Fi, ZigBee, LoRa, Bluetooth, NB-IoT тощо) і мати різні ступені захисту від зовнішнього впливу. Часто вони є частиною розподіленої мережі, де кожен пристрій працює незалежно або в координації з іншими вузлами.

Особливу роль відіграє безпека на крайовому рівні, оскільки саме тут система є найвразливішою до атак: перехоплення даних, підміна сенсорної інформації, фізичний доступ до обладнання. Для цього застосовуються методи шифрування на рівні пристроїв, автентифікація, технології блокчейну для фіксації транзакцій, а також впровадження механізмів «zero trust» та оновлення мікропрограм через захищені канали.

Загалом, крайовий рівень є ключовим елементом у розумному розподілі ресурсів, оскільки він визначає, які саме дані та в якому обсязі повинні передаватись далі [80]. Його ефективна організація суттєво підвищує загальну продуктивність fog-IoT-архітектури, знижує затримки, забезпечує стабільну роботу в умовах обмеженої доступності мережі та підтримує масштабування системи за рахунок додавання нових пристроїв.

Крайові обчислення стають дедалі потужнішими завдяки появі нових платформ, таких як NVIDIA Jetson, Google Coral, Raspberry Pi 5, Arduino з AI-модулями, що дозволяє реалізовувати навіть складні алгоритми розпізнавання образів, класифікації або локального прогнозування без необхідності надсилати дані в хмару. Це прокладає шлях до створення по-справжньому автономних і саморегульованих IoT-систем.

Туманний рівень у багаторівневій архітектурі IoT-технологій виконує функцію проміжного шару між крайовими пристроями та хмарною інфраструктурою. Він об'єднує функціональні можливості обох рівнів, забезпечуючи локальну обробку, зберігання та фільтрацію даних, а також

частковий аналіз в реальному часі, не покладаючись повністю на центральну хмару. Термін "туманні обчислення" (fog computing), вперше запропонований компанією Cisco, описує обчислювальну парадигму, у якій ресурси розташовані ближче до джерел даних, що дозволяє зменшити затримки та знизити обсяг переданого трафіку.

На цьому рівні працюють пристрої, що мають значно вищу обчислювальну потужність, ніж крайові вузли, але не досягають рівня централізованих хмарних дата-центрів [81]. Це можуть бути шлюзи, маршрутизатори, локальні сервери, комутатори з вбудованою аналітикою, промислові контролери, міні-ЦОДи (micro data centers) та інші мережеві пристрої з підтримкою обробки та зберігання даних.

Основною особливістю туманного рівня є його здатність виконувати попередню обробку даних та аналітику на місці, тобто ближче до джерел інформації. Наприклад, замість надсилання всіх сирих даних від тисяч датчиків на хмару, fog-вузол може агрегувати, фільтрувати, стискати або аналізувати інформацію, надсилаючи до хмари лише потрібні результати або сигнали про події.

Цей рівень часто використовується для обслуговування часочутливих завдань, які не можуть залежати від затримок, пов’язаних із хмарною обробкою [82]. До таких завдань належать, наприклад, автоматичне гальмування автомобіля при виявленні перешкоди, виявлення аномалій у виробничому процесі, або запуск аварійної сигналізації в умовах підвищеної температури чи витоку газу. У таких випадках туманний рівень забезпечує мінімальний час реакції та оперативність управління.

Ще однією важливою функцією туманного рівня є управління мережею, включаючи маршрутизацію, балансування навантаження, управління потоками даних та безпекою. На цьому рівні можуть розміщуватися служби автентифікації, шифрування, контроль доступу та інші механізми захисту інформації, що покращує загальну безпеку IoT-інфраструктури.

У порівнянні з крайовим рівнем, туманний рівень має більшу обчислювальну потужність і ресурси, а у порівнянні з хмарою – меншу затримку та ближчий фізичний доступ до пристроїв користувача. Він відіграє критичну роль у

забезпеченні гнучкості й масштабованості, дозволяючи IoT-системам адаптуватися до різних вимог в реальному часі [83].

У загальній архітектурі IoT систем туманний рівень:

– виступає локальним обчислювальним вузлом, що дозволяє зменшити навантаження на хмару;

– слугує буфером між великою кількістю розподілених пристроїв і централізованими дата-центрами;

– забезпечує відмовостійкість, оскільки частина завдань може виконуватись навіть у випадку втрати з’єднання з хмарною інфраструктурою;

– виконує гнучке розподілення обчислювальних навантажень, що особливо важливо в умовах високої щільності IoT-пристроїв.

Таким чином, туманний рівень є ключовою ланкою в ієрархії IoT-інфраструктури, що дозволяє поєднати обчислювальну потужність, мобільність, адаптивність та безпеку в одному середовищі, формуючи ефективну, розподілену систему обробки даних, орієнтовану на потреби сучасних кіберфізичних систем і Smart-рішень.

Хмарний рівень (Cloud Level) є верхнім рівнем у багатошаровій архітектурі IoT-туманних обчислень. Він виконує функцію централізованого оброблювального ядра та довготривалого зберігання даних. На відміну від крайового та туманного рівнів, які зосереджуються на обробці даних ближче до джерел, хмарний рівень надає практично необмежені обчислювальні ресурси, можливості для глибокої аналітики, навчання моделей штучного інтелекту, управління великими масивами даних (Big Data) та централізоване адміністрування систем [84].

У загальному випадку хмарна інфраструктура реалізована на основі високопродуктивних дата-центрів, які можуть розміщуватись у будь-якій географічній точці та об'єднані глобальною мережею. Через інтернет вони забезпечують взаємодію з IoT-пристроями, fog-нодами, сервісами користувача та зовнішніми API.

Однією з ключових функцій хмарного рівня є зберігання великих обсягів даних, які надходять із сенсорних вузлів та обробляються на попередніх рівнях. У

хмарі накопичуються історичні дані, які потім використовуються для аналітики, прогнозування, побудови статистичних моделей та машинного навчання. Це забезпечує стратегічну гнучкість – завдяки аналізу даних за довгий період можна виявляти тренди, закономірності, аномалії та оптимізувати роботу IoT-систем.

Іншою важливою особливістю хмарного рівня є можливість централізованого керування системою – зокрема оновленням програмного забезпечення, налаштуванням політик безпеки, управлінням користувачами, ліцензуванням сервісів, моніторингом стану пристроїв та їхньою телеметрією. Це дозволяє зменшити навантаження на локальні ресурси та полегшує підтримку великомасштабних розгортань.

Завдяки своїм можливостям хмарний рівень широко використовується для інтелектуального аналізу даних. Тут реалізуються сервіси штучного інтелекту, глибокого навчання, візуалізації даних, створення дашбордів і звітів. Хмара надає інструменти для побудови, навчання та розгортання моделей – наприклад, прогнозування споживання енергії, поведінки користувачів або можливих збоїв у роботі пристроїв [85].

Хмарний рівень часто реалізується у вигляді хмарних платформ IoT (IoT cloud platforms), таких як Amazon AWS IoT, Microsoft Azure IoT Hub, Google Cloud IoT Core, IBM Watson IoT тощо. Вони забезпечують повноцінну інфраструктуру як сервіс (IaaS), платформу як сервіс (PaaS) або програмне забезпечення як сервіс (SaaS), що дозволяє розробникам легко інтегрувати IoT-пристрої та швидко запускати нові сервіси.

Проте використання хмари також має певні обмеження. Через географічну віддаленість хмарних дата-центрів час відгуку може бути занадто великим для критичних задач реального часу, тому вона не завжди підходить для оперативного реагування. У таких випадках задачі делегуються на fog- або edge-рівні, а хмара зберігає стратегічну функцію довготривалої обробки.

Також важливими є питання конфіденційності та безпеки даних, які обробляються у хмарі. Оскільки хмарна інфраструктура зазвичай управляється сторонніми провайдерами, необхідно дотримуватись міжнародних стандартів

безпеки, таких як ISO/IEC 27001, GDPR, HIPAA тощо. Тут застосовуються методи шифрування даних, багатофакторна автентифікація, політики доступу, системи аудиту та моніторингу інцидентів.

Таким чином, хмарний рівень в архітектурі fog-IoT виступає як аналітичний, координуючий та зберігальний центр, що забезпечує масштабованість, обчислювальну гнучкість і централізовану обробку. Його потужності доповнюють ресурси fog- і edge-рівнів, створюючи ефективну гібридну архітектуру, де кожен рівень виконує своє специфічне завдання, забезпечуючи високу продуктивність, надійність і безперервність функціонування IoT-систем.

1.3 Огляд існуючих досліджень та постановка задачі

Під час аналізу існуючих рішень потрібно враховувати різноманітні підходи до обробки даних і передачі їх в межах мережі IoT. Враховуючи масштаби IoT-систем, зростаючі вимоги до пропускної здатності та зменшення затримок, традиційні рішення, засновані на централізованих хмарних обчисленнях, часто виявляються неефективними. Це зумовлено значними обмеженнями в області затримок і пропускної здатності, що особливо актуально для таких галузей, як автономне водіння, охорона здоров’я, розумні міста та інші критично важливі застосунки [86].

Важливим кроком для вирішення цих проблем є концепція туманних обчислень (Fog Computing), яка дозволяє здійснювати обробку даних на більш близькому рівні до користувача або пристрою, що значно знижує затримки та підвищує ефективність передачі даних. Замість того, щоб передавати всі дані в хмару для обробки, частину з них можна обробляти на локальних або розподілених серверах, розташованих ближче до кінцевих користувачів. Цей підхід дозволяє істотно зменшити навантаження на основні канали зв’язку і хмарні платформи, забезпечуючи необхідний рівень часу відгуку [87].

Існуючі рішення, орієнтовані на IoT, враховують різні аспекти системи, починаючи від мережевих технологій до методів обробки даних на різних етапах їх

передачі. Одним з таких підходів є використання крайових обчислень (Edge Computing), яке дозволяє здійснювати обробку даних безпосередньо на пристроях або в їх безпосередньому оточенні. Це дозволяє знизити затримки при обробці даних, однак такі системи мають обмеження в плані масштабованості та ресурсів, що доступні для обробки великих обсягів даних [88].

Туманні обчислення ж дозволяють вирішувати ці обмеження завдяки наявності додаткових проміжних рівнів між крайовим рівнем і хмарою, що дає змогу здійснювати ефективну обробку даних на більш гнучких і масштабованих розподілених системах. Рішення, які використовують такі технології, повинні оптимізувати баланс між швидкістю обробки даних і їхньою передачею для забезпечення високої ефективності роботи IoT-систем.

Постановка задачі в даному контексті включає кілька ключових аспектів, що необхідно вирішити для успішної реалізації туманних обчислень в IoT. Однією з таких задач є оптимізація маршрутизації даних між кінцевими пристроями, проміжними туманними вузлами та хмарними сервісами. Для цього потрібні алгоритми, які будуть враховувати специфіку мережевої інфраструктури, обмеження по пропускній здатності, затримках та енергоспоживанню, щоб мінімізувати витрати на передачу даних і забезпечити високий рівень продуктивності.

Ще однією важливою задачею є розподіл обчислювальних завдань між різними рівнями обробки, від пристроїв на крайовому рівні до туманних і хмарних сервісів. Для цього слід розробити стратегії, які дозволяють розумно вирішувати, які завдання повинні бути оброблені на місці (на рівні крайових пристроїв), а які можна передати на туманні вузли чи в хмару, де доступні потужніші ресурси для складнішої обробки.

Іншим важливим аспектом є управління безпекою та конфіденційністю даних в процесі їх передачі між різними рівнями. Для забезпечення цілісності та конфіденційності даних необхідно розробити надійні методи шифрування, а також механізми для захисту від несанкціонованого доступу. Враховуючи чутливість

даних в IoT-системах, як, наприклад, в медицині або транспорті, це питання є критично важливим [89].

Енергетична ефективність також є однією з ключових задач при розробці рішень для IoT. Більшість пристроїв в IoT працюють на батарейках, що робить енергоефективність важливим аспектом, який потребує особливої уваги. Тому розробка методів для зменшення енергоспоживання на кожному етапі обробки даних – від крайового до туманного та хмарного рівнів – є важливою складовою в рішенні задачі [90].

Таким чином, завдання постановки включає вирішення ряду проблем, пов’язаних з ефективністю обробки даних, їхньою передачею, безпекою, енергоспоживанням та адаптивністю системи до змінних умов. Вирішення цих проблем дозволить значно підвищити ефективність і надійність IoT-мереж, а також створить умови для розвитку нових, більш потужних і гнучких рішень у сфері туманних обчислень.

1.4 Висновки до першого розділу

Отже, було розглянуто основні концепції туманних обчислень та Інтернету речей (IoT), що дозволяє зрозуміти ключові принципи роботи цих технологій та їх взаємодію. IoT є основою для створення розумних пристроїв, які здатні взаємодіяти та обмінюватися даними через мережу, а туманні обчислення, в свою чергу, забезпечують необхідну обробку даних в реальному часі на проміжних вузлах мережі, що розташовані ближче до кінцевих пристроїв. Це дозволяє значно знизити затримки, оптимізувати передачу даних та зменшити навантаження на центральні сервери. Важливо, що туманні обчислення можуть бути застосовані в численних сферах, таких як розумні міста, транспортні системи, охорона здоров’я та інші, де важлива висока швидкість та ефективність обробки даних.

Також було детально розглянуто сучасні технології бездротового зв’язку для IoT, серед яких Wi-Fi, Bluetooth Low Energy (BLE), LPWAN, LoRaWAN та NB-IoT. Кожна з цих технологій має свої переваги та недоліки, що визначають їх

застосування в залежності від конкретних вимог до пропускної здатності, радіусу дії, енергоспоживання та інших факторів. Зокрема, Wi-Fi використовується для підключення пристроїв з високими вимогами до пропускної здатності, BLE є оптимальним для пристроїв з низьким енергоспоживанням, а LPWAN, LoRaWAN та NB-IoT забезпечують підключення на великих відстанях при мінімальному енергоспоживанні.

Окрім того, було висвітлено важливість архітектурних моделей, які підтримують інфраструктуру IoT та туманних обчислень. Зокрема, було розглянуто три основні рівні архітектури: крайовий, туманний і хмарний, кожен з яких має свою роль у забезпеченні обробки даних та керуванні ресурсами. Крайовий рівень відповідає за збирання даних з пристроїв і первинну обробку, туманний рівень здійснює більш складну обробку та маршрутизацію, а хмарний рівень забезпечує глобальне зберігання та аналіз даних. Комбінація цих рівнів дозволяє створювати ефективні і масштабовані рішення для IoT-систем.

В огляді існуючих досліджень було визначено, що на даний момент існують численні виклики в області обробки та передачі даних у рамках IoT, включаючи питання зниження затримок, оптимізації використання енергоресурсів та забезпечення безпеки даних. Однак розвиток нових алгоритмів, підходів до маршрутизації та управління ресурсами, а також вдосконалення технологій бездротового зв’язку дозволяють створювати більш ефективні та адаптивні рішення для таких систем.

Таким чином, основні задачі, які постають перед сучасними IoT-системами, включають оптимізацію балансування навантаження між різними рівнями обробки даних, розробку ефективних алгоритмів маршрутизації та забезпечення енергетичної ефективності. Дослідження в цій галузі є важливими для подальшого розвитку та вдосконалення технологій туманних обчислень і IoT, що в свою чергу дозволить забезпечити високопродуктивні, масштабовані та енергоефективні рішення для різних галузей.

2 КОНТРОЛЬ ПЕРЕДАЧІ ДАНИХ У IoT-ФОГ СЕРЕДОВИЩАХ

2.1 Методи оптимізації передачі даних у розподілених системах

Оптимізація передачі даних у розподілених системах, зокрема в контексті Інтернету речей (IoT) та туманних обчислень, є критичним аспектом для забезпечення ефективної, безперебійної та енергоефективної роботи мережі. Розподілені обчислювальні системи характеризуються наявністю великої кількості вузлів, розміщених у різних фізичних точках мережі, які взаємодіють між собою через мережу зв’язку. У таких системах існує висока залежність між якістю каналів передачі, швидкістю обробки інформації на кожному рівні (крайовому, туманному та хмарному) та ефективністю роботи всієї архітектури.

Адаптивна маршрутизація є ключовим елементом забезпечення ефективної, гнучкої та надійної передачі даних у сучасних розподілених системах, таких як Інтернет речей (IoT) та архітектури туманних обчислень. У таких системах інформація генерується тисячами або навіть мільйонами пристроїв, розташованих у різних географічних точках, і має бути швидко та безпечно доставлена до відповідних вузлів для обробки. З огляду на динамічний характер таких середовищ, фіксовані маршрути втрачають ефективність, тому виникає необхідність у гнучких, адаптивних рішеннях.

Адаптивна маршрутизація передбачає зміну маршрутів передавання даних у режимі реального часу на основі поточного стану мережі. Основна ідея полягає в тому, що мережа сама оцінює свої умови – затримки, пропускну здатність, рівень навантаження, кількість втрат пакетів, енергетичний ресурс вузлів тощо – і приймає рішення щодо найкращого шляху для кожного повідомлення або потоку даних. Такий підхід забезпечує підвищення ефективності роботи всієї системи, зниження затримок, збільшення надійності та покращення використання ресурсів – рисунок 2.1.

У контексті IoT-систем адаптивна маршрутизація часто реалізується у вигляді протоколів з урахуванням якості обслуговування (QoS). Ці протоколи враховують критичність даних, необхідну швидкість доставки, допустимі затримки

та інші параметри, що дозволяє пріоритезувати трафік. Наприклад, трафік з відеокамер системи безпеки буде мати вищий пріоритет, ніж показники температури з аграрних сенсорів. Завдяки цьому критично важливі дані не втрачаються і доставляються вчасно.

Однією з найважливіших характеристик адаптивної маршрутизації є її здатність до реакції на збої в мережі. Якщо певний вузол або маршрут вийшов з ладу, система здатна швидко знайти альтернативний шлях, уникаючи втрати даних. Це особливо важливо для систем з високим ступенем динамічності, де вузли можуть підключатися або відключатися в будь-який момент, наприклад, у розумних містах, мобільних сенсорних мережах або в системах транспорту.

У системах туманних обчислень адаптивна маршрутизація також тісно пов’язана з балансуванням навантаження. Вибір маршруту враховує поточне навантаження на обчислювальні вузли, зокрема на рівнях fog та edge. Наприклад, якщо один fog-вузол перевантажений, трафік може бути перенаправлений до іншого, менш навантаженого, навіть якщо той розташований трохи далі. Це забезпечує рівномірний розподіл ресурсів і стабільну роботу всієї архітектури.

Ще одним напрямком є енергоефективна адаптивна маршрутизація, що має критичне значення для пристроїв IoT, які працюють на батареях. У цьому випадку маршрутизаційні рішення приймаються з урахуванням залишкового енергетичного ресурсу вузлів. Система намагається мінімізувати кількість хопів (переходів), використовує вузли з більшим енергетичним запасом і уникає перевантаження окремих елементів. Це подовжує загальний термін експлуатації мережі.

Адаптивна маршрутизація також враховує географічні характеристики та контекст середовища. Наприклад, у розумних сільськогосподарських системах маршрути можуть будуватися з урахуванням змін погодних умов, щільності рослинності або наявності перешкод для радіосигналу. У міських умовах може використовуватись інформація про рух транспорту або пішоходів.

Інтелектуальні механізми адаптації включають застосування машинного навчання та штучного інтелекту. Алгоритми класифікації, прогнозування або підкріплення використовуються для виявлення закономірностей у поведінці

мережі та визначення найефективніших маршрутів залежно від історичних і поточних даних. Наприклад, система може навчитися передбачати години пікового навантаження та завчасно змінювати маршрути, запобігаючи перевантаженню мережі.

Нарешті, адаптивна маршрутизація підтримує гнучке масштабування системи. Завдяки динамічному обрахунку маршрутів можна безпечно додавати нові пристрої або вузли без потреби в повному перепланування мережі. Це особливо важливо в умовах постійного зростання кількості IoT-пристроїв та еволюції вимог до архітектур.

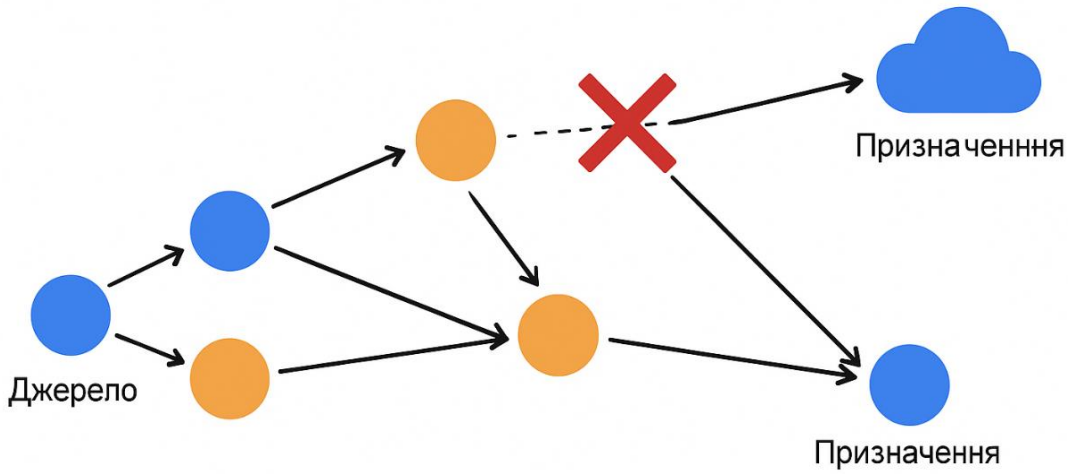


Рисунок 2.1 – Приклад схеми адаптивної маршрутизації

У розподілених системах Інтернету речей (IoT), що базуються на туманних обчисленнях, пріоритезація даних відіграє важливу роль в оптимізації обробки та передачі інформації. Завдяки великій кількості пристроїв, підключених до мережі, щоденно генеруються терабайти даних, які мають різну важливість, часочутливість та вимоги до безпеки. Саме тому система повинна вміти розрізняти, які дані мають бути оброблені негайно, а які можуть бути передані або збережені пізніше.

Пріоритезація даних дозволяє встановити рівень важливості інформації, що передається, на основі її контексту, джерела, цілі та поточних умов у мережі. Наприклад, повідомлення з медичного датчика про критичний стан пацієнта

повинно мати абсолютний пріоритет порівняно з інформацією про температуру навколишнього середовища. У системах туманних обчислень такий підхід дозволяє максимально ефективно використовувати обмежені ресурси на крайовому та туманному рівнях, де приймаються ключові рішення до передачі даних у хмару.

У практиці впровадження пріоритезації використовуються спеціальні алгоритми, які автоматично класифікують дані на критичні, середньої важливості та другорядні. Ця класифікація може базуватись на політиках, визначених системними адміністраторами, або на динамічному аналізі ситуацій у режимі реального часу. Відповідні мітки або теги додаються до кожного пакету даних, що надходить до мережі, після чого система приймає рішення, як саме діяти з ним: передати негайно, зберегти локально або відкласти на пізніше – рисунок 2.2.

Важливо, що пріоритезація тісно пов’язана з іншими аспектами архітектури системи. Вона впливає на маршрутизацію, балансування навантаження, використання черг обробки та планування обчислювальних задач. Наприклад, використання пріоритетних черг дозволяє гарантувати, що критичні повідомлення отримають обслуговування без затримок, навіть за умов високого навантаження на мережу або обчислювальні ресурси. У свою чергу, інтелектуальні системи можуть автоматично коригувати політики пріоритезації відповідно до змін у поведінці користувачів або змін у мережевому середовищі.

Впровадження механізмів пріоритезації сприяє значному зниженню загального часу реакції системи, підвищенню ефективності обробки даних у реальному часі та зменшенню навантаження на канали зв’язку. Це особливо важливо у критичних інфраструктурах, таких як медицина, розумні міста чи виробничі підприємства, де будь-яке запізнення в обробці даних може мати серйозні наслідки.

У результаті пріоритезація даних у туманних IoT-системах забезпечує не лише технічну ефективність, а й підвищує загальну стійкість та надійність інформаційного середовища, дозволяючи системі адаптуватися до реальних умов та забезпечувати найвищий рівень якості обслуговування.

У складних системах важливо враховувати не лише сам вміст даних, але й метаінформацію, наприклад, часові мітки, ідентифікатори джерел, тип пристрою або контекст спостереження. Це дозволяє створити більш гнучку багаторівневу модель пріоритетів, в якій окремі типи пристроїв (наприклад, відеокамери, сенсори руху, мікрофони) можуть мати власні внутрішні політики передачі даних. У результаті пріоритезація здійснюється не тільки між категоріями даних, але й всередині кожної категорії відповідно до ситуаційної важливості.

Окрему роль відіграє концепція QoS (Quality of Service) – забезпечення необхідної якості обслуговування для даних з високим пріоритетом. У цьому випадку використовується не лише логічна ієрархія важливості, а й технічні методи управління пропускною здатністю каналів зв’язку, контроль затримок і втрат пакетів. Дані, що позначені як високопріоритетні, можуть отримувати пріоритетний доступ до каналів з низькою затримкою або бути передані альтернативними маршрутами, якщо основний шлях перевантажений.

Зростає також роль машинного навчання у процесах пріоритезації. Алгоритми на основі штучного інтелекту здатні динамічно вивчати шаблони поведінки пристроїв і користувачів та адаптувати пріоритети на основі зміни середовища або бізнес-вимог. Наприклад, в системі "розумного міста" інтенсивність руху на перехресті може автоматично підвищити пріоритет для датчиків дорожнього навантаження в певний час доби. Таким чином, пріоритет не є жорстко зафіксованим, а динамічно формується на основі контексту.

Крім цього, пріоритезація може здійснюватися на кількох рівнях системи – на пристрої (локально), на шлюзі туманного рівня, або на хмарному сервері. У багатьох випадках локальна фільтрація даних дозволяє зменшити навантаження ще до передачі в мережу, тобто дані з низьким пріоритетом навіть не надсилаються далі, якщо це не потрібно. Такий підхід підвищує енергоефективність системи, що є критично важливим для батарейних IoT-пристроїв.

Іншим важливим аспектом є інтеграція пріоритезації з безпекою. Дані, що мають високий пріоритет, часто містять конфіденційну або критичну інформацію. Відповідно, системи мають забезпечити не лише швидку обробку, а й захист таких

даних на всіх етапах передачі: шифрування, автентифікацію та контроль доступу. Пріоритезація, поєднана з механізмами безпеки, дозволяє уникнути витоків інформації або спотворень у критичних каналах зв'язку.

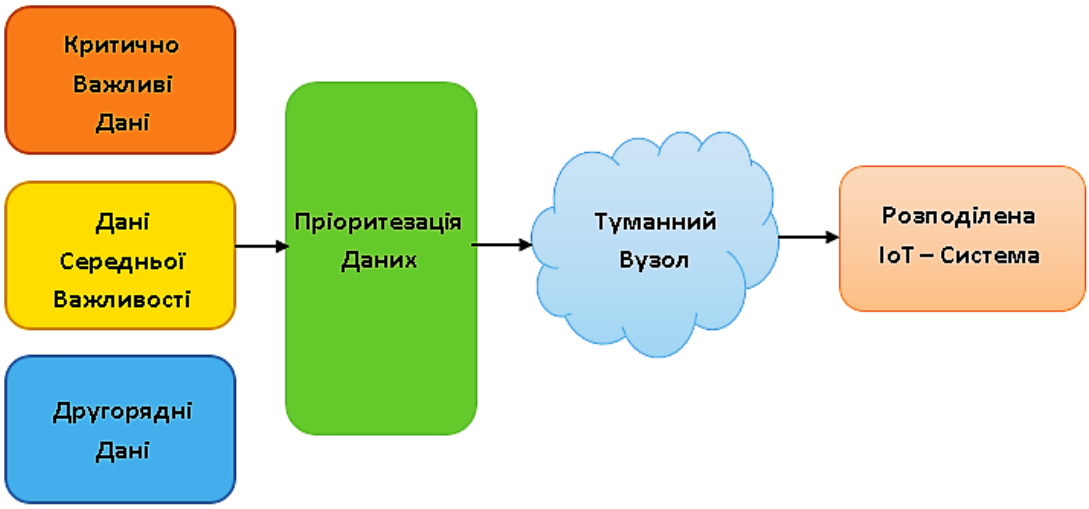


Рисунок 2.2 – Приклад схеми пріоритезації даних

Механізми агрегації даних у розподілених системах, зокрема в архітектурі туманних обчислень для IoT, є важливим інструментом для ефективного управління великими обсягами інформації, що генеруються численними сенсорами та пристроями. Основна ідея агрегації полягає у зведенні, фільтрації та попередній обробці даних безпосередньо на рівні, наближеному до джерела їх виникнення, до того, як вони будуть передані далі – на туманний або хмарний рівень. Це дозволяє зменшити трафік, затримки у передачі, навантаження на канали зв'язку та обчислювальні ресурси, а також підвищити енергоефективність системи – рисунок 2.3.

Завдяки агрегації, повторювані або малозначущі дані не потрапляють у ядро обробки, а об'єднуються в більш узагальнені форми. Наприклад, температурні сенсори, які фіксують значення кожні кілька секунд, можуть передавати середнє значення за хвилину, замість кожного окремого зчитування. Це мінімізує кількість повідомлень і дозволяє виявляти лише значущі зміни. У разі просторової агрегації

дані з кількох пристроїв, розміщених у межах однієї зони, поєднуються для створення єдиної репрезентативної картини. У багатьох випадках дані передаються лише при фіксації певної події, наприклад, перевищенні допустимого рівня забруднення повітря або виявленні несанкціонованого руху.

Механізми агрегації можуть також поєднуватися із засобами стиснення та попереднього аналізу, щоб ще більше зменшити навантаження на інфраструктуру. У системах туманних обчислень важливою перевагою є те, що ця обробка виконується ближче до джерела даних, завдяки чому досягається вища швидкість реагування, підвищується надійність і забезпечується більший контроль над безпекою та приватністю даних.

У промислових додатках агрегація використовується для контролю параметрів обладнання, де зведені дані допомагають виявляти потенційні несправності. У розумних містах агреговані дані про транспорт або стан навколишнього середовища використовуються для прийняття управлінських рішень. У сфері охорони здоров’я агрегація дозволяє формувати загальну картину стану пацієнта на основі численних показників життєдіяльності, які надходять від різних сенсорів.

Найбільшим викликом у реалізації ефективної агрегації є необхідність збереження балансу між зменшенням обсягів даних і збереженням їхньої релевантності. Важливо, щоб система могла адаптувати свою стратегію агрегації залежно від типу даних, сценарію використання та критичності інформації. Деякі ситуації потребують передачі "сирих" даних для глибокої аналітики, тоді як в інших достатньо базової статистики. Гнучкість, адаптивність та інтелектуальність механізмів агрегації – запорука ефективності IoT-рішень на базі туманних обчислень у сучасних умовах швидкого зростання обсягів даних.

Окрім зменшення обсягів даних, механізми агрегації відіграють ключову роль у забезпеченні структурованості інформаційного потоку, що надходить із гетерогенних IoT-пристроїв. Ці пристрої часто різняться за характеристиками, частотою генерації даних, обчислювальними можливостями та енергоспоживанням. Агрегація дозволяє уніфікувати ці потоки, створюючи єдиний

узгоджений формат даних, зручний для подальшої обробки в туманних або хмарних вузлах.

У багатьох випадках агрегація реалізується не лише у вигляді зведення значень, а й через побудову метаданих – описових характеристик набору даних. Наприклад, замість передачі повного набору телеметричних даних, система може сформувати опис їхніх змін, коливань, тенденцій або виявлених аномалій. Це значно підвищує інформативність при мінімальному обсязі трафіку.

Ще одним важливим аспектом є безпека та конфіденційність даних. Завдяки тому, що агрегація відбувається на ранніх етапах обробки, можливо локалізувати чутливу інформацію на периферійних вузлах, не передаючи її в централізовані середовища. Це особливо актуально у сфері медицини, фінансів або розумного житла, де обробка персональних даних вимагає дотримання високих стандартів приватності. Агрегація дозволяє передавати лише узагальнену інформацію, що не містить ідентифікуючих ознак.

Існують також динамічні схеми агрегації, які автоматично змінюють спосіб обробки залежно від мережевої ситуації або навантаження на вузли. Наприклад, за високої затримки чи перевантаженості каналу дані агрегуються агресивніше (із вищим ступенем узагальнення), тоді як у стабільних умовах можуть передаватися більш детально. Це дозволяє системі залишатися адаптивною до змін середовища без зниження якості обслуговування.

Особливу цінність механізми агрегації мають у контексті обмежених ресурсів IoT-вузлів. У пристроях із мінімальним запасом енергії, таких як сенсори з живленням від батарей або harvesting-пристрої, агрегація мінімізує частоту передачі, знижуючи енергоспоживання. Такі пристрої можуть виконувати попереднє об’єднання даних у пакетах з мінімальним розміром, що оптимізує витрати енергії на передачу.

На практиці використовуються різноманітні методи агрегації: часові вікна, фільтрація по порогу, семплінг, згладжування, обчислення трендів, стиснення тощо. Вибір методики залежить від цільового застосування, характеристик мережі, критичності даних та пріоритетів системи. Розробка ефективних стратегій агрегації

є важливою складовою сучасних досліджень у сфері туманних обчислень, спрямованих на створення масштабованих, інтелектуальних та ресурсоефективних IoT-систем.

Рисунок 2.3 – Схема механізмів агрегації даних

Компресія даних є критично важливим етапом в інфраструктурі IoT та туманних обчисленнях, оскільки дозволяє ефективно зменшувати обсяг переданої інформації між вузлами, а також скорочувати затрати на зберігання та обробку. Особливо важливо це у середовищах з обмеженою пропускною здатністю, де навіть незначна оптимізація обсягу даних може мати значний вплив на продуктивність системи.

У системах IoT дані, що надходять від численних сенсорів і пристроїв, часто є надлишковими або містять повторювану інформацію. Компресія дозволяє зменшити цей надлишок ще на рівні крайових або туманних вузлів, перш ніж інформація буде передана далі по ієрархії, зокрема до хмарних центрів обробки. Це

мінімізує навантаження на мережу та знижує загальну затримку при передачі. У свою чергу, це дозволяє забезпечити кращу швидкодію для критично важливих додатків, таких як системи моніторингу здоров’я або розумне місто.

На рівні обчислювальних моделей використовуються як класичні, так і спеціалізовані алгоритми стиснення. Наприклад, у середовищах з обмеженими ресурсами перевага надається алгоритмам з низькою обчислювальною складністю та мінімальними вимогами до пам’яті. Найчастіше застосовуються адаптовані версії алгоритмів на зразок Huffman coding, LZW або Delta encoding, які дають змогу ефективно стиснути великі масиви схожих або корельованих значень, зокрема у часових рядах.

У більш складних сценаріях можуть застосовуватись алгоритми з елементами штучного інтелекту, які дозволяють виявити шаблони в потоках даних і застосувати до них відповідну компресію на основі класифікації, прогнозування чи моделювання поведінки пристроїв. Такий підхід є особливо ефективним для прогнозованих або періодичних потоків даних, де заздалегідь відома структура інформації.

Особливу увагу варто приділяти компресії з урахуванням критичності даних. Наприклад, у ситуаціях, коли від втрати частини інформації не знижується точність загального аналізу, можливо застосовувати втратні методи стиснення. Це дозволяє досягати значно вищого коефіцієнта стиснення у порівнянні з безвтратними методами. Водночас, для конфіденційних або надзвичайно чутливих даних важливо зберігати цілісність і повноту, що накладає обмеження на вибір алгоритмів.

Також у сучасних системах компресія може бути динамічною – тобто адаптуватися до умов мережі, поточного навантаження або контексту. Це особливо корисно у середовищах з високою змінністю параметрів (наприклад, у мобільних IoT-сценаріях), де фіксована модель стиснення може бути неефективною.

Компресія даних у контексті туманних обчислень також тісно пов’язана з іншими аспектами оптимізації – наприклад, з агрегацією та пріоритезацією даних. Ефективна інтеграція цих механізмів дозволяє не лише зменшити обсяг інформації,

що передається, але й мінімізувати витрати енергії пристроїв, а також підвищити якість обслуговування (QoS) у реальному часі – рисунок 2.4.

Окрім технічного зменшення обсягу даних, компресія має стратегічне значення у забезпеченні узгодженості між обмеженими можливостями пристроїв і зростаючими вимогами до обробки в реальному часі.

Однією з важливих характеристик, яка визначає вибір методів стиснення в IoT-середовищах, є енергетична ефективність. Багато IoT-пристроїв живляться від акумуляторів або мають обмежений доступ до джерел живлення, тому компресія не повинна створювати додаткового навантаження на обчислювальні ресурси. У цьому контексті актуальними є методи, що виконують стиснення «на льоту», не вимагаючи попереднього накопичення великих масивів даних. Наприклад, поточкове (on-the-fly) кодування або легковагові предиктивні алгоритми дозволяють ефективно стискати дані без затримки в часі.

Значну роль у компресії відіграє також тип переданих даних. Наприклад, сенсори можуть генерувати числові, мультимедійні або бінарні дані. Кожен з цих типів потребує специфічних методів. Для текстових і числових потоків часто застосовують коди змінної довжини, наприклад Arithmetic coding. У випадку передавання зображень, відео або аудіо потоку – використовуються методи DCT (дискретне косинусне перетворення), вейвлет-перетворення або even lightweight варіанти JPEG/MPEG, адаптовані для обмежених платформ.

У розподілених системах із багатьма вузлами компресія також сприяє зменшенню надлишковості при дублюванні даних між вузлами. Наприклад, у сценаріях, коли дані одного сенсора можуть бути передані кількома маршрутами, система може виконати первинне стиснення, зменшуючи не лише обсяг, а й дублікати. У поєднанні з механізмами дедуплікації це дозволяє значно оптимізувати маршрутизацію трафіку в туманній архітектурі.

Іншим цікавим напрямом є використання компресії з урахуванням семантичного контексту даних. У таких випадках система аналізує не лише форму, а й значення інформації. Наприклад, дані з сенсорів температури можуть передаватись не у вигляді точних значень, а як семантичні категорії («нормально»,

«високо», «низько»), що займає менше місця при збереженні релевантності інформації. Такий підхід відкриває можливості для гібридної компресії – поєднання математичних методів з когнітивним моделюванням.

Рисунок 2.4 – Схема принципу роботи компресії даних

У туманних середовищах компресія може бути також пов’язана з концепцією локальної обробки (preprocessing). Наприклад, вузли Fog можуть агрегувати або фільтрувати лише релевантну інформацію, перш ніж виконувати стиснення. У деяких сценаріях навіть застосовуються адаптивні компресійні протоколи, які динамічно змінюють алгоритм стиснення в залежності від пропускної здатності каналу, затримок або типу пристрою-відправника.

Окремо слід розглядати безпеку у процесі стиснення. У випадках, коли компресія передує шифруванню, існує ризик втрати конфіденційності через можливість реконструкції вихідних даних. Для уникнення цього, у сучасних розробках застосовуються методи одночасної компресії та шифрування (наприклад, Compressive Sensing Encryption), що дозволяють зберігати цілісність і захищеність даних без збільшення навантаження на систему.

2.2 Класифікація даних та забезпечення приватності

У системах IoT та туманних обчисленнях передача і обробка даних відбувається в умовах обмежених ресурсів, розподілених середовищ і високих вимог до безпеки. Одним із ключових завдань у таких системах є класифікація даних – тобто розподіл інформації за рівнями важливості, чутливості та критичності. Це дозволяє ефективно керувати ризиками, пов’язаними з приватністю, захистом даних і контролем доступу.

Класифікація даних в IoT-середовищах є ключовим процесом, який дозволяє впорядкувати інформацію за ступенем її чутливості, важливості та критичності для системи. Це особливо важливо в умовах обмежених ресурсів fog-архітектур, де ефективна обробка і захист даних значною мірою залежить від їх правильного розподілу за категоріями. Дані, що генеруються IoT-пристроями, можуть значно відрізнятись за своїми характеристиками та ризиками, пов’язаними з їх передачею, зберіганням або обробкою.

У загальному випадку дані поділяються на конфіденційні, критично важливі для функціонування системи та звичайні (неконфіденційні).

Конфіденційні дані в системах Інтернету речей (IoT) та туманних обчисленнях становлять одну з найбільш уразливих і водночас найбільш критичних категорій інформації. Їх визначальними характеристиками є висока чутливість до несанкціонованого доступу, потенційна шкода при розголошенні та необхідність забезпечення цілісності під час передачі й зберігання. У контексті IoT конфіденційна інформація може охоплювати персональні дані користувачів,

біометричні або медичні показники, фінансові транзакції, дані геолокації, цифрові ідентифікатори пристроїв (MAC-адреси, IP-ідентифікатори), а також керуючі сигнали для фізичних пристроїв, таких як замки, камери, системи охорони тощо.

Однією з головних загроз при обробці конфіденційних даних у децентралізованих середовищах є ризик їх перехоплення або маніпуляції під час передачі через відкриті або ненадійні канали зв'язку. Це зумовлює необхідність впровадження обов'язкових засобів шифрування як на рівні каналу, так і на рівні самих даних. Крім того, щоб запобігти можливості відновлення первинної інформації зі стиснутих чи агрегованих даних, часто використовуються методи анонімізації, псевдонімізації та маскування.

Особливої уваги вимагає обробка конфіденційної інформації в умовах обмежених ресурсів fog- або edge-пристроїв. В таких сценаріях забезпечення безпеки ускладнюється нестачею обчислювальної потужності, оперативної пам'яті або енергозабезпечення. Тому при розробці архітектур туманних обчислень велике значення має застосування полегшених криптографічних алгоритмів, апаратної підтримки шифрування, а також реалізація локальної обробки, яка дозволяє уникати пересилання конфіденційних даних до віддалених серверів або хмарних платформ.

Конфіденційні дані, як правило, обробляються відповідно до політик конфіденційності, які встановлюють правила зберігання, доступу, термінів утримання та умов знищення. Такі політики можуть бути адаптивними, тобто змінюватись в залежності від контексту, прав доступу, рівня ризику або зміни середовища. Наприклад, ті самі дані можуть вважатися конфіденційними лише в певному географічному регіоні або за наявності конкретного користувача в системі.

Не менш важливим є аудит та моніторинг дій над конфіденційними даними. Інфраструктура повинна бути здатною реєструвати всі операції доступу до таких даних, що дозволяє виявляти потенційні інциденти безпеки, забезпечувати відповідність до нормативно-правових актів (наприклад, GDPR) та підтримувати прозорість функціонування системи.

Критично важливі дані в контексті IoT-систем і туманних обчислень являють собою інформацію, яка безпосередньо впливає на стабільність, функціонування та безпечність як окремих пристроїв, так і всієї інфраструктури. Вони мають вирішальне значення у сценаріях, де затримки в обробці або втрата даних можуть призвести до серйозних наслідків – від зниження якості сервісу до небезпеки для життя або порушення роботи критичних систем. До таких даних належать, зокрема, сигнали від сенсорів аварійної безпеки, дані телеметрії в транспорті, показники контролю виробничих процесів, а також операційні команди для дій пристроїв у режимі реального часу.

На відміну від інших категорій, ці дані мають підвищені вимоги до надійності доставки та мінімізації затримок у передачі. Будь-які спроби оптимізувати обсяг або маршрутизацію таких потоків мають враховувати їхню чутливість до часу. Саме тому в туманній архітектурі для критично важливої інформації часто реалізується обробка на найближчому можливому рівні – без передачі до хмари або центральних вузлів. Це дозволяє не лише скоротити час реакції системи, а й зменшити залежність від стабільності зовнішніх мереж.

Особливістю таких даних є також їхня контекстна природа: вони можуть втрачати цінність буквально через кілька секунд після генерації. Через це важливо не просто захистити їх від втрат, а забезпечити можливість негайного реагування. У деяких випадках для них резервуються окремі канали зв’язку або встановлюється пріоритет у системах передачі трафіку. Крім того, у випадку з динамічними або мобільними IoT-сценаріями, система має бути здатна переорієнтовуватись і самостійно шукати альтернативні маршрути або вузли обробки, якщо первинний шлях стає недоступним.

Важливо розуміти, що критично важлива інформація не завжди є чутливою чи конфіденційною – її значення полягає передусім у її ролі для процесів управління та підтримки безперервності. Це зумовлює необхідність її відокремленого розгляду в системах класифікації даних, оскільки критерії її захисту більше стосуються доступності, точності й оперативності, а не приватності.

Звичайні або неконфіденційні дані в системах IoT та туманних обчислень охоплюють інформацію, яка не містить чутливих персональних відомостей, не становить комерційної або державної таємниці й не має критичного значення для безпеки чи безперервності роботи інфраструктури. Це можуть бути стандартні параметри навколишнього середовища, службова інформація про стан пристроїв, публічні повідомлення або агреговані показники, які не дозволяють ідентифікувати конкретного користувача чи об’єкт.

У більшості випадків така інформація використовується для звітності, статистичного аналізу, трендової аналітики або довготривалого моніторингу без негайної необхідності у швидкому реагуванні. Наприклад, температурні значення у відкритому середовищі, рівень шуму, кількість підключених пристроїв або середні показники енергоспоживання за день можуть вважатися неконфіденційними, якщо вони не пов’язані з конкретним користувачем або критичним застосуванням. Це дозволяє обробляти їх з використанням менш ресурсоємних засобів і без потреби застосування складних механізмів шифрування чи спеціального контролю доступу.

Передача таких даних у туманній архітектурі, як правило, допускає більшу гнучкість щодо маршрутів, часу затримки чи частоти оновлення. Вони можуть бути буферизовані, пакетовані для подальшої обробки або зберігатись локально до моменту, коли система матиме вільні ресурси для передачі. Крім того, неконфіденційні дані часто виступають як фоновий інформаційний потік, на основі якого працюють алгоритми оптимізації мережевого трафіку, балансування навантаження або прогнозування попиту.

Важливо, що такі дані можуть перетворитися на чутливі або критично важливі у разі поєднання з іншими типами інформації. Наприклад, на перший погляд нейтральні дані про геолокацію або активність пристрою, в комбінації з певним контекстом, можуть стати основою для профілювання користувача чи визначення його поведінки. Саме тому в сучасних системах безпеки навіть звичайні дані іноді обробляються з урахуванням потенційного ризику, залежно від середовища використання.

Забезпечення приватності в системах Інтернету речей (IoT) та туманних обчислень є важливим аспектом, який охоплює кілька рівнів та підходів для захисту особистої інформації користувачів та конфіденційних даних. Це завдання ускладнюється через величезну кількість пристроїв, які збирають, обробляють та передають дані в реальному часі. Приватність в таких системах передбачає не лише захист чутливих даних, а й впровадження технологій, що дозволяють зберігати анонімність користувачів і забезпечити контроль над їхніми персональними відомостями.

Шифрування даних є основним методом забезпечення конфіденційності та безпеки в системах Інтернету речей (IoT) та туманних обчислень, перетворюючи інформацію в нечитабельний формат для захисту від несанкціонованого доступу. Для цього використовуються різні алгоритми, що забезпечують захист як при зберіганні даних, так і під час їх передачі через мережу.

Існують два основні типи шифрування: симетричне і асиметричне. У симетричному шифруванні для шифрування і дешифрування використовується один і той самий ключ. Такий метод є швидким і ефективним, але вимагає надійного обміну ключами між сторонами, щоб уникнути компрометації. Наприклад, алгоритм AES є широко застосовуваним для шифрування даних в системах IoT завдяки своїй швидкості і високому рівню безпеки.

Асиметричне шифрування використовує пару ключів: публічний, який може бути відкрито переданий іншим користувачам, і приватний, що зберігається в таємниці. Це дозволяє безпечно обмінюватися даними, оскільки тільки особа, що володіє приватним ключем, може розшифрувати зашифровану інформацію. Протоколи, такі як RSA, є прикладами асиметричного шифрування, що використовуються для забезпечення захисту даних при їх передачі через Інтернет.

Шифрування не обмежується лише зберіганням даних, а також використовується для захисту комунікації між пристроями через мережу. Протоколи, як TLS і IPSec, забезпечують шифрування даних на етапі їх передачі, забезпечуючи безпечне з'єднання між пристроями або серверами. TLS, наприклад,

використовується для забезпечення захищеного з'єднання через Інтернет, а IPSec – для захисту даних на рівні мережі.

У системах IoT, де пристрої часто мають обмежені ресурси, швидкість обробки і ефективність шифрування є важливими факторами. Зазвичай використовуються менш ресурсоємні алгоритми, такі як AES, або інші методи, що підходять для обмежених обчислювальних потужностей. Для оптимізації використання шифрування можуть застосовуватися апаратні модулі безпеки або інші методи, що дозволяють ефективно працювати з обмеженими ресурсами пристроїв.

Особливу увагу слід приділяти безпеці управління ключами. Це включає як безпечне зберігання, так і обмін ключами між пристроями або серверами. У багатьох системах IoT використовуються апаратні модулі для зберігання ключів або спеціальні механізми для безпечного обміну ними через мережу. Алгоритми, такі як Diffie-Hellman, можуть бути використані для безпечного обміну ключами в розподілених системах, забезпечуючи захист навіть у разі, коли канали зв'язку є незахищеними.

Анонімність і псевдонімізація є важливими механізмами забезпечення конфіденційності та захисту приватності в системах Інтернету речей (IoT) і туманних обчисленнях. Вони дозволяють приховувати або змінювати ідентифікаційні дані користувачів, зберігаючи при цьому можливість обробляти інформацію без розкриття особистої ідентичності.

Анонімність передбачає, що всі особисті дані, які можуть бути використані для ідентифікації користувача, будуть повністю видалені або змінені таким чином, що їх неможливо буде віднести до конкретної особи. Це означає, що навіть при доступі до даних зловмисники не зможуть встановити, хто є власником цих даних, не маючи додаткової інформації. Анонімність зазвичай використовується в ситуаціях, коли особисті дані не є необхідними для виконання завдання, але є потреба у зборі або обробці інформації. Наприклад, у випадку збору статистичних даних про трафік у місті або аналізу активності без необхідності ідентифікувати

окремих користувачів. Анонімізація може включати в себе зміну ідентифікаційних ознак або видалення особистих відомостей з баз даних.

Псевдонімізація є менш радикальним методом, ніж анонімізація, оскільки він не передбачає повного видалення ідентифікаційної інформації, а заміняє її на псевдоніми – неособисті ідентифікатори, що не дозволяють прямо пов'язати дані з конкретною особою. У разі псевдонімізації можна відновити реальну ідентичність користувача за наявності спеціального ключа або додаткової інформації. Цей метод дозволяє зберігати зв'язок між даними, наприклад, для подальшого аналізу чи верифікації, але без небезпеки для конфіденційності особистості. Псевдонімізація використовується в ситуаціях, коли необхідно проводити обробку даних для статистики чи прогнозування, але без прямого доступу до особистої інформації користувача.

У системах IoT анонімізація та псевдонімізація використовуються для захисту даних, що збираються з пристроїв або сенсорів. Наприклад, інформація про місцезнаходження або активність користувача може бути зібрана без ідентифікації конкретної особи або заміщена на псевдонімізовані дані, що не порушують приватності. Це дозволяє здійснювати аналіз даних, зберігаючи при цьому високий рівень конфіденційності.

Псевдонімізація і анонімність мають велике значення для забезпечення приватності в туманних обчисленнях, де дані часто обробляються на периферії і передаються на сервери для подальшого аналізу. У цьому контексті важливо, щоб передача інформації між пристроями та серверами не порушувала приватності користувачів. Проте, на відміну від анонімізації, псевдонімізація дозволяє зберегти певний рівень зв'язку між даними для подальшої обробки або верифікації, що важливо для певних застосувань.

Обидва методи сприяють захисту приватності, зменшуючи можливість несанкціонованого використання особистих даних, і дозволяють організаціям збирати і обробляти інформацію в безпечний спосіб. Псевдонімізація і анонімність можуть бути використані для зниження ризиків при зборі даних, особливо в системах, де обробляється велика кількість чутливих або персональних відомостей.

Вони дозволяють зберігати баланс між необхідністю отримання даних для аналітики та забезпеченням безпеки та конфіденційності.

Контроль доступу є одним із ключових компонентів забезпечення безпеки в системах Інтернету речей (IoT) та туманних обчислень. Він дозволяє управляти тим, хто і з якими правами може отримувати доступ до інформації та ресурсів у системі. Основною метою контролю доступу є гарантування, що тільки уповноважені користувачі та пристрої можуть взаємодіяти з критичними системами або отримувати чутливу інформацію, при цьому мінімізуючи можливість несанкціонованого доступу та атак.

Контроль доступу реалізується через різні механізми, які визначають правила доступу до ресурсів. Одним з основних підходів є використання механізмів автентифікації та авторизації. Автентифікація є процесом перевірки особистості користувача або пристрою, тоді як авторизація визначає, які права та ресурси доступні після успішної автентифікації.

Існують різні моделі контролю доступу, серед яких найбільш поширеними є модель доступу на основі ролей (RBAC) та модель доступу на основі атрибутів (ABAC). У моделі RBAC доступ до ресурсів визначається на основі ролей, які призначаються користувачам. Кожній ролі надаються певні права доступу, і користувачам надаються ці ролі залежно від їхніх обов'язків або функцій у системі. Модель ABAC, в свою чергу, надає доступ до ресурсів на основі атрибутів, таких як час, місцезнаходження або конкретні характеристики користувача чи пристрою.

Для систем IoT і туманних обчислень важливо забезпечити не тільки правильний доступ до даних, але й можливість динамічного управління доступом у реальному часі. Оскільки в таких системах може бути велика кількість різноманітних пристроїв, які взаємодіють один з одним, контроль доступу повинен бути адаптованим до змінюваних умов і вимог. Наприклад, в системах IoT можуть застосовуватись тимчасові обмеження доступу для пристроїв, залежно від їхніх характеристик або поточного стану. Це дозволяє знизити ризики несанкціонованого доступу до критичних ресурсів, забезпечуючи контроль над тим, хто і коли має право взаємодіяти з системою.

Механізми контролю доступу можуть включати в себе мультифакторну автентифікацію, що вимагає використання більше одного методу для підтвердження особистості користувача або пристрою. Це може бути, наприклад, поєднання пароля з біометричними даними або одноразовими кодами, що надсилаються на мобільний пристрій. Крім того, для підвищення безпеки можуть використовуватися технології шифрування для забезпечення захищеного каналу зв'язку між пристроями та серверами, що ускладнює перехоплення та маніпуляцію даними під час обміну. Важливим аспектом є також управління доступом на основі контексту, яке враховує додаткові фактори, такі як місцезнаходження, поточний стан пристроїв або час доби. Наприклад, деякі пристрої можуть мати обмеження доступу в залежності від того, чи є користувач фізично присутнім у певному місці, або ж доступ може бути дозволений тільки за певних умов, таких як підключення до безпечної мережі. Контроль доступу є особливо важливим у туманних обчисленнях, де дані обробляються на локальних пристроях, а потім передаються на центральні сервери для подальшої обробки. Тут необхідно чітко визначити, які пристрої можуть обробляти і зберігати дані, а також гарантувати, що тільки уповноважені користувачі можуть отримувати доступ до результатів обробки. Це зменшує ризик несанкціонованого доступу або витоку даних, особливо якщо вони містять чутливу інформацію. В системах IoT, де можуть бути обмежені ресурси та різноманітні типи пристроїв, контроль доступу повинен бути оптимізованим, щоб не створювати додаткове навантаження на систему. Для цього можуть застосовуватися механізми легковагого управління доступом, які працюють в реальному часі, а також протоколи, що дозволяють швидко адаптувати права доступу залежно від ситуації.

2.3 Висновки до другого розділу

Отже, розділі розглянуто ключові аспекти контролю передачі даних у системах IoT-фог обчислень, що є важливими для забезпечення ефективної, безпечної та приватної роботи цих розподілених середовищ.

Методи оптимізації передачі даних у таких системах є критичними для забезпечення високої продуктивності та низької затримки. Завдяки фільтрації, агрегації, стисненню та буферизації даних вдалося знизити навантаження на мережу, зменшити кількість передач та забезпечити ефективне використання обмежених ресурсів. Розподілене оброблення даних, яке здійснюється безпосередньо на периферії мережі, дозволяє зменшити затримки і знизити навантаження на центральні сервери, що є важливим для систем, що потребують обробки великих обсягів інформації в реальному часі.

Щодо класифікації даних, було з'ясовано, що IoT-фог середовища обробляють різні типи даних, від неконфіденційних до критичних і чутливих. Правильна класифікація даних дозволяє застосовувати відповідні методи захисту, що знижують ризики витоку або маніпуляцій з інформацією. Дані, що мають високу чутливість або критичне значення для безпеки, вимагають застосування найсучасніших методів шифрування, а також анонімізації або псевдонімізації для захисту приватності користувачів.

Забезпечення приватності в IoT-фог середовищах є важливою частиною загальної стратегії безпеки. Шифрування даних, контроль доступу, анонімізація та псевдонімізація дозволяють створити надійний захист особистої інформації та даних, що обробляються в таких системах. Водночас, правильне управління передачею даних і їх класифікацією допомагає мінімізувати ризики несанкціонованого доступу та витоку даних.

Ефективне управління передачі даних, їх оптимізація, класифікація та забезпечення приватності є важливими аспектами, що сприяють високій ефективності роботи систем IoT та туманних обчислень. Вони дозволяють досягнути балансу між швидкістю обробки даних, збереженням ресурсів та захистом чутливих даних, що є важливим для забезпечення безпеки та приватності у таких середовищах.

3 АРХІТЕКТУРА ІОТ ТУМАННИХ ОБЧИСЛЕНЬ НА ОСНОВІ КОНТРОЛЮ ПЕРЕДАЧІ ДАНИХ ТА ТУМАННИХ ОБЧИСЛЕНЬ

3.1 Архітектура IoT туманних обчислень

Загальна концепція архітектури IoT туманних обчислень включає в себе інтеграцію туманних шлюзів для обробки даних, зниження затримок передачі інформації та забезпечення конфіденційності. Ця архітектура створюється для того, щоб оптимізувати ефективність IoT-систем за рахунок застосування туманних обчислень, де обчислення і зберігання даних частково виконуються на локальних шлюзах, що зменшує навантаження на хмарну інфраструктуру і знижує затримки. Основною метою цієї архітектури є надання гнучкої і масштабованої системи, що здатна адаптуватися до змінних умов роботи мережі, обсягів трафіку та вимог до конфіденційності даних.

Основні компоненти архітектури IoT туманних обчислень включають кілька ключових елементів, кожен з яких виконує специфічні функції для забезпечення ефективної роботи системи.

Кінцеві пристрої – це основні елементи, що відповідають за збір даних з навколишнього середовища через різноманітні сенсори та акутори. Кінцеві пристрої можуть бути представлені датчиками температури, вологості, руху та іншими IoT-пристроями, які використовуються для моніторингу фізичних явищ у реальному часі.

Туманний шлюз виступає в ролі з'єднувального елементу між кінцевими пристроями та хмарною інфраструктурою. Він приймає дані від кінцевих пристроїв, проводить попередню обробку (наприклад, фільтрацію, агрегацію, передобробку даних) і може зберігати деякі з цих даних локально, щоб знизити затримки та обсяг трафіку, що передається до хмари. Також він має можливість здійснювати обробку повідомлень залежно від політики конфіденційності (наприклад, приватні дані можуть зберігатися локально, тоді як інші – передаватися в хмару).

Хмара виконує функції зберігання великих обсягів даних і більш складної аналітики. Хмара є центральним елементом для обробки великих масивів даних, які надходять від різних туманних шлюзів, і зберігає ці дані для подальшого аналізу або архівування. У хмарі можуть здійснюватися аналітичні операції, які потребують значних обчислювальних ресурсів, а також зберігання даних, що не є приватними.

Механізм контролю передачі даних відповідає за динамічне управління потоком даних між кінцевими пристроями, туманним шлюзом і хмарною інфраструктурою. Він дозволяє налаштовувати, які дані передавати на хмару, а які обробляти локально на шлюзі. Також цей компонент враховує політики конфіденційності і може адаптуватися до змін в умовах мережі та навантаження.

Механізм забезпечення конфіденційності відповідає за обробку приватних і публічних даних, забезпечуючи їх відповідне зберігання та обробку. Приватні дані зберігаються локально на туманному шлюзі, що дозволяє знизити ризик їх витоку при передачі через мережу, в той час як інші, менш чутливі дані передаються в хмару для подальшої обробки.

Програмне забезпечення для управління з'єднаннями – компонент, що керує з'єднанням між туманним шлюзом і хмарною інфраструктурою, враховуючи можливі зміни в мережі, втрату з'єднання або зміни в пропускній здатності каналу. Він також дозволяє автоматично перемикати режим обробки (локальний або хмарний) залежно від стану з'єднання.

Веб-інтерфейс дозволяє користувачам або адміністраторам керувати системою, вибираючи режим обробки даних, налаштовувати політики конфіденційності та моніторити стан всієї архітектури. Це інтерфейс, через який можна управляти всіма аспектами роботи системи, включаючи налаштування режимів обробки, зберігання даних, а також взаємодію з кінцевими пристроями і хмарою.

Система управління сесіями та ключами – компонент, що гарантує безпеку даних, керуючи сесіями та шифрувальними ключами, що використовуються для захисту переданих даних. Він забезпечує конфіденційність повідомлень, що

передаються між кінцевими пристроями, туманними шлюзами і хмарною інфраструктурою, шифруючи чутливі дані під час передачі та зберігання.

Важливою особливістю архітектури є здатність туманного шлюзу обробляти дані на місці, що значно знижує затримки в порівнянні з традиційними хмарними рішеннями. Це особливо важливо для реальних застосунків, де час обробки має критичне значення.

У рамках архітектури передбачено механізм забезпечення конфіденційності даних: приватні дані обробляються і зберігаються локально на туманному шлюзі, тоді як непри приватні дані можуть бути відправлені до хмари. Це забезпечує високий рівень захисту даних і відповідність політикам конфіденційності.

Система здатна автоматично переходити в автономний режим роботи в разі втрати з'єднання з хмарою. Це дозволяє продовжувати обробку та зберігання даних локально на туманному шлюзі, що важливо для систем, де безперервність обробки є критично важливою.

Механізм контролю за передачею даних дозволяє системі адаптуватися до зміни умов з'єднання та вимог до конфіденційності, що робить архітектуру гнучкою та ефективною в різних умовах роботи.

Локальна обробка даних зменшує час, необхідний для передачі та обробки інформації, що робить систему швидшою і більш ефективною для критичних застосунків IoT.

Збереження конфіденційних даних на туманному шлюзі забезпечує їхній захист від несанкціонованого доступу, що є важливим аспектом для безпеки системи.

Перехід в оффлайн-режим при втраті з'єднання з хмарою гарантує безперервну роботу системи, що особливо важливо для застосунків, де постійний доступ до мережі не гарантований.

Архітектура дозволяє легко масштабувати систему шляхом додавання нових туманних шлюзів, що дозволяє адаптувати її до збільшення навантаження або зміни умов роботи мережі.

Експериментальне тестування цієї архітектури дасть можливість перевірити її ефективність у реальних умовах.

Очікується, що час обробки повідомлень у локальному режимі буде значно знижений, порівняно з хмарними рішеннями. Це дозволить підтвердити гіпотезу про зниження затримок за рахунок локальної обробки.

Тестування має показати, що система коректно обробляє приватні дані, забезпечуючи їхнє збереження на туманному шлюзі, а також перевірити здатність архітектури відповідати вимогам конфіденційності.

Під час втрати з'єднання з хмарною інфраструктурою система повинна автоматично перейти в оффлайн-режим, продовжуючи працювати без перерв. Це дозволить оцінити здатність системи до автономного функціонування.

Порівняльне тестування з існуючими архітектурами, такими як TTN і ChirpStack, дозволить підтвердити стабільність роботи та ефективність запропонованої системи в умовах змінних мережевих умов.

3.1.1 Архітектура А

У стандартній архітектурі LoRaWAN усі шлюзи є окремими модулями, що складаються з RF-концентратора, антени та обчислювального блоку. RF-концентратор здатен приймати пакети, надіслані різними кінцевими пристроями, тоді як обчислювальний блок здійснює перетворення та ретрансляцію цих пакетів.

Мережеві та прикладні сервери відокремлені від шлюзів і працюють на виділеному обладнанні, до якого шлюзи підключаються через Інтернет.

Запропонована архітектура А ґрунтується на ідеї інтеграції всіх стандартних елементів мережі LoRaWAN в один пристрій, який називається туманним шлюзом. Таким чином, кожен шлюз, окрім приймання, перетворення та ретрансляції пакетів, виконує також функції приватних мережевого та прикладного серверів. Такий підхід може забезпечити швидшу реакцію на поточні умови завдяки обробці актуальних даних на місці.

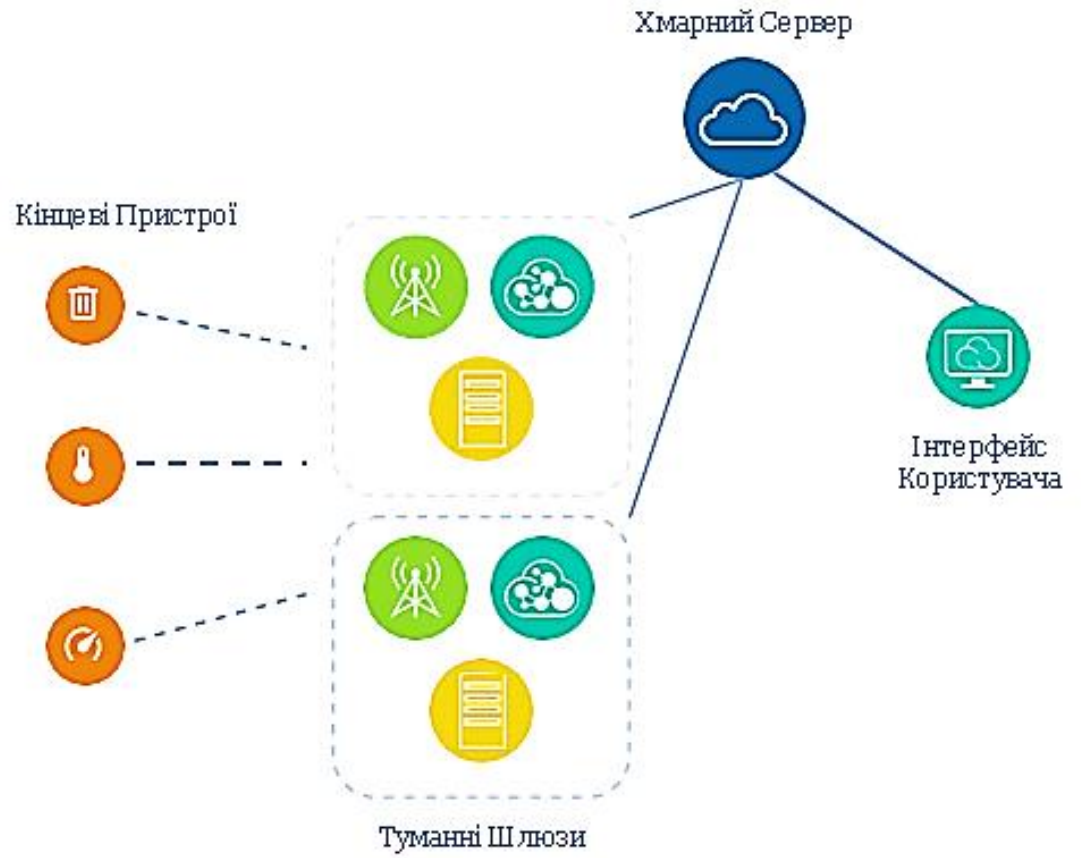


Рисунок 3.1 – Діаграма Архітектури А

Однак обробка на рівні туману не завжди є необхідною або можливою. Особливо це стосується ситуацій, коли передається великий обсяг даних – тоді може виникнути надмірне навантаження з огляду на обмежені обчислювальні ресурси туманних шлюзів. Для вирішення цієї проблеми в архітектурі додатково передбачено віддалений хмарний сервер. У всій мережі LoRaWAN наявний лише один хмарний сервер, спільний для всіх туманних шлюзів. Цей сервер забезпечує веб-інтерфейс та API для керування туманними шлюзами, пристроями й додатками – рисунок 3.1.

3.1.2 Архітектура В

Архітектура В пропонує дещо інший підхід до оптимізації. На відміну від архітектури А, запропонована архітектура В не вимагає інтеграції мережевого та

прикладного серверів у кожен туманний шлюз – вони інтегруються лише в один шлюз у мережі. Решта шлюзів працює так само, як у стандартній архітектурі LoRaWAN, – тобто тільки приймає, перетворює та ретранслює пакети.

Для коректної роботи цієї архітектури необхідно застосувати певну модель комунікації. Оскільки один складний вузол керує кількома вузлами з обмеженою логікою, застосовується модель master/slave.

Модель master/slave – це модель комунікації, за якої один вузол (master) керує кількома підлеглими вузлами (slave) та опосередковує комунікацію між ними. У даному випадку як master, так і slave-вузли виконують функції шлюзів. Підлеглий шлюз функціонує як стандартний шлюз LoRaWAN і передає весь трафік головному шлюзу. Головний шлюз містить мережевий і прикладний сервери, які розшифровують корисне навантаження та керують ретрансляцією пакетів від підлеглих шлюзів.

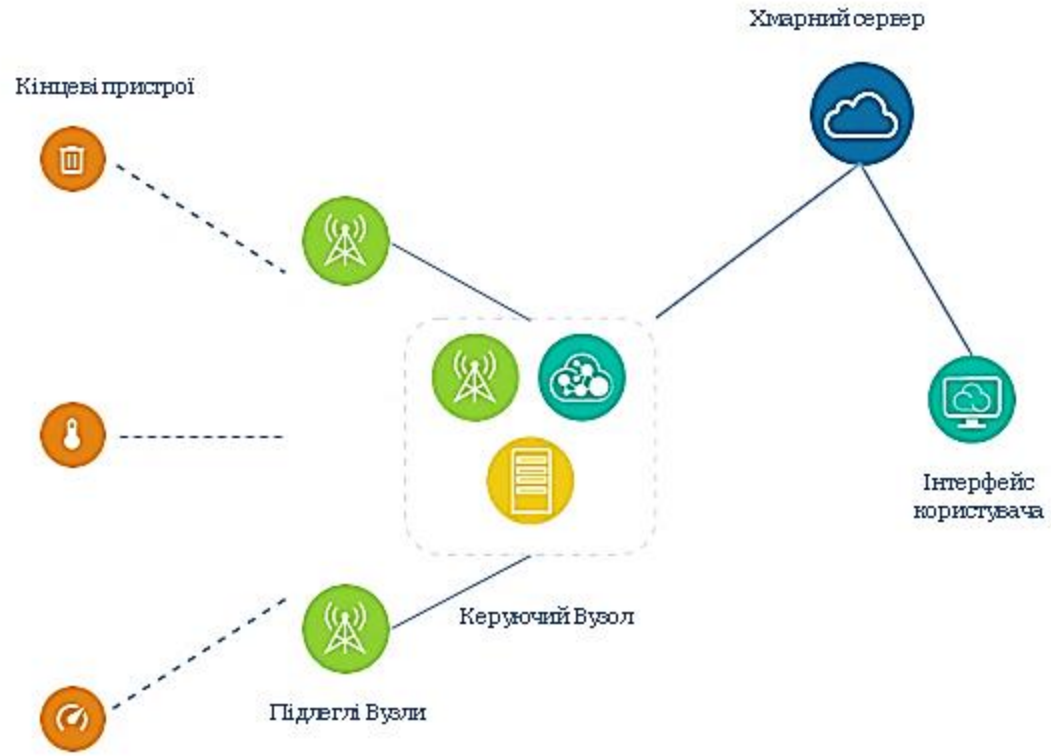


Рисунок 3.2 – Діаграма Архітектури B

З огляду на обмежені обчислювальні та сховищні ресурси туманного шлюзу, в архітектурі також передбачено віддалений хмарний сервер. Цей хмарний сервер з'єднаний з головним шлюзом через Інтернет і використовується для зберігання даних у довготривалій перспективі – рисунок 3.2.

Основною перевагою цієї архітектури є усунення потреби в підключенні кожного шлюзу до Інтернету. У стандартній архітектурі LoRaWAN інтернет-з'єднання є обов'язковим, оскільки мережеві та прикладні сервери розміщені в хмарі. В архітектурі В лише головний шлюз підключений до Інтернету, оскільки саме він здійснює зв'язок з віддаленим хмарним сервером.

3.1.3 Архітектура С

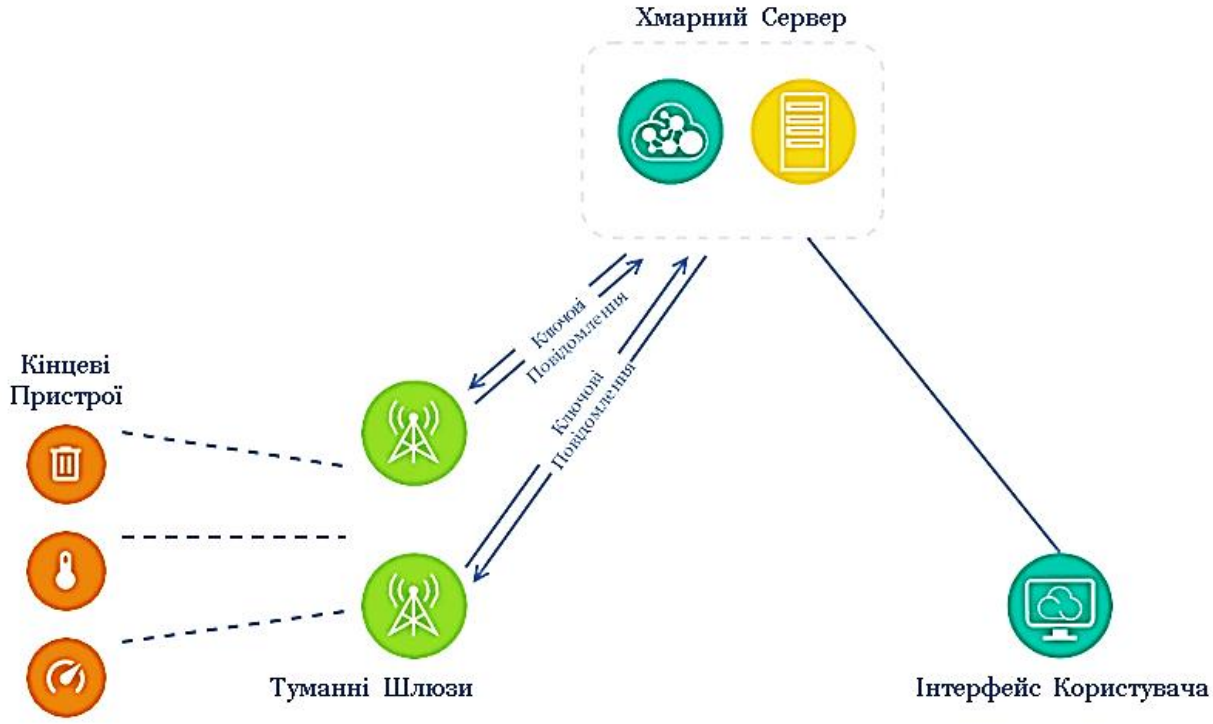


Рисунок 3.3 – Діаграма Архітектури С

Архітектура С ґрунтується на принципово іншому підході до оптимізації, ніж попередні архітектури. Замість повторного розміщення окремих компонентів стандартної архітектури LoRaWAN, головна відмінність полягає у способі

комунікації між шлюзом і сервером додатків. Таким чином, з точки зору структури, архітектура C не відрізняється від стандартної архітектури LoRaWAN.

Як зазначалося раніше, через наскрізне шифрування сам шлюз LoRa не має доступу до сесійних ключів і не може розшифрувати корисне навантаження – це завдання виконує сервер додатків. Основна ідея цієї оптимізації полягає в тому, що між певним туманним шлюзом і сервером додатків відбувається узгодження, у результаті якого шлюз отримує та зберігає необхідні ключі для окремих кінцевих пристроїв у захищеній локальній базі даних. У такий спосіб туманний шлюз надсилає запит до сервера додатків лише під час першого сеансу зв'язку з конкретним кінцевим пристроєм або за необхідності. Таким чином, взаємодія із сервером додатків зводиться до мінімуму – рисунок 3.3.

Для комунікації спочатку потрібно успішно пройти персоналізацію та активацію конкретного кінцевого пристрою, який може передавати повідомлення через мережу LoRaWAN.

Після того, як туманний шлюз у цій мережі LoRaWAN отримує повідомлення, він перевіряє DevAddr у заголовку повідомлення та порівнює його з вмістом локальної бази даних і якщо вказаний DevAddr наявний у локальній базі даних, це означає, що пристрій уже взаємодіяв із цим туманним шлюзом, і сесійні ключі можуть бути завантажені для розшифрування корисного навантаження. Якщо ж DevAddr відсутній у локальній базі даних, це означає перше з'єднання з цим пристроєм. У такому випадку туманний шлюз повинен узгодити сесійні ключі із сервером додатків і зберегти їх у приватній локальній базі даних для подальшого використання.

У підсумку, після завантаження сесійних ключів туманний шлюз може розшифрувати та обробити корисне навантаження повідомлення.

3.2 Алгоритми класифікації та метрики оцінки моделей

У цьому розділі розглядаються обрані алгоритми класифікації, які пізніше будуть використані у вирішенні завдань.

Класифікація є доцільною, коли тип мітки – категоріальний, і мета алгоритму – поділити набір даних на певні класи.

3.2.1 Дерево рішень

Цей алгоритм застосовується як для класифікації, так і для регресії. Він побудований у вигляді дерева, де кожен внутрішній вузол – це умова, а лист – кінцевий результат (категорія). Лист також може містити вектор ймовірностей. Дерево рішень працює з числовими та категоріальними ознаками.

Щоб уникнути перенавчання, задають мінімальну кількість прикладів для кожного листа або обмежують максимальну глибину дерева. Також можна "обрізати" дерево, видаляючи гілки з незначними ознаками.

Головна перевага – простота інтерпретації. Недоліки: зі збільшенням кількості ознак збільшується час навчання; також є ризик перенавчання, якщо не вжити згаданих заходів.

3.2.2 Випадковий ліс

Головний недолік дерева рішень – схильність до перенавчання, що знижує точність на тестових даних. Випадковий ліс – це ансамбль дерев рішень, які працюють разом. Його концепція базується на принципі "мудрості натовпу". Навіть якщо окремі дерева помиляються, більшість дає правильну відповідь.

Перевагою ж є висока ефективність і точність, навіть з великими наборами даних; уникнення перенавчання завдяки бутстрепінгу (використання підвибірок даних).

3.2.3 Метод k найближчих сусідів (k-НС)

Метод k найближчих сусідів (k-НС), як і дерева рішень чи випадкові ліси, використовується для виконання завдань класифікації та регресії. Його принцип

роботи полягає в припущенні, що об’єкти з подібними ознаками зазвичай мають схожі класи. Тому для прийняття рішення необхідно знайти найближчих сусідів об'єкта, що аналізується, з використанням певної метрики відстані. Оскільки використання лише одного сусіда може спричинити помилки через чутливість до викидів, метод застосовує пошук k найближчих сусідів, де k – параметр, що вимагає ретельного налаштування. Надто мале значення k може викликати переобучення, а надто велике – втрату точності через включення надмірної кількості непотрібних сусідів. Тому зазвичай значення k вибирається невеликим цілим числом.

Окремим різновидом є зважений k-НС, в якому сусідам присвоюються ваги за допомогою ядрової функції. Ця функція забезпечує вищу вагу ближчим сусідам і нижчу – більш віддаленим.

Один з важливих аспектів алгоритму – це вибір методу вимірювання відстані. У разі числових атрибутів найчастіше застосовується евклідова відстань – формула 3.1

$$\sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}. \tag{3.1}$$

де x та y – вектори ознак, а n – кількість ознак у векторах. Недоліком евклідової відстані є її чутливість до екстремальних значень.

Для категоріальних ознак, навпаки, використовують відстань Хеммінга, яка визначається кількістю позицій, у яких символи двох векторів відрізняються. Якщо значення збігаються – відстань дорівнює нулю, інакше – одиниці.

3.2.4 Метод машин опорних векторів

Метод машин опорних векторів (МОВ) є ефективним інструментом лінійної класифікації, що застосовується здебільшого у задачах бінарного розділення класів. Він базується на ідеї побудови такої гіперплощини, яка максимально розділяє приклади двох класів у багатовимірному просторі ознак. У разі

двовимірного простору гіперплощина зводиться до прямої, а в тривимірному – до площини.

Мета алгоритму – визначити таку гіперплощину, яка забезпечує максимальну відстань до найближчих прикладів обох класів. Точки, що лежать на межі цього розділення, називаються опорними векторами.

У ситуаціях, коли дані не можна розділити лінійно, застосовують ядрові функції, які перетворюють початковий простір ознак у новий – з вищою розмірністю, де вже можлива лінійна класифікація.

Метод МОВ має високу здатність до узагальнення, що дозволяє уникнути перенавчання. Проте, значним недоліком є низька інтерпретованість моделі.

3.2.5 Наївний байєсівський класифікатор

Наївний баєсівський підхід базується на припущенні про незалежність усіх вхідних ознак та їх однакову важливість. Така гіпотеза рідко виконується у практиці, однак, незважаючи на це, алгоритм демонструє хороші результати точності.

Класифікація здійснюється за допомогою формули умовної ймовірності Байєса, яка дозволяє розрахувати ймовірність події А за умови, що сталася подія В – формула 3.2.

$$P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}. \quad (3.2)$$

Для задач класифікації наївний баєсівський алгоритм обчислює ймовірність належності об’єкта з вектором ознак X=[x1,...,xn] до класу с – формула 3.3.

$$P(c|x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n|c)\cdot P(c)}{P(x_1,\dots,x_n)}. \quad (3.3)$$

Існує проблема, коли деяке значення ознаки ніколи не зустрічається в певному класі в навчальному наборі. Це призводить до нульової умовної ймовірності, яка анулює весь добуток, що спотворює результат. Для уникнення цієї ситуації застосовують згладжування Лапласа, додаючи невелике значення (наприклад, 1) до кожної частотної комірки, забезпечуючи ненульові ймовірності.

Перевагою алгоритму є його ефективність при роботі з малими обсягами даних.

3.2.6 Штучні нейронні мережі (ШНМ)

Концепція штучних нейронних мереж (ШНМ) була розроблена на основі принципів функціонування нейронних мереж у мозку живих організмів. Біологічні нейрони приймають сигнали через дендрити, передають їх через аксони, а взаємодія між нейронами здійснюється через синапси, які можуть підсилювати або послаблювати сигнал.

В аналогічний спосіб, штучні нейрони (перцептрони) мають входи та виходи. Кожен вхід має ваговий коефіцієнт (w), що визначає його вплив, а також додається значення зміщення (bias). Вихід формується за допомогою активаційної функції.

Однією з найбільш поширених активаційних функцій є сигмоїдна функція, яка створює згладжений, неперервний вихід, обмежений значеннями між 0 і 1. Вихід сигмоїдного нейрона обчислюється за формулами 3.4 і 3.5.

$$z = bias + \sum_{i=1}^{n} w_i x_i. \tag{3.4}$$

$$y = \frac{1}{1+e^{-z}}, \ . \tag{3.5}$$

ШНМ формуються з трьох основних типів шарів: вхідного, який передає дані мережі; одного або кількох прихованих шарів, які виконують обробку; та вихідного шару, що формує остаточне рішення – рисунок 3.4.

Якщо зв’язки між шарами направлені лише вперед – від входу до виходу – така архітектура називається прямопередавальною нейронною мережею.

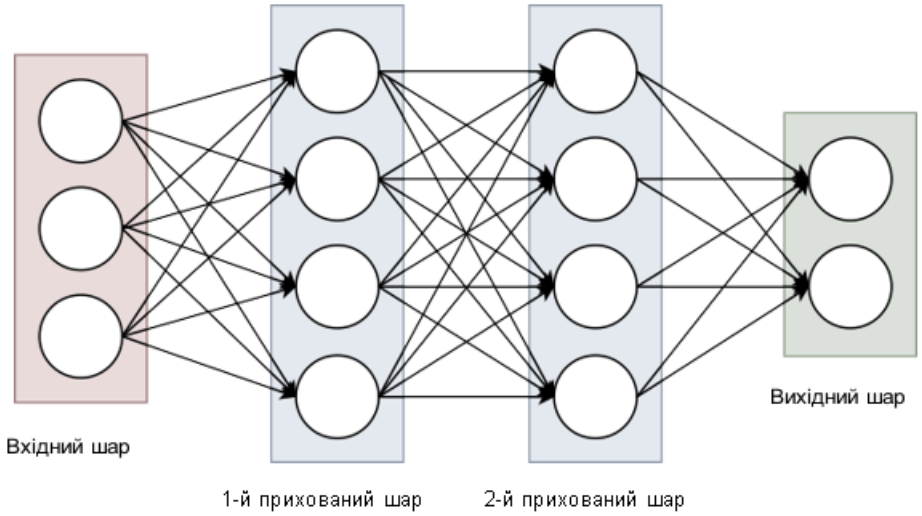
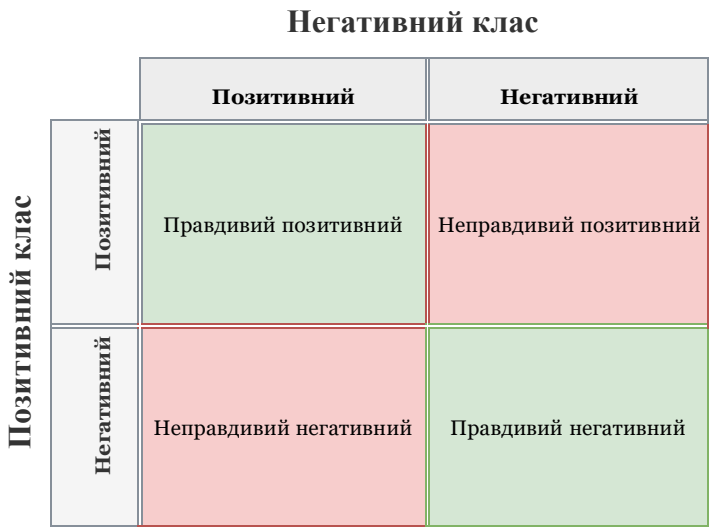


Рисунок 3.4 – Діаграма ШНМ

Для досягнення найточніших результатів штучною нейронною мережею (ШНМ), необхідно правильно налаштувати вагові коефіцієнти між окремими нейронами. Ці ваги коригуються в процесі навчання, щоб підібрати оптимальні значення. Одним з методів навчання багатошарової ШНМ є зворотне поширення помилки (backpropagation), яке дозволяє покращити початкові випадково згенеровані ваги. Метою цього методу є зменшення різниці між фактичним і очікуваним виходом мережі. На рисунку 3.4 показано схему прямої нейронної мережі з двома прихованими шарами, де вузли відповідають окремим штучним нейронам.

У зв’язку з особливостями багатошарових ШНМ, дуже важливо правильно вибрати їхню топологію. Якщо у мережі недостатньо нейронів, вона не зможе ефективно навчитися виявляти закономірності у даних. Натомість надмірна кількість нейронів може призвести до перенавчання, коли модель занадто точно повторює навчальні дані, але погано справляється з новими вхідними даними.



Матриця 3.1 — Матриця плутанини (confusion matrix)

Після завершення навчання моделі необхідно оцінити її точність. Для цього використовуються певні метрики, зокрема точність (accuracy) та рівень помилок (error rate). Загальна точність розраховується за формулою 3.6.

$$A = \frac{N_{correct}}{N_{total}} \cdot 100\%. \tag{3.6}$$

де $N_{correct}$ – кількість правильно класифікованих прикладів, а N_{total} – загальна кількість вхідних даних.

Для більш глибокого аналізу результатів замість простої точності застосовується матриця плутанини (confusion matrix). У цій матриці стовпці представляють передбачені моделі значення, а рядки – фактичні класи. Значення на головній діагоналі – це правильні класифікації. Помилки – це значення поза діагоналлю – матриця 3.1

Розглянемо основні метрики оцінювання.

Точність (Accuracy) – частка правильних прогнозів серед усіх передбачень – формула 3.7.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \cdot 100\%. \tag{3.7}$$

$$Recall = \frac{TP}{TP+FN} \cdot 100\%. \quad (3.8)$$

Повнота / Recall (чутливість, TPR) – здатність правильно ідентифікувати позитивні приклади – формула 3.8.

Специфічність (Specificity) – здатність правильно ідентифікувати негативні приклади – формула 3.9

$$Specificity = \frac{TN}{TN+FP} \cdot 100\%. \quad (3.9)$$

Рівень хибнопозитивних результатів (False Positive Rate) – формула 3.10

$$FPR = \frac{FP}{FP+TN} \cdot 100\%. \quad (3.10)$$

Точність передбачень (Precision) – формула 3.11

$$Precision = \frac{TP}{TP+FP} \cdot 100\%. \quad (3.11)$$

Для візуального аналізу ефективності класифікатора часто застосовується ROC-крива. Вона відображає співвідношення між рівнем хибнопозитивних результатів (1 - специфічність) по осі X та рівнем істиннопозитивних результатів (Recall) по осі Y.

Ідеальний класифікатор буде представлений точкою (0,1), тоді як випадковий класифікатор – лінією від (0,0) до (1,1). Площа під ROC-кривою (AUC) використовується як показник загальної ефективності моделі – рисунок 3.4.

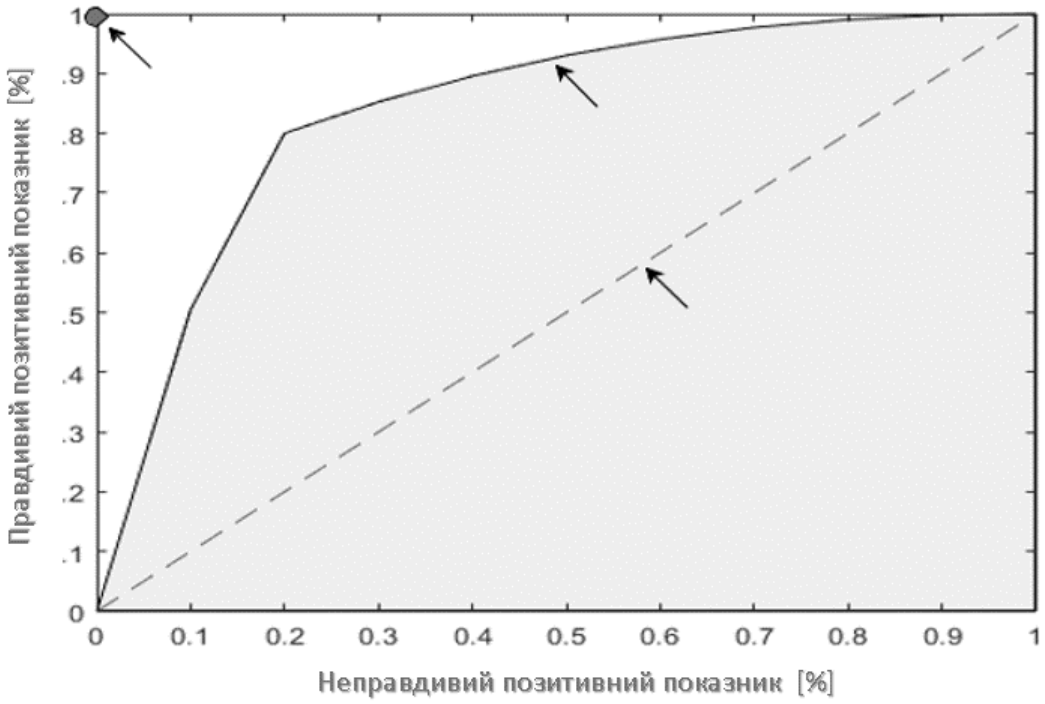


Рисунок 3.5 – Приклад ROC-кривої

Крос-валідація є методом перевірки ефективності моделей машинного навчання. Вона дає змогу об'єктивно оцінити якість моделі та допомагає обрати найвідповідніший алгоритм для вирішення конкретного завдання прогнозування. На відміну від звичайного навчання, при крос-валідації використовується не весь набір даних одночасно, а лише певні його підмножини, що дозволяє зменшити ймовірність перенавчання або недонавчання. Існує декілька популярних стратегій розбиття даних для крос-валідації.

Метод відкладеної вибірки (Hold-out) – найпростіший підхід, при якому набір даних ділять на дві частини: одна використовується для навчання моделі, інша – для її тестування. Цей метод є обчислювально ефективним, але результати залежать від того, як саме були поділені дані.

k-кратна крос-валідація (k-fold cross-validation) – удосконалення попереднього методу. Набір даних ділиться на *k* частин (звичайно k = 5 або 10). Модель тренується *k–1* раз на різних комбінаціях даних, а кожна частина по черзі виступає тестовою. Такий підхід дає точніші результати, але потребує більше часу на обчислення.

Багаторазове випадкове розділення (Repeated random sub-sampling) – дані випадково розділяються на навчальну та тестову частини кілька разів. На відміну від k-fold, тут кількість розділень не обмежена кількістю підмножин. Перевага – гнучкість у виборі розмірів наборів, але є ризик, що окремі приклади можуть не потрапити до жодної підмножини або ж навпаки – потрапляти надто часто.

Leave-One-Out (LOO) – це крайній випадок методу k-fold, коли k дорівнює загальній кількості прикладів у наборі. Для кожної ітерації модель навчається на всіх прикладах, окрім одного, який використовується для тестування. Хоча цей метод є дуже точним, він потребує значних обчислювальних ресурсів.

3.3 Висновки до третього розділу

Отже, було проведено всебічний аналіз можливостей застосування туманних обчислень у середовищах IoT, а також запропоновано три варіанти архітектур, які реалізують різні підходи до розміщення функціональних компонентів LoRaWAN-мережі та контролю передавання даних.

Архітектура A передбачає повну інтеграцію серверів мережі та додатків безпосередньо в шлюз туманних обчислень. Це дозволяє виконувати обробку даних ближче до джерела, що потенційно зменшує затримки та навантаження на хмарні ресурси. Проте, при високій інтенсивності трафіку така архітектура може бути обмеженою через обмежені ресурси fog-вузлів, що вирішується використанням централізованого хмарного сервера для управління мережею.

Архітектура B реалізує модель "ведучий-підлеглий", у якій лише один шлюз містить сервери мережі та додатків, а інші шлюзи виконують базові функції передавання пакетів. Такий підхід дозволяє зменшити потребу в підключенні кожного шлюзу до Інтернету, оскільки лише головний шлюз комунікує з хмарним сервером, що оптимізує використання інфраструктури у розподілених середовищах.

Архітектура C зберігає структурну схожість зі стандартною LoRaWAN, проте вводить новий принцип обробки: шлюзи туманних обчислень зберігають ключі

сесії у своїй локальній базі даних після початкового запиту до сервера додатків. Завдяки цьому кількість звернень до хмарного сервера мінімізується, що зменшує затримки та підвищує автономність fog-вузлів.

Крім архітектурних рішень, у розділі було розглянуто найдоцільніші алгоритми машинного навчання для fog-середовищ: дерева рішень, випадкові ліси, k-ближчих сусідів, метод опорних векторів, наївний баєсівський класифікатор та штучні нейронні мережі. Було встановлено, що прості алгоритми краще підходять для обмежених обчислювальних середовищ, тоді як складні моделі, як-от нейронні мережі, потребують додаткової оптимізації, але можуть забезпечити вищу точність.

Оцінка ефективності моделей здійснювалась за допомогою таких метрик, як точність, повнота, специфічність, точність позитивного прогнозу, ROC-криві та площа під кривою (AUC). Це дозволяє враховувати як загальні характеристики моделі, так і її здатність до роботи з дисбалансованими даними, що є типовим у реальних IoT-сценаріях.

4 ПРОЄКТУВАННЯ, РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА IoT-ФОГ АРХІТЕКТУРИ

4.1 Порівняльний аналіз та моделювання архітектур A, B та C

У даному розділі порівнюються три запропоновані у розділі 3.1 архітектури IoT-мереж, а також обговорюється, яка з них є оптимальною. Це впливає на загальні характеристики, доступні кінцевим пристроям.

Було дотримано таких властивостей, щоб реалізувати парадигму туманних обчислень:

1) Близьке розташування кінцевих пристроїв дозволяє мережі швидко реагувати на їхні потреби та адаптуватися до вимог.

2) Забезпечення повної кооперацією між рівнями туману і хмари. У разі необхідності обчислення і зберігання даних можуть передаватися між рівнями.

3) Низька затримка, яка досягалася за рахунок обробки даних на краю мережі, що особливо важливо для додатків, чутливих до часу, як-от сигналізаційні чи медичні системи.

Аналізуючи стандартну архітектуру LoRaWAN визначено, що така архітектура дозволяє створювати приватні мережі, а її відкритість забезпечує можливість змінювати архітектуру мережі. Основними компонентами стандартної архітектури LoRaWAN є кінцеві пристрої (датчикики або виконавчі пристрої), шлюзи, мережеві сервери та сервери додатків.

Шлюз виступає мостом між низькопотужною мережею LoRaWAN і мережею з високою пропускною здатністю (Ethernet, Wi-Fi, Cellular) – приймає пакети, перетворює їх у IP-пакети та пересилає далі.

Мережевий сервер керує мережею, фільтрує дублікати пакетів, виконує перевірку безпеки, контролює адаптивну швидкість передачі тощо.

Сервер додатків – це кінцевий пункт для даних, де вирішується, що з ними робити (наприклад, зберегти в базі даних або візуалізувати).

Оскільки шлюз є найближчим до кінцевих пристроїв елементом і виконує пасивну роль у процесі обробки даних, то уся обробка і зберігання відбуваються на сервері додатків у хмарі, що відповідає парадигмі хмарних обчислень.

Шляхом внесення певних змін до базових елементів LoRaWAN їх можна трансформувати в архітектуру, яка відповідає парадигмі туманних обчислень. Основна проблема полягає в шифруванні корисного навантаження, яке здійснюється за принципом "end-to-end" між кінцевим пристроєм та сервером додатків. Шлюз не має доступу до розшифрованих даних, отже, зберігання та обробка інформації неможливі.

У пам’яті кінцевого пристрою зберігаються два ключі:

- Мережевий сесійний ключ (NwkSKey) – використовується для перевірки цілісності повідомлень між сервером та пристроєм.
- Сесійний ключ додатку (AppSKey) – використовується для шифрування/дешифрування корисного навантаження.

Оскільки цілісність даних додатку не перевіряється, існує ризик зміни повідомлення мережею. Проте мережеві сервери зазвичай вважаються надійними.

Порівнюючи три різні архітектури з метою визначення найоптимальнішої, результати залежатимуть від обчислювальної продуктивності компонентів, однак за умови однакового апаратного забезпечення для кожної з запропонованих архітектур результати моделювання не повинні містити жодних викривлень.

Для отримання часу обслуговування компонентів кожної архітектури було проведено серію експериментальних вимірювань. Як апаратне забезпечення для LoRa-шлюзу використовувалась плата Raspberry Pi 4 Model B з 2 ГБ оперативної пам’яті та операційною системою Raspbian. Для функцій мережевого та прикладного серверів було обрано віртуальну машину з 2 ГБ оперативної пам’яті та ОС Debian, яка забезпечувала достатню обчислювальну потужність для виконання необхідних функцій.

Оскільки час розповсюдження повідомлення у радіосередовищі не мав істотного значення для цих експериментів, час виконання вимірювався з моменту отримання повідомлення шлюзом.

Для запропонованих архітектур A і B час виконання шлюзу (форвардера пакетів) (Tpktfwd) вимірювався до моменту пересилання повідомлення на мережевий сервер, який далі здійснював обробку повідомлення. Час виконання мережевого сервера вимірювався з моменту отримання ним повідомлення до моменту його обробки та пересилання на сервер застосунку. Час виконання на сервері застосунку вимірювався з моменту отримання повідомлення до моменту додавання розшифрованого корисного навантаження до бази даних. Експериментальні вимірювання були проведені для 1000 повідомлень. За результатами вимірювань були обчислені середні значення часу – таблиця 4.1.

Таблиця 4.1 – Час виконання компонентів

Компонент	Середній час виконання (мс)
Форвардер пакетів (шлюз)	0.16
Мережевий сервер	194.62
Сервер застосунку	15.16

У запропонованій Архітектурі C використовується інший підхід, оскільки форвардер пакетів спеціально модифікований для виконання функцій туманних обчислень. У цьому випадку форвардер пакетів не лише пересилає дані на мережевий сервер, але й самостійно декодує повідомлення та розшифровує корисне навантаження. Тому до експериментально виміряного часу виконання форвардера необхідно додати час виконання окремих функцій туманних обчислень, а саме: декодування повідомлення (T_{decode}), розшифрування корисного навантаження ($T_{decrypt}$) та подальшого збереження цього навантаження в базі даних ($T_{database}$):

$$T_{fog} = T_{decode} + T_{decrypt} + T_{database}, \quad (4.1)$$

$$T_{total} = T_{pktfwd} + T_{fog}. \quad (4.2)$$

Наведені вище формули 4.1 і 4.2 використовуються для визначення загального часу виконання обробки повідомлення, необхідного для виконання функцій туманних обчислень, пов'язаних із запропонованою Архітектурою С. Як і в попередньому випадку, для експериментальних вимірювань як апаратне забезпечення шлюзу використовувалась плата Raspberry Pi 4 Model B. Було поступово виконано 1000 вимірювань кожної з окремо згаданих функцій. Для кожного вимірювання декодування та розшифрування алгоритм генерував псевдовипадковий рядок щонайменше з 20 символів. Подальше збереження метаданих та корисного навантаження в базу даних також вимірювалося для отримання фінального компоненту часу. Результати цих експериментальних вимірювань наведені в таблиці 4.2.

Таблиця 4.2 – Час виконання шлюзу туману

Компонент часу	Середній час виконання (мс)
T_{decode}	1.95
$T_{decrypt}$	1.19
$T_{database}$	7.05
T_{fog}	10.19
T_{total}	10.35

Під час моделювання запропонованої Архітектури А першим блоком, включеним у моделювання, був блок з n-серверами, що представляв форвардер пакетів на стороні шлюзу. Тут середній час обслуговування був встановлений як 0.16 мс, а кількість ліній (n) – 8. Запит потім передавався до блоку з нескінченною кількістю серверів (n = ∞), який представляв мережевий сервер із середнім часом обслуговування 194.62 мс, а далі – до наступного ланцюгового блоку нескінченного сервера, який представляв сервер застосунку зі середнім часом обслуговування 15.16 мс.

Після обробки цими ланцюговими системами черг, сервер застосунку перевіряв, чи є дані приватними. Якщо так – поточний час обробки зберігався, і цикл моделювання завершувався. Якщо ні – сервер застосунку пересилав повідомлення до віддаленого хмарного сервера. Затримка на лінії становила 0.5 мс і враховувалася в загальному часі виконання. Оскільки шлюз туману передавав дані до хмарного сервера після обробки мережевим і прикладним серверами, хмарний сервер лише записував дані до бази даних. Середній час його виконання був тому незначним і становив 7 мс відповідно до результатів експериментальних вимірювань. Після збереження розшифрованого корисного навантаження в базу даних загальний час виконання зберігався, і цикл моделювання завершувався – рисунок 4.1.

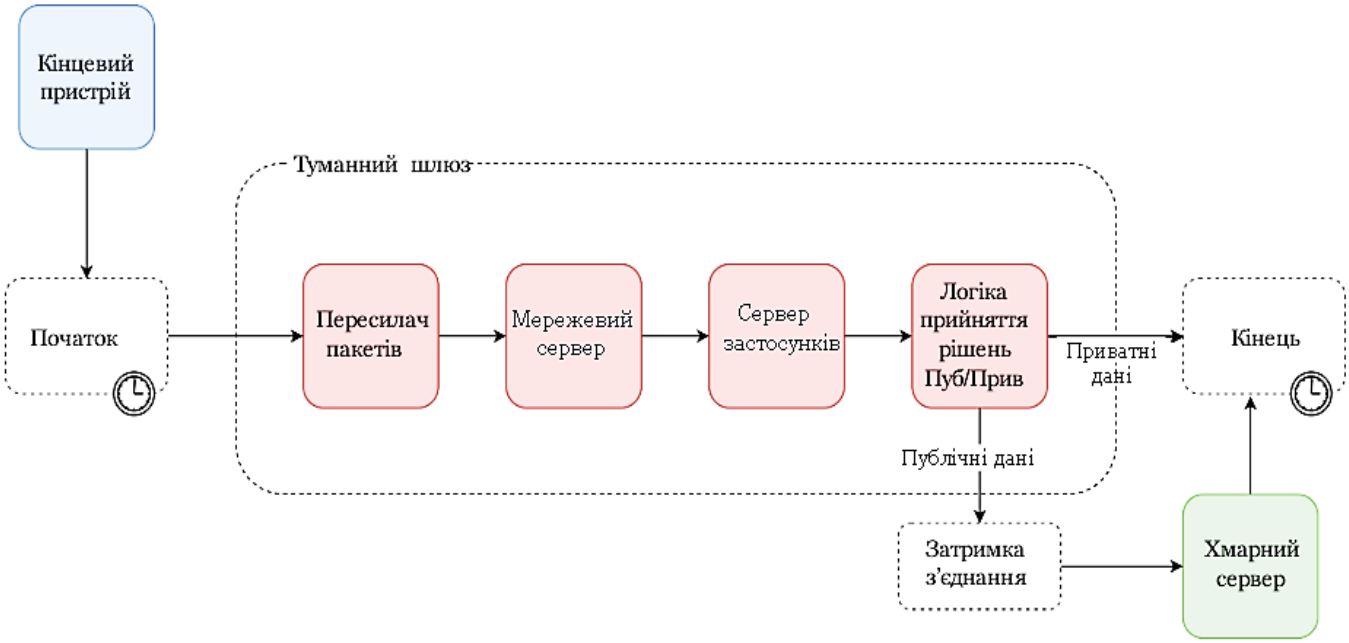


Рисунок 4.1 – Діаграма імітаційної моделі Архітектури А

Оскільки Архітектура В є лише варіацією Архітектури А, у якій мережеві та прикладні сервери не були реалізовані на кожному шлюзі, а лише на головному шлюзі, не було необхідності проводити окреме моделювання для цієї архітектури. Різниця у часах виконання полягала лише у затримці на лінії, яка очікувано

становила 0.5 мс під час передачі між підлеглим і головним шлюзом, що не потрібна в Архітектурі А.

Для запропонованої Архітектури С першим блоком, включеним у моделювання, був блок із n-серверами, який представляв форвардер пакетів. Середній час обслуговування цього блоку було встановлено на рівні 0.16 мс, а кількість ліній – 8. Потім дані оцінювалися на предмет того, чи є вони приватними чи публічними – рисунок 4.2.

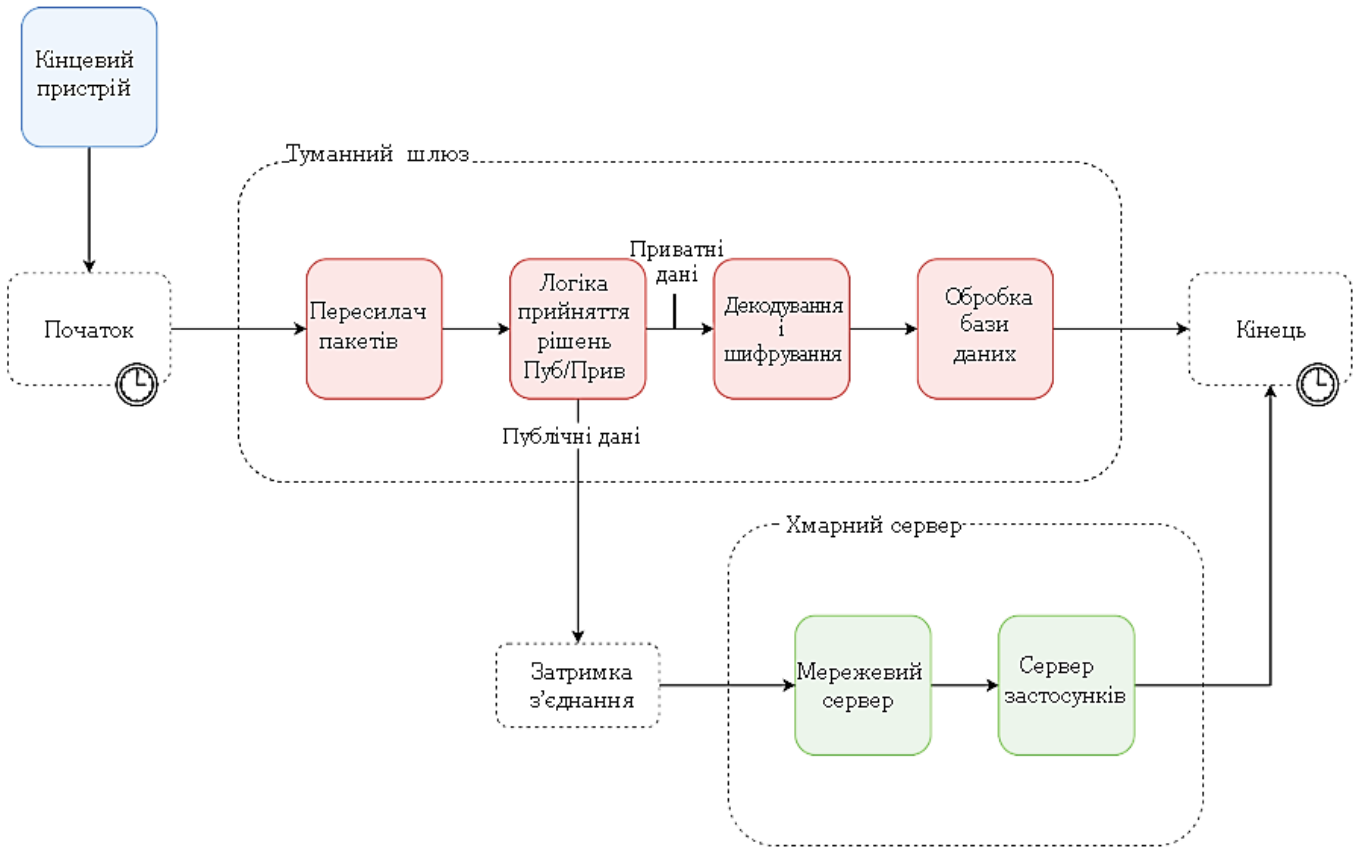


Рисунок 4.2 – Діаграма імітаційної моделі Архітектури С

У випадку приватних даних, запит на обробку перенаправлявся до ланцюга блоків із нескінченною кількістю серверів, відповідальних за декодування, розшифрування та збереження в локальну базу даних. Ці блоки використовували середні часи обслуговування відповідно до T_{decode}, $T_{decrypt}$ та $T_{database}$. На цьому

етапі обробка повідомлення завершувалася у випадку приватних даних. Загальний час виконання зберігався, і цикл моделювання завершувався.

У випадку передачі публічних даних, повідомлення пересилалося до хмари із затримкою на лінії в 0.5 мс. Хмара складалася з мережевого та прикладного серверів із середніми часами обслуговування 194.62 мс та 15.16 мс відповідно. Після завершення обробки цими системами черг загальний час виконання зберігався, і цикл моделювання завершувався.

Порівняння загального часу обслуговування, отриманого в результаті моделювання при 5% ймовірності надсилання публічних повідомлень, наведено на рисунку 16. Графік показує, що Архітектура С має значно менший загальний час виконання повідомлень – середній час обслуговування становить 19.44 мс, у порівнянні з 209.96 мс для Архітектури А. Звісно, це стосується лише ситуацій, коли переважають приватні дані, а основна обробка здійснюється шлюзом туманних обчислень – що і є принципом туманних обчислень та суттю запропонованого рішення в цьому дослідженні.

В іншому випадку, коли приватні дані не домінують, часи виконання для Архітектур А та С майже однакові.

Також можна спостерігати значні стрибки у часі виконання в Архітектурі С, що свідчить про передачу повідомлень до публічного хмарного сервера, де вони надалі обробляються. Якщо ж моделювання обмежити лише обробкою на шлюзі туманних обчислень, таких стрибків не спостерігається – рисунок 4.3 і 4.4.

Симуляція без передачі до хмарного сервера представлена на рисунку 4.1. У цьому випадку середній час обслуговування становив 10.28 мс в архітектурі С у порівнянні з 209.81 мс в архітектурі А. Це є найкращий можливий час виконання (BCET), оскільки мова йде про найкоротший час виконання для будь-якої можливої комбінації вхідних даних. Графік демонструє, що BCET для архітектури С був нижчим завдяки відсутності значних стрибків і тому, що дані не потребували обробки мережевим і прикладним серверами.

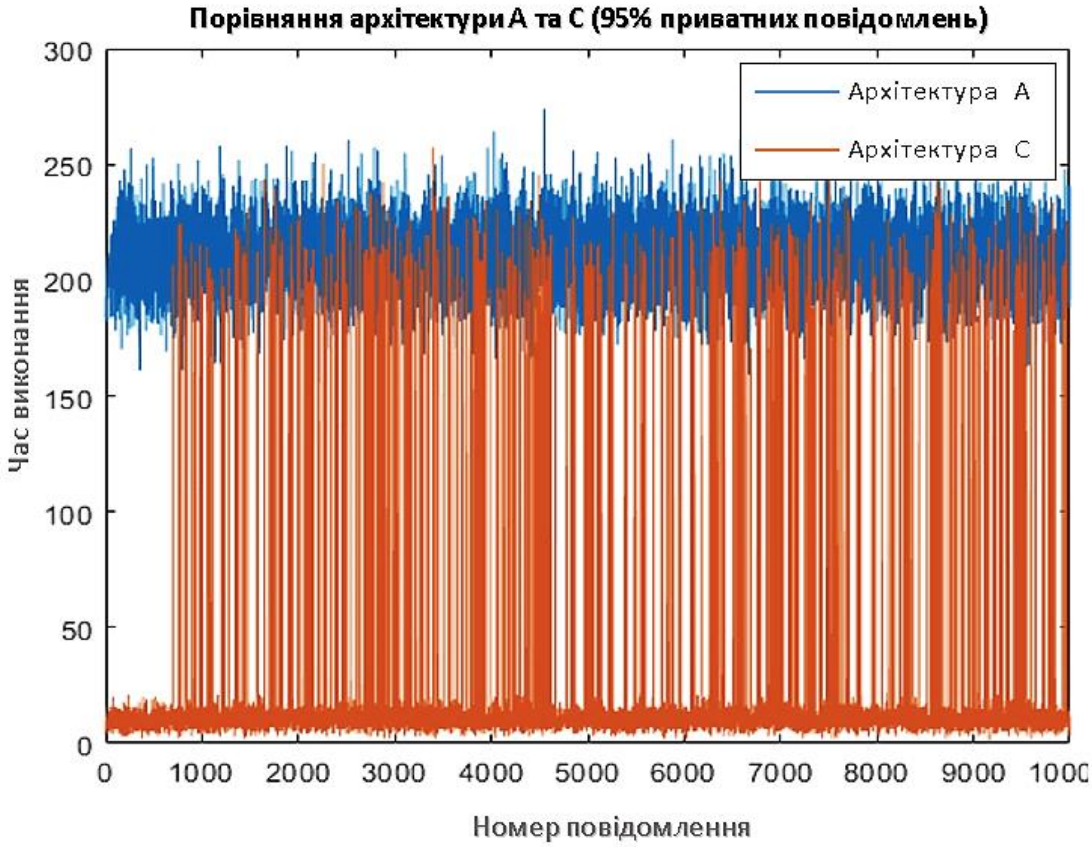


Рисунок 4.3 – Порівняння архітектур A та C (95% приватних повідомлень)

У випадку архітектури A, WCET був подібним до ситуації з 5% ймовірністю публічних повідомлень. Така подібність зумовлена наявністю мережевого та прикладного серверів на кожному шлюзі туманних обчислень – таким чином основна обробка все одно виконувалася локально, а хмарний сервер лише вставляв дані до бази.

Найгірший можливий час виконання (WCET) спостерігався у випадку, коли всі дані були публічними, і, відповідно, обчислення на шлюзі не виконувалися. У такому випадку WCET для архітектури C становив 210.41 мс, що подібно до WCET для архітектури A – 217.78 мс.

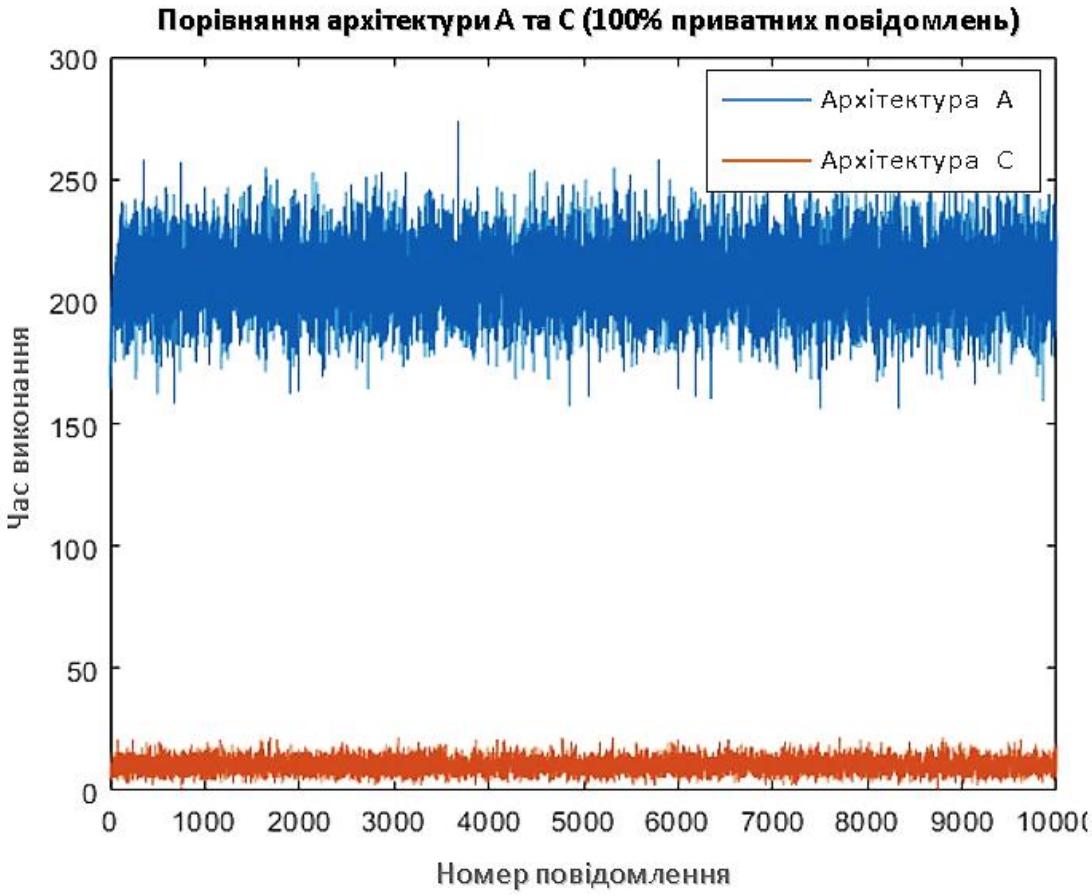


Рисунок 4.4 – Порівняння архітектур A та C (100% приватних повідомлень)

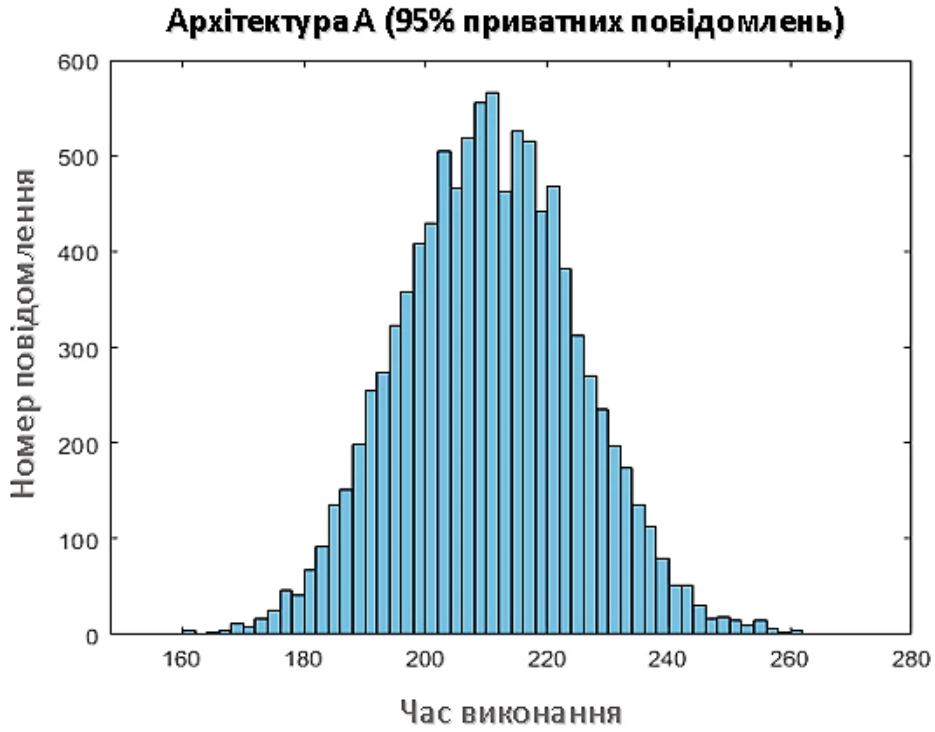


Рисунок 4.5 – Гістограма для архітектури A (95% приватних повідомлень)

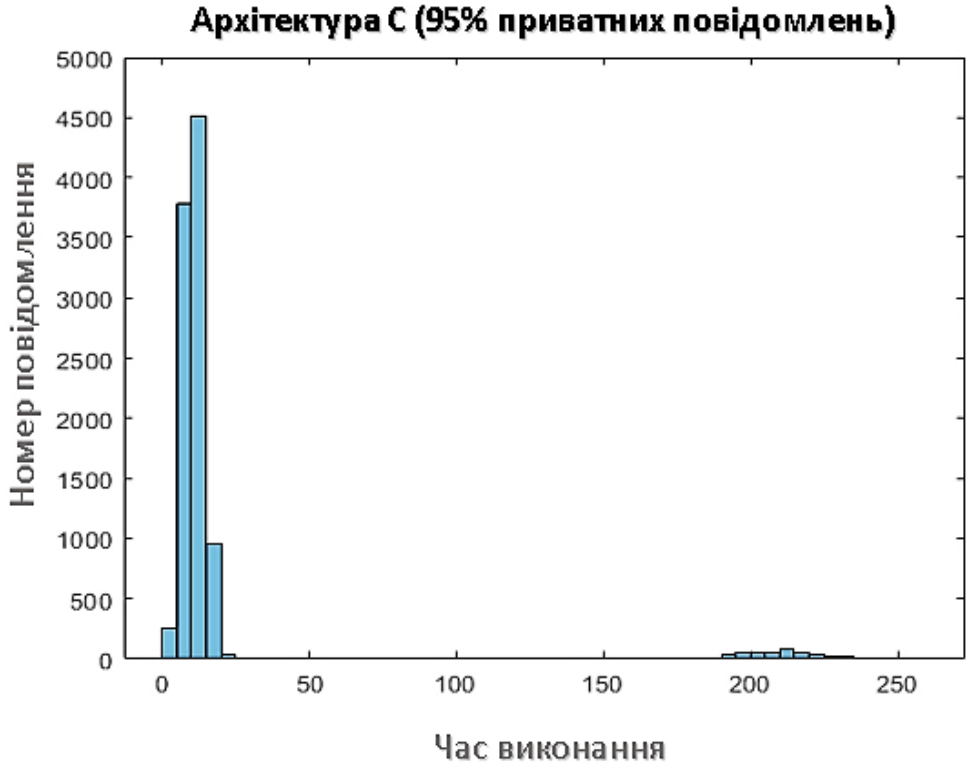


Рисунок 4.6 – Гістограма для архітектури C (95% приватних повідомлень)

Рисунки 4.5 та 4.6 демонструють відповідні гістограми для архітектур A та C при 95% ймовірності приватних повідомлень. Порівняння цих гістограм показує, що значення загального часу обслуговування для архітектури A були менш розсіяними, тоді як гістограма для архітектури C була значно ширшою, і дисперсія значень – вищою. Це пов’язано з уже згаданими відмінностями в обробці приватних та публічних повідомлень.

Рисунки 4.7 та 4.8 демонструють гістограми для симуляцій архітектур A та C при 100% приватних повідомлень. Ці гістограми свідчать, що розподіл гістограми для архітектури A майже не змінився в обох випадках, тоді як для архітектури C дисперсія значень при 95% приватних повідомлень була значно вищою, ніж у випадку обробки лише приватних даних.

Для апроксимації отриманого часу обслуговування для будь-якого апаратного рішення, можна вибрати базовий час T, що відповідає BCET архітектури C (10 мс). Тоді отримані значення часу можна масштабувати відповідно до цього базового рівня – таблиця 4.3.

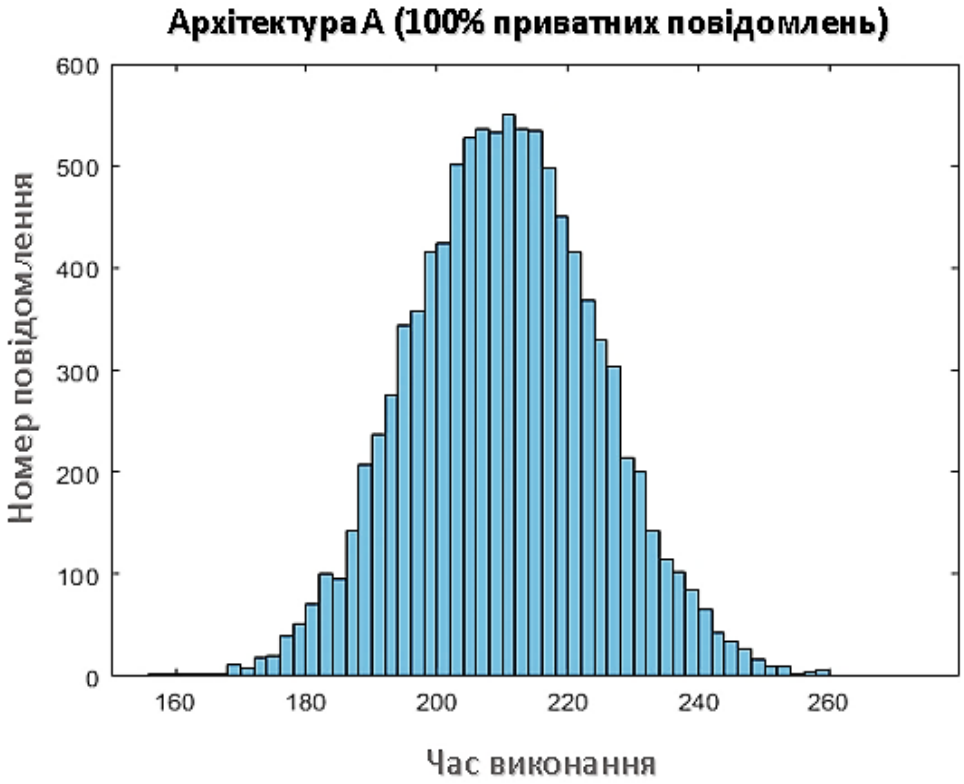


Рисунок 4.7 – Гістограма для архітектури А (100% приватних повідомлень)

Таблиця 4.3 – Порівняння результатів симуляції

Ознака	Архітектура А	Архітектура В	Архітектура С
Середній час виконання	$21 \cdot T$	$21 \cdot T + T_{\text{link}}$	$2 \cdot T$
BCET	$21 \cdot T$	$21 \cdot T + T_{\text{link}}$	T
WCET	$22 \cdot T$	$22 \cdot T + T_{\text{link}}$	$21 \cdot T$

Оскільки кожна із запропонованих архітектур по-різному вирішує задачу оптимізації, важливо порівняти їх з погляду функціональних характеристик. Ці властивості впливають на потенційні сценарії застосування архітектур та можливість інтеграції туманного шлюзу у вже існуючу інфраструктуру.

Архітектура А вирізняється інтеграцією мережевого та прикладного серверів у кожен туманний шлюз. Такий підхід дозволяє уникнути необхідності пересилання кожного повідомлення від кінцевого пристрою до віддалених мережевих і прикладних серверів. Таким чином можна запобігти проблемам із

недоступністю підключення. Оскільки не кожне повідомлення передається до публічного хмарного сервера, зберігається приватність переданих даних, що дещо знижує час обробки повідомлення.

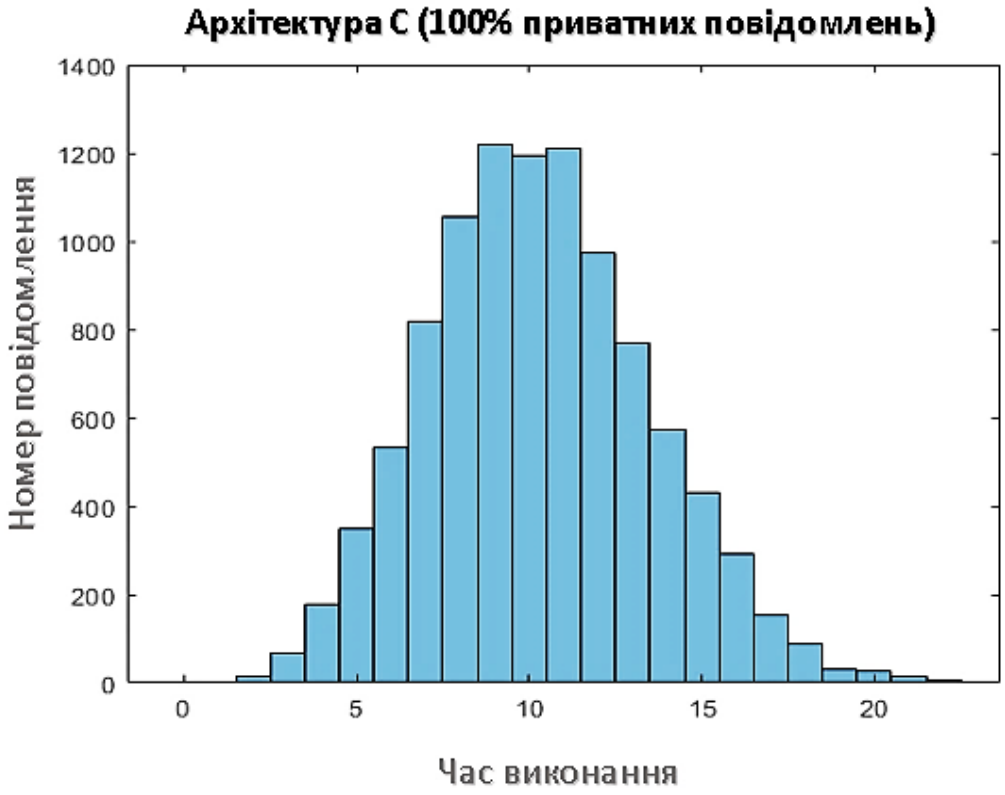


Рисунок 4.8 – Гістограма для архітектури C (100% приватних повідомлень)

Архітектура B реалізує підхід “майстер/підлеглий”. Підлеглі шлюзи працюють як звичайні LoRa-шлюзи й лише передають дані до одного головного шлюзу, який включає мережевий і прикладний сервери. Такий підхід доцільний у випадках, коли не всі шлюзи мають підключення до Інтернету. Шлюз без доступу до Інтернету надсилає повідомлення лише до головного шлюзу. Головний шлюз визначає, чи є повідомлення приватним чи публічним. Якщо повідомлення приватне, його зберігають у локальній базі даних. Якщо ні – передають до хмарного сервера. Це, по суті, модифікована версія архітектури A, яка має перевагу в тому, що сервери не потрібно розміщувати у кожному шлюзі. Синхронізація між шлюзами не потрібна, але усі підлеглі шлюзи мають бути з’єднані з головним.

На відміну від попередніх, архітектура С не потребує постійного підключення до мережевих і прикладних серверів для належного функціонування. Якщо туманний шлюз має локально збережені адреси кінцевих пристроїв і їхні сесійні ключі, він може повністю самостійно обробляти всі повідомлення. Отже, шлюз може працювати автономно, без участі віддалених серверів.

Важливою перевагою є легка інтеграція такого шлюзу в стандартну архітектуру LoRaWAN, особливо якщо достатньо одного шлюзу для покриття всієї території. Якщо туманний шлюз приймає приватне повідомлення, він одразу починає обробку.

Однак, якщо декілька шлюзів отримають однакове повідомлення, кожен із них окремо його обробить, що призведе до дублювання збережених даних. Такий підхід недоцільний, тому слід уникати перекриття зон покриття або реалізувати механізм координації між шлюзами.

Один із можливих варіантів – впровадити сигнальний механізм через протокол MQTT: шлюз, що першим прийняв повідомлення, надсилає сповіщення іншим шлюзам, які відмовляються від обробки. У такому випадку розшифровані дані кінцевого пристрою можуть зберігатися у різних базах даних, залежно від того, який саме шлюз обробив конкретне повідомлення.

Альтернативний підхід – призначити кожен туманний шлюз для обробки повідомлень від певних кінцевих пристроїв. Якщо шлюз отримує повідомлення від пристрою, за який він відповідає, – він його обробляє. Інакше – припиняє обробку. Цей підхід дозволяє уникнути фрагментації даних одного пристрою між базами даних різних шлюзів.

Виконане порівняння дозволило обрати оптимальну архітектуру серед трьох запропонованих.

Середній час виконання при ймовірності 95% приватних повідомлень був найнижчим у запропонованій архітектурі С – 19.44 мс. Такий час досягнуто завдяки відсутності обробки на мережевих і прикладних серверах у разі обробки на рівні туману. Ще кращим був час у випадку найкоротшого можливого сценарію обробки (BCET) – лише 10.28 мс. У той час як у архітектурах А та В, де мережеві й

прикладні сервери інтегровані в кожен туманний шлюз, часи обробки були значно вищими.

Оскільки оцінка приватності даних у архітектурі C відбувається до розшифрування повідомлення, шлюз може передати отримане повідомлення до публічного хмарного сервера без змін. Це забезпечує сумісність з типовою архітектурою LoRaWAN.

В архітектурах A та B ситуація інша: оскільки кожен шлюз містить мережевий та прикладний сервер, повідомлення проходить усі етапи обробки ще до надсилання. Таким чином, до хмарного сервера передається вже розшифроване повідомлення, що робить такі шлюзи несумісними з типовою інфраструктурою LoRaWAN.

Таблиця 4.4 – Підсумок характеристик запропонованих архітектур

Характеристика	Архітектура A	Архітектура B	Архітектура C
Середній час виконання	209.96 мс	Подібний до арх. A	19.44 мс
BCET (найкращий сценарій)	209.81 мс	Подібний до арх. A	10.28 мс
WCET (найгірший сценарій)	217.78 мс	Подібний до арх. A	210.41 мс
Можливість впровадження в LoRaWAN	Ні	Ні	Так
Автономна робота	Так	Так	Так
Обчислювальні вимоги	Високі	Високі	Низькі

Варто зазначити, що в усіх запропонованих архітектурах туманні шлюзи можуть працювати автономно, якщо зв’язок з хмарним сервером втрачено.

Очевидно також, що інтеграція серверів у туманний шлюз значно підвищує обчислювальні вимоги до нього.

Підсумовуючи основні характеристики кожної з архітектур, видно, що найкращі результати показала архітектура С – таблиця 4.4. Тому для подальшої реалізації прототипу було обрано архітектуру С.

4.2 Реалізація прототипу та експериментальне тестування

Для реалізації було вирішено використати програмне забеспечення ChirpStack – відкрите програмне забезпечення для реалізації стека мережі LoRaWAN, яке забезпечує гнучкість у створенні приватних LoRaWAN-мереж без залежності від загальної інфраструктури. У цій роботі ChirpStack використовується для побудови базової інфраструктури LoRaWAN.

Основними компонентами ChirpStack є ChirpStack Gateway Bridge, ChirpStack Network Server та ChirpStack Application Server.

ChirpStack Gateway Bridge – служба, що працює між пакетним ретранслятором шлюзу LoRaWAN та сервером MQTT. Вона перетворює отримані пакети LoRa у формат JSON і передає їх через MQTT.

ChirpStack Network Server – компонент, який відповідає за усунення дублюючих кадрів, автентифікацію отриманих кадрів, комунікацію з сервером додатків та планування передачі даних назад.

ChirpStack Application Server керує пристроями та виконує шифрування/дешифрування навантаження. Також має веб-додаток для управління пристроями та додатками через API.

Заголовок MHDR містить важливу інформацію, яка вказує на тип повідомлення (MType) – таблиця 4.5. У контексті цієї роботи особливу важливість мають типи Unconfirmed/Confirmed Data Up, які передають прикладні дані.

Оскільки архітектура С є модифікацією стандартної LoRaWAN-архітектури, де основні зміни стосуються fog-шлюзу, який забезпечує дешифрування

навантаження, то для дешифрування на шлюзі необхідно знати сесійні ключі – рисунок 4.7. Основні кроки реалізації:

1. Шлюз отримує DevAddr з кожного повідомлення LoRaWAN.
2. Якщо сесійні ключі вже збережено в локальній базі даних, шлюз використовує їх для дешифрування.
3. Якщо ключів немає, шлюз звертається до сервера додатків за допомогою API для отримання DevEUI і сесійних ключів.
4. Структура повідомлення LoRaWAN включає PHYPayload, який містить MHDR (заголовок повідомлення) та MIC (код перевірки цілісності повідомлення).

Шифрування даних здійснюється за допомогою AES-128 з блоковим шифруванням.

Алгоритм шифрування:

1. Ai – блоки даних, які шифруються.
2. Si = aes128_encrypt(Key, Ai) для кожного блоку.

Після шифрування, результат застосовується до корисного навантаження, що дозволяє його дешифрувати.

Оскільки стандартні шлюзи LoRaWAN приймають повідомлення від усіх кінцевих пристроїв, що перебувають у зоні покриття, а перевірка на дублювання виконується лише на мережевому сервері, може трапитися, що ідентифікатор DevAddr у поточному повідомленні не належить до цієї LoRaWAN-мережі. У такому випадку шлюз туманных обчислень (fog gateway) може проігнорувати таке повідомлення.

Щоб уникнути повторюваних запитів до серверу застосунків у випадках, коли кінцевий пристрій періодично надсилає повідомлення, але не є частиною цієї LoRaWAN-мережі, DevAddr такого пристрою додається до локальної бази даних шлюзу. Для нього AppSKey та NwkSKey встановлюються у значення нуль. Таким чином, fog gateway одразу розпізнає, що це пристрій, зареєстрований в іншій LoRaWAN-мережі, і припиняє обробку повідомлення. Повідомлення не надсилається на мережевий сервер, що дозволяє уникнути зайвого навантаження.

Поточна архітектура мережі на кампусі VSB-TUO включає кінцеві пристрої, шлюзи, сервери The Things Network (TTN), сервер KoIoT і веб-додаток для користувачів. Шлюзи LoRaWAN виступають містком між бездротовими і проводовими компонентами мережі. Кожен шлюз підключений до Інтернету через Ethernet для забезпечення з'єднання з бекендом TTN. TTN виконує функції маршрутизації IoT даних між кінцевими пристроями та додатками. Інша важлива частина інфраструктури - сервер KoIoT, який управляється Департаментом телекомунікацій і використовується для зберігання даних та хостингу веб-інтерфейсу для кінцевих користувачів.

Таблиця 4.5 – Типи MAC повідомлень

MType	Опис
00	Запит на підключення
01	Прийняття підключення
010	Непідтверджене повідомлення (Data Up)
011	Непідтверджене повідомлення (Data Down)
100	Підтверджене повідомлення (Data Up)
101	Підтверджене повідомлення (Data Down)
110	Резервовано для майбутнього використання
111	Власні повідомлення

Для порівняння часу обробки між оптимізованою архітектурою та іншими двома рішеннями – поточною архітектурою на кампусі і мережею на базі ChirpStack – було виміряно загальний час обробки повідомлення, від його надходження на шлюз до зберігання розшифрованого payload у базі даних.

В оптимізованій архітектурі обробка payload відбувається на fog gateway, що значно знижує час обробки.

Для проведення тестування були використані два кінцеві пристрої, які надсилали повідомлення кожні 10 секунд. Один пристрій оброблявся через fog gateway, а інший – через хмару.

Середній час обробки в оптимізованій архітектурі становив 121.74 мс, коли деякі повідомлення оброблялися в хмарі.

При обробці усіх повідомлень на fog gateway середній час становив 10.44 мс.

Для мереж на базі ChirpStack середній час обробки становив 231.92 мс, а для поточної мережі на кампусі з TTN – 297.93 мс.

У порівнянні з поточною мережею на кампусі оптимізована архітектура дозволяє зменшити час обробки більш ніж удвічі.

Мета тесту – перевірити захист конфіденційності даних. Якщо повідомлення від кінцевого пристрою позначено як приватне, fog gateway обробляє та зберігає його в локальній базі даних, тоді як непри privadosічні повідомлення пересилаються в хмару. Тест включав два пристрої, один з яких відправляв приватні повідомлення, а інший – публічні.

Це показує, що система працює коректно – приватні повідомлення зберігаються в локальній базі даних fog gateway, а публічні – в хмарі.

Туманна обчислювальна парадигма дозволяє обробку повідомлень без постійного з'єднання з хмарою. Для перевірки цієї функції проводився тест з навмисним відключенням з'єднання між fog gateway і хмарою. Коли з'єднання було відновлено, обробка повідомлень повернулася в хмару.

Таблиця 4.6 – Результати тесту конфіденційності.

Місце зберігання БД	Приватний кінцевий пристрій	Публічний кінцевий пристрій
Fog gateway	99	0
Cloud	1	100
Total	100	100

Після відключення з'єднання всі повідомлення оброблялися локально на fog gateway.

Як тільки з'єднання з хмарою було відновлено, обробка повідомлень перенеслася назад на хмару.

Тест порівняння часу обробки підтвердив, що оптимізована архітектура забезпечує значно нижчу затримку в обробці повідомлень, оскільки fog gateway може обробляти повідомлення локально – рисунок 4.9.

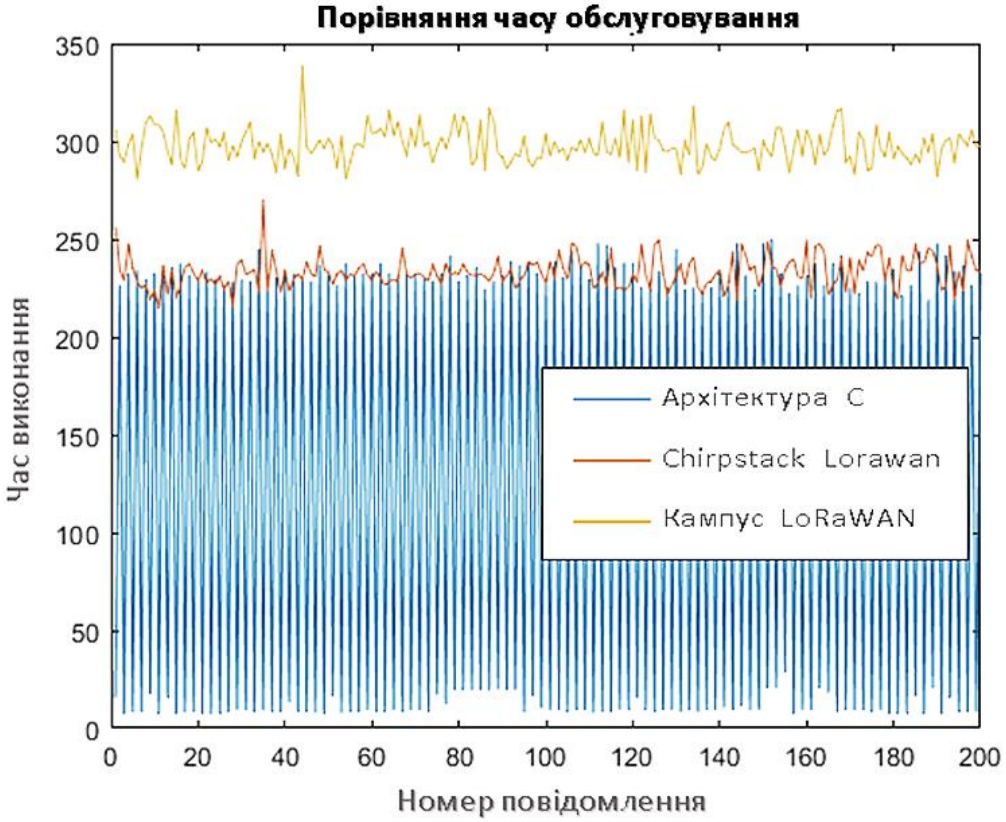


Рисунок 4.9 – Результати порівняння часу обслуговування

Тест конфіденційності підтвердив, що приватні повідомлення обробляються і зберігаються локально на fog gateway, а непри приватні – відправляються в хмару.

Тест оффлайн-обробки показав, що fog gateway здатен працювати автономно при відсутності з'єднання з хмарою, зберігаючи можливість обробки і зберігання повідомлень навіть в оффлайн-режимі.

Загалом, тестування підтвердило ефективність та надійність запропонованої архітектури з fog computing для зменшення затримок, збереження конфіденційності даних і підтримки роботи в оффлайн-режимі.

4.3 Висновки до четвертого розділу

Проведений аналіз архітектур показав, що запропонована оптимізована IoT-архітектура з використанням туманних обчислень здатна значно зменшити час обробки повідомлень у порівнянні зі стандартними рішеннями на базі TTN та ChirpStack. Завдяки можливості локальної обробки даних на шлюзах, система демонструє покращену масштабованість, гнучкість та здатність до обробки приватних повідомлень без залучення хмарної інфраструктури.

Порівняльний аналіз виявив ключові переваги туманної архітектури у контексті часу обробки, розподілу навантаження та захисту приватності. Таким чином, запропоновані архітектурні рішення довели свою ефективність та практичну доцільність для впровадження у сучасних IoT-системах, що функціонують у гетерогенних і динамічних середовищах.

Експериментальна реалізація та тестування прототипу підтвердили функціональність і ефективність запропонованої архітектури. Результати порівняння часу обробки показали, що локальна обробка на шлюзі дозволяє зменшити середній час обробки повідомлень у понад два рази порівняно з хмарними рішеннями. Зокрема, в умовах виключно туманної обробки досягнуто середнього часу 10.44 мс, що значно перевищує показники TTN (297.93 мс) та ChirpStack (231.92 мс).

Тест приватності підтвердив здатність системи до коректної обробки приватних повідомлень, забезпечуючи збереження даних лише на локальному рівні. Крім того, перевірка офлайн-режиму показала, що при втраті з’єднання з хмарою система автоматично переходить у локальний режим, зберігаючи безперервність обробки. Це підтверджує здатність архітектури до автономного функціонування, що є ключовою вимогою до туманних IoT-систем. Отримані

результати свідчать про високу надійність запропонованого підходу та його придатність до використання у реальних сценаріях.

ВИСНОВОК

У результаті виконання роботи було розроблено, обґрунтовано та експериментально досліджено архітектурні рішення для IoT-систем із підтримкою туманних обчислень, що враховують контроль передачі даних, зниження затримок та забезпечення приватності інформації. Запропонований підхід дозволяє ефективно розподіляти обчислювальні навантаження між туманними вузлами і хмарною інфраструктурою, адаптуючись до стану мережі, обсягу трафіку та обмежень ресурсів.

Було розроблено архітектуру з можливістю гнучкого вибору обробки даних (туман/хмара) та механізмами динамічного реагування на статус з’єднання і політику конфіденційності. Основною інновацією є інтеграція функціоналу локального зберігання, попередньої обробки повідомлень та автоматичного перемикання в офлайн-режим, що забезпечує автономність IoT-вузлів у критичних ситуаціях.

Алгоритмічна реалізація включала створення прототипу на основі Raspberry Pi 4 та модуля iC880A, інтегрованих із системами TTN та ChirpStack. Було розроблено веб-інтерфейс для вибору режиму обробки, а також реалізовано функціонал обліку сесій, зберігання ключів і контролю конфіденційності повідомлень.

У процесі експериментального тестування підтверджено такі ключові властивості системи:

- зменшення часу обробки повідомлень у понад 20 разів у режимі локальної обробки (до 10.44 мс проти 297.93 мс у TTN);
- коректне функціонування механізму приватності з повним збереженням приватних даних на шлюзі;
- автоматичне перемикання у режим офлайн з локальним зберіганням даних при втраті з’єднання з хмарою;
- сталість часу обробки у мережах ChirpStack та TTN підтверджує об'єктивність результатів;

– забезпечення масштабованості та стабільності при змінних умовах мережі.

Отримані результати підтвердили гіпотези, сформульовані в теоретичних розділах, та довели доцільність використання туманних обчислень у сучасних IoT-мережах. Запропонований підхід має практичне застосування у розумних містах, промисловому моніторингу, системах критичної інфраструктури та сценаріях з обмеженим доступом до мережі.

Перспективи подальших досліджень включають інтеграцію енергозберігаючих алгоритмів, розширення підтримки багаторівневої конфіденційності та дослідження безпеки при зловмисних атаках і мережевих збоях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Habibi P., Farhoudi M., Kazemian S., Khorsandi S., Leon-Garcia A. Fog computing: a comprehensive architectural survey. *IEEE access*. 2020. Vol. *8*. P. 69105-69133.

2. Puliafito C., Mingozzi E., Longo F., Puliafito, A., Rana O. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology (TOIT)*. 2019. Vol. 19(2), P. 1-41.

3. Costa B., Bachiega Jr. J., De Carvalho L. R., Araujo A. P. Orchestration in fog computing: a comprehensive survey. *ACM Computing Surveys (CSUR)*. 2022. Vol. 55(2). P. 1–34.

4. Das R., Inuwa M. M. A review on fog computing: issues, characteristics, challenges, and potential applications. *Telematics and Informatics Reports*. 2023. Vol. 10. P. 100049.

5. Mahmud R., Ramamohanarao K., Buyya R. Application management in fog computing environments: a taxonomy, review and future directions. *ACM Computing Surveys (CSUR)*. 2020. Vol. 53(4). P. 1–43.

6. Abdulkareem K. H., Mohammed M. A., Gunasekaran S. S., Al-Mhiqani M. N., Mutlag A. A., Mostafa S. A., Ibrahim D. A. A review of fog computing and machine learning: concepts, applications, challenges, and open issues. *IEEE Access*. 2019. Vol. 7. P. 153123–153140.

7. Laghari A. A., Jumani A. K., Laghari R. A. Review and state of art of fog computing. *Archives of Computational Methods in Engineering*. 2021. Vol. 28(5). P. 3631–3643

8. Sabireen H., Neelanarayanan V. J. A review on fog computing: architecture, fog with IoT, algorithms and research challenges. *ICT Express*. 2021. Vol. 7(2). P. 162–176.

9. Ren J., Zhang D., He S., Zhang Y., Li T. A survey on end-edge-cloud orchestrated network computing paradigms: transparent computing, mobile edge

computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)*. 2019. Vol. 52(6). P. 1–36.

10. Keshari N., Singh D., Maurya A. K. A survey on vehicular fog computing: current state-of-the-art and future directions. *Vehicular Communications*. 2022. Vol. 38. P. 100512.

11. Martinez I., Hafid A. S., Jarray A. Design, resource management, and evaluation of fog computing systems: a survey. *IEEE Internet of Things Journal*. 2020. Vol. 8(4). P. 2494–2516.

12. Javadzadeh G., Rahmani A. M. Fog computing applications in smart cities: a systematic survey. *Wireless Networks*. 2020. Vol. 26(2). P. 1433–1457.

13. Ometov A., Molua O. L., Komarov M., Nurmi J. A survey of security in cloud, edge, and fog computing. *Sensors*. 2022. Vol. 22(3). P. 927.

14. Tange K., De Donno M., Fafoutis X., Dragoni N. A systematic survey of industrial internet of things security: requirements and fog computing opportunities. *IEEE Communications Surveys & Tutorials*. 2020. Vol. 22(4). P. 2489–2520.

15. Kumari N., Yadav A., Jana P. K. Task offloading in fog computing: a survey of algorithms and optimization techniques. *Computer Networks*. 2022. Vol. 214. P. 109137.

16. Srirama S. N. A decade of research in fog computing: relevance, challenges, and future directions. *Software: Practice and Experience*. 2024. Vol. 54(1). P. 3–23.

17. Alwakeel A. M. An overview of fog computing and edge computing security and privacy issues. *Sensors*. 2021. Vol. 21(24). P. 8226.

18. Markus A., Kertesz A. A survey and taxonomy of simulation environments modelling fog computing. *Simulation Modelling Practice and Theory*. 2020. Vol. 101. P. 102042.

19. Hazra A., Rana P., Adhikari M., Amgoth T. Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges. *Computer Science Review*. 2023. Vol. 48. P. 100549.

20. Shewring N. IoT Made Simple: An Easy-to-Understand Guide to IoT Architecture for Beginners. 2024. 135 p.

21. Rossman J. The Amazon Way on IoT: 10 Principles for Every Leader from the World's Leading Internet of Things Strategies. 2021. 170 p.

22. Jasbir Singh Dhaliwal. Architectural Patterns and Techniques for Developing IoT Solutions: Build IoT applications using digital twins, gateways, rule engines, AI/ML integration, and related patterns. 2023. 153 p.

23. Perry L. IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security. 2020. 128 p.

24. Jamil B., Ijaz H., Shojafar M., Munir K., Buyya R. Resource allocation and task scheduling in fog computing and internet of everything environments: a taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*. 2022. Vol. 54(11s). P. 1–38.

25. Nayeri Z. M., Ghafarian T., Javadi B. Application placement in fog computing with AI approach: taxonomy and a state of the art survey. *Journal of Network and Computer Applications*. 2021. Vol. 185. P. 103078.

26. Goudarzi M., Palaniswami M., Buyya R. Scheduling IoT applications in edge and fog computing environments: a taxonomy and future directions. *ACM Computing Surveys*. 2022. Vol. 55(7). P. 1–41.

27. Islam M. S. U., Kumar A., Hu Y. C. Context-aware scheduling in fog computing: a survey, taxonomy, challenges and future directions. *Journal of Network and Computer Applications*. 2021. Vol. 180. P. 103008.

28. Yang X., Rahmani N. Task scheduling mechanisms in fog computing: review, trends, and perspectives. *Kybernetes*. 2021. Vol. 50(1). P. 22–38.

29. Behravan K., Farzaneh N., Jahanshahi M., Seno S. A. H. A comprehensive survey on using fog computing in vehicular networks. *Vehicular Communications*. 2023. Vol. 42. P. 100604.

30. Samann F. E., Abdulazeez A. M. Askar S. Fog computing based on machine learning: a review. *International Journal of Interactive Mobile Technologies*. 2021. Vol. 15(12).

31. Raza M. R., Varol A., Varol N. Cloud and fog computing: a survey to the concept and challenges. *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*. 2020. P. 1–6. IEEE.

32. Tmamna J., Ayed E. B., Ayed M. B. Deep learning for internet of things in fog computing: survey and open issues. *2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. 2020. P. 1–6. IEEE.

33. Matrouk K., Alatoun K. Scheduling algorithms in fog computing: a survey. *International Journal of Networked and Distributed Computing*. 2021. Vol. 9(1). P. 59–74.

34. Ghobaei-Arani M., Souri A., Rahmanian A. A. Resource management approaches in fog computing: a comprehensive review. *Journal of Grid Computing*. 2020. Vol. 18(1). P. 1–42.

35. Jumani A. K., Shi J., Laghari A. A., Hu Z., Nabi A. U. Qian H. Fog computing security: a review. *Security and Privacy*. 2023. Vol. 6(6). P. 313.

36. Kashani M. H., Mahdipour E. Load balancing algorithms in fog computing. *IEEE Transactions on Services Computing*. 2022. Vol. 16(2). P. 1505–1521.

37. Laroui M., Nour B., Moungla H., Cherif M. A., Afifi H., Guizani M. Edge and fog computing for IoT: a survey on current research activities & future directions. *Computer Communications*. 2021. Vol. 180. P. 210–231.

38. Gasmi K., Dilek S., Tosun S., Ozdemir S. A survey on computation offloading and service placement in fog computing-based IoT. *The Journal of Supercomputing*. 2022. Vol. 78(2). P. 1983–2014.

39. Hosseinzadeh M., Azhir E., Lansky J., Mildeova S., Ahmed O. H., Malik M. H., Khan F. Task scheduling mechanisms for fog computing: a systematic survey. *IEEE Access*. 2023. Vol. 11. P. 50994–51017.

40. Basir R., Qaisar S., Ali M., Aldwairi M., Ashraf M. I., Mahmood A., Gidlund M. Fog computing enabling industrial internet of things: state-of-the-art and research challenges. *Sensors*. 2019. Vol. 19(21). P. 4807.

41. Belmahdi R., Mechta D., Harous S. A survey on various methods and algorithms of scheduling in fog computing. *Ingénierie des Systèmes d'Information*. 2021. Vol. 26(2). P. 211–224.

42. Caiza G., Saeteros M., Oñate W., Garcia M. V. fog computing at industrial level, architecture, latency, energy, and security: a review. *Heliyon*. 2020. Vol. 6(4).

43. Margariti S. V., Dimakopoulos V. V., Tsoumanis G. Modeling and simulation tools for fog computing – a comprehensive survey from a cost perspective. *Future Internet*. 2020. Vol. 12(5). P. 89.

44. Almobaideen W., Altarawneh M. Fog computing: survey on decoy information technology. *International Journal of Security and Networks*. 2020. Vol. 15(2). P. 111–121.

45. Shakarami A., Shakarami H., Ghobaei-Arani M., Nikougoftar E., Faraji-Mehmandar M. Resource provisioning in edge/fog computing: a comprehensive and systematic review. *Journal of Systems Architecture*. 2022. Vol. 122. P. 102362.

46. Ebneyousef S., Shirmarz A. A taxonomy of load balancing algorithms and approaches in fog computing: a survey. *Cluster Computing*. 2023. Vol. 26(5). P. 3187–3208.

47. Alsadie D. A comprehensive review of AI techniques for resource management in fog computing: Trends, challenges and future directions. *IEEE Access*. 2024. P. 153.

48. Gomes E., Costa F., De Rolt C., Plentz P., Dantas M. A survey from real-time to near real-time applications in fog computing environments. *Telecom*. 2021. Vol. 2(4). P. 489–517.

49. Pallewatta S., Kostakos V., Buyya R. Placement of microservices-based IoT applications in fog computing: A taxonomy and future directions. *ACM Computing Surveys*. 2023. Vol. 55(14s). P. 1–43.

50. Alizadeh M. R., Khajehvand V., Rahmani A. M., Akbari E. Task scheduling approaches in fog computing: A systematic review. *International Journal of Communication Systems*. 2020. Vol. 33(16). P. 4583.

51. Nazih O., Benamar N., Lamaazi H., Chaoui H. Toward secure and trustworthy vehicular fog computing: A survey. *IEEE Access*. 2024. Vol. 12. P. 35154–35171.

52. Arivazhagan C., Natarajan V. A Survey on Fog computing paradigms, Challenges and Opportunities in IoT. *In 2020 International Conference on Communication and Signal Processing (ICCSP)*. 2020. P. 0385–0389.

53. Haghi Kashani M., Rahmani A. M., Jafari N. Quality of service-aware approaches in fog computing. *International Journal of Communication Systems*. 2020. Vol. 33(8). P. 4340.

54. Chuan W. C., Manickam S., Ashraf E., Karuppayah S. Challenges and opportunities in fog computing scheduling: A literature review. *IEEE Access*. 2025.

55. Wang H., Liu T., Kim B., Lin C. W., Shiraishi S., Xie J., Han Z. Architectural design alternatives based on cloud/edge/fog computing for connected vehicles. *IEEE Communications Surveys & Tutorials*. 2020. Vol. 22(4). P. 2349–2377.

56. Hosseinioun P., Kheirabadi M., Kamel Tabbakh S. R., Ghaemi R. Task scheduling approaches in fog computing: A survey. *Transactions on Emerging Telecommunications Technologies*. 2022. Vol. 33(3). P. 3792.

57. Fersi G. Fog computing and Internet of Things in one building block: A survey and an overview of interacting technologies. *Cluster Computing*. 2021. Vol. 24(4). P. 2757–2787.

58. Da Silva T. P., Batista T., Lopes F., Neto A. R., Delicato F. C., Pires P. F., Da Rocha A. R. Fog computing platforms for smart city applications: A survey. *ACM Transactions on Internet Technology*. 2022. Vol. 22(4). P. 1–32.

59. Apat H. K., Nayak R., Sahoo B. A comprehensive review on Internet of Things application placement in Fog computing environment. *Internet of Things*. 2023. Vol. 23. P. 100866.

60. Aleisa M. A., Abuhussein A., Sheldon F. T. Access control in fog computing: Challenges and research agenda. *IEEE Access*. 2020. Vol. 8. P. 83986–83999.

61. Pareek K., Tiwari P. K., Bhatnagar V. Fog computing in healthcare: A review. *In IOP Conference Series: Materials Science and Engineering*. 2021. Vol. 1099(1). P. 012025.

62. Tuli S., Mirhakimi F., Pallewatta S., Zawad S., Casale G., Javadi B. Jennings N. R. AI augmented Edge and Fog computing: Trends and challenges. *Journal of Network and Computer Applications*. 2023. Vol. 216. P. 103648.

63. Burhan M., Alam H., Arsalan A., Rehman R. A., Anwar M., Faheem M., Ashraf M. W. A comprehensive survey on the cooperation of fog computing paradigm-based IoT applications: Layered architecture, real-time security issues, and solutions. *IEEE Access*. 2023. Vol. 11. P. 73303–73329.

64. Songhorabadi M., Rahimi M., MoghadamFarid A., Kashani M. H. Fog computing approaches in IoT-enabled smart cities. *Journal of Network and Computer Applications*. 2023. Vol. 211. P. 103557.

65. Ahammad I. Fog computing complete review: Concepts, trends, architectures, technologies, simulators, security issues, applications, and open research fields. *SN Computer Science*. 2023. Vol. 4(6). P. 765.

66. Moura J., Hutchison D. Fog computing systems: State of the art, research issues and future trends, with a focus on resilience. *Journal of Network and Computer Applications*. 2020. Vol. 169. P. 102784.

67. Gupta A., Gupta S. K. A survey on green unmanned aerial vehicles-based fog computing: Challenges and future perspective. *Transactions on Emerging Telecommunications Technologies*. 2022. Vol. 33(11). P. 4603.

68. Khan S., Shah I. A., Ahmad S., Khan J. A., Anwar M. S., Aurangzeb K. A. Comprehensive Survey on Multi-Facet Fog-Computing Resource Management Techniques, Trends, Applications and Future Directions. *Expert Systems*. 2025. Vol. 42(4). P. 70019.

69. Ahmad S., Shakeel I., Mehfuz S., Ahmad J. Deep learning models for cloud, edge, fog, and IoT computing paradigms: Survey, recent advances, and future directions. *Computer Science Review*. 2023. Vol. 49. P. 100568.

70. Rani S., Kataria A., Chauhan M. Fog computing in industry 4.0: Applications and challenges – A research roadmap. *Energy conservation solutions for fog-edge computing paradigms*. 2022. P. 173–190.

71. Sicari S., Rizzardi A., Coen-Porisini A. Insights into security and privacy towards fog computing evolution. *Computers & Security*. 2022. Vol. 120. P. 102822.

72. Quy V. K., Hau N. V., Anh D. V., Ngoc L. A. Smart healthcare IoT applications based on fog computing: architecture, applications and challenges. *Complex & Intelligent Systems*. 2022. Vol. 8(5). P. 3805–3815.

73. De Moura Costa H. J., Da Costa C. A., Da Rosa Righi R., Antunes R. S. Fog computing in health: A systematic literature review. *Health and Technology*. 2020. Vol. 10(5). P. 1025–1044.

74. Sadri A. A., Rahmani A. M., Saberikamarposhti M., Hosseinzadeh M. Data reduction in fog computing and internet of things: A systematic literature survey. *Internet of Things*. 2022. Vol. 20. P. 100629.

75. Bouachir O., Aloqaily M., Tseng L., Boukerche A. Blockchain and fog computing for cyberphysical systems: The case of smart industry. *Computer*. 2020. Vol. 53(9). P. 36–45.

76. Ogundoyin S. O., Kamil I. A., Optimization techniques and applications in fog computing: An exhaustive survey. *Swarm and Evolutionary Computation*. 2021. Vol. 66. P. 100937.

77. Alzoubi Y. I., Al-Ahmad A., Kahtan H. Blockchain technology as a Fog computing security and privacy solution: An overview. *Computer Communications*. 2022. Vol. 182. P. 129–152.

78. Tran-Dang H. ,Bhardwaj S., Rahim T., Musaddiq A., Kim D. S. Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues. *Journal of Communications and Networks*. 2022. Vol. 24(1). P. 83–98.

79. Aqib M., Kumar D., Tripathi S. Machine learning for fog computing: review, opportunities and a fog application classifier and scheduler. *Wireless Personal Communications*. 2023. Vol. 129(2). P. 853–880.

80. Torabi E., Ghobaei-Arani M., Shahidinejad A. Data replica placement approaches in fog computing: a review. *Cluster Computing*. 2022. Vol. 25(5). P. 3561–3589.

81. Kalyani Y., Collier R. A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture. *Sensors*. 2021. Vol. 21(17). P. 5922.

82. Gill M., Singh D. A comprehensive study of simulation frameworks and research directions in fog computing. *Computer Science Review*. 2021. Vol. 40. P. 100391.

83. Asghari A., Sohrabi M. K. Server placement in mobile cloud computing: A comprehensive survey for edge computing, fog computing and cloudlet. *Computer Science Review*. 2024. Vol. 51. P. 100616.

84. Muniswamaiah M., Agerwala T., Tappert C. C. Fog computing and the internet of things (IoT): a review. *In 2021 8th IEEE international conference on cyber security and cloud computing (CSCloud)/2021 7th IEEE international conference on edge computing and scalable cloud (EdgeCom)*. 2021. P. 10-13.

85. Bansal S., Aggarwal H., Aggarwal M. A systematic review of task scheduling approaches in fog computing. *Transactions on Emerging Telecommunications Technologies*. 2022. Vol. 33(9). P. 4523.

86. Alzoubi Y. I., Gill A., Mishra A. A systematic review of the purposes of Blockchain and fog computing integration: classification and open issues. *Journal of Cloud Computing*. 2022. Vol. 11(1). P. 80.

87. Abdulazeez D. H., Askar S. K. Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment. *IEEE Access*. 2023. Vol. 11. P. 12555–12586.

88. Meng Y., Naeem M. A., Almagrabi A. O., Ali R., Kim H. S. Advancing the state of the fog computing to enable 5g network technologies. *Sensors*. 2020. Vol. 20(6). P. 1754.

89. Rahimikhanghah A., Tajkey M., Rezazadeh B., Rahmani A. M. Resource scheduling methods in cloud and fog computing environments: a systematic literature review. *Cluster Computing*. 2022. P. 1–35.

90. Sharief F., Ijaz H., Shojafar M., Naeem M. A. Multi-Class Imbalanced Data Handling with Concept Drift in Fog Computing: A Taxonomy, Review, and Future Directions. *ACM Computing Surveys*. 2024. Vol. 57(1). P. 1–48.

ДОДАТОК А
(обов’язковий)

ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

```
# Вхід: Повідомлення у форматі Base64
# Результат: Передати повідомлення на хмарний сервер або зберегти розшифроване повідомлення у локальну базу даних


for Each received message do      # Для кожного отриманого повідомлення
dataHex ← Decode Base64 encoded message to hexadecimal format
# Декодувати Base64 у шістнадцятковий формат
mhdr ← getMhdr(dataHex)          # Отримати заголовок MAC (MHDR)
if mhdr = Unconfirmed Data Up then          # Якщо це непідтверджене повідомлення "вгору"
macPayload ← getMacPayload(dataHex)          # Отримати MAC корисне навантаження
fhdr ← getFhdr(macPayload)       # Отримати MAC-заголовок (FHDR)
devAddr ← getDevAddr(fhdr)          # Отримати адресу пристрою (DevAddr)
if devAddr is not in local database then          # Якщо адреси немає в локальній базі
Contact application server to negotiate nwkSKey and appSKey
# Зв'язатися з сервером застосунків для отримання ключів сеансу
else
if devAddr belongs to current LoRaWAN network then   # Якщо адреса належить цій мережі
if devAddr belongs to private device then        # Якщо це приватний пристрій
nwkSKey ← loadNwkSKey(devAddr)           # Завантажити мережевий ключ
appSKey ← loadAppSKey(devAddr)          # Завантажити прикладний ключ
frmPayload ← getFrmPayload(macPayload)      # Отримати FRMPayload
fPort ← getFPort(macPayload)          # Отримати номер порту
```

```
fCnt ← getFCnt(fhdr)                    # Отримати номер кадру
dir ← 1          # Напрямок передачі (вгору)
if fPort = 0 then
key ← nwkSKey          # Якщо порт 0 — використовуємо NwkSKey
else
key ← appSKey          # Інакше використовуємо AppSKey
end if
decryptedPayload ← decryptPayload(frmPayload, devAddr, dir, fCnt, key)
# Розшифрувати корисне навантаження
Store decryptedPayload in local database
# Зберегти розшифроване повідомлення в локальну базу
end if
end if
end if
end if
devAddr does not belong to private device then
# Якщо пристрій не є приватним
Forward message to cloud server
# Передати повідомлення на хмарний сервер
end if
end for
```

ДОДАТОК Б
(обов’язковий)

НАУКОВА ПРАЦЯ ЗДОБУВАЧА

У сучасному цифровому середовищі Інтернет речей (IoT) стрімко інтегрується в повсякденне життя – від «розумних» міст до автоматизованих виробництв і систем охорони здоров’я. Передавання усіх даних до віддаленого хмарного середовища є нераціональним через затримки, перевантаження мережі та ризики втрати даних.

Метою цієї роботи є аналіз архітектур IoT з урахуванням контролю передачі даних та концепції туманних обчислень (Fog Computing) для підвищення ефективності, безпеки та енергоефективності систем.

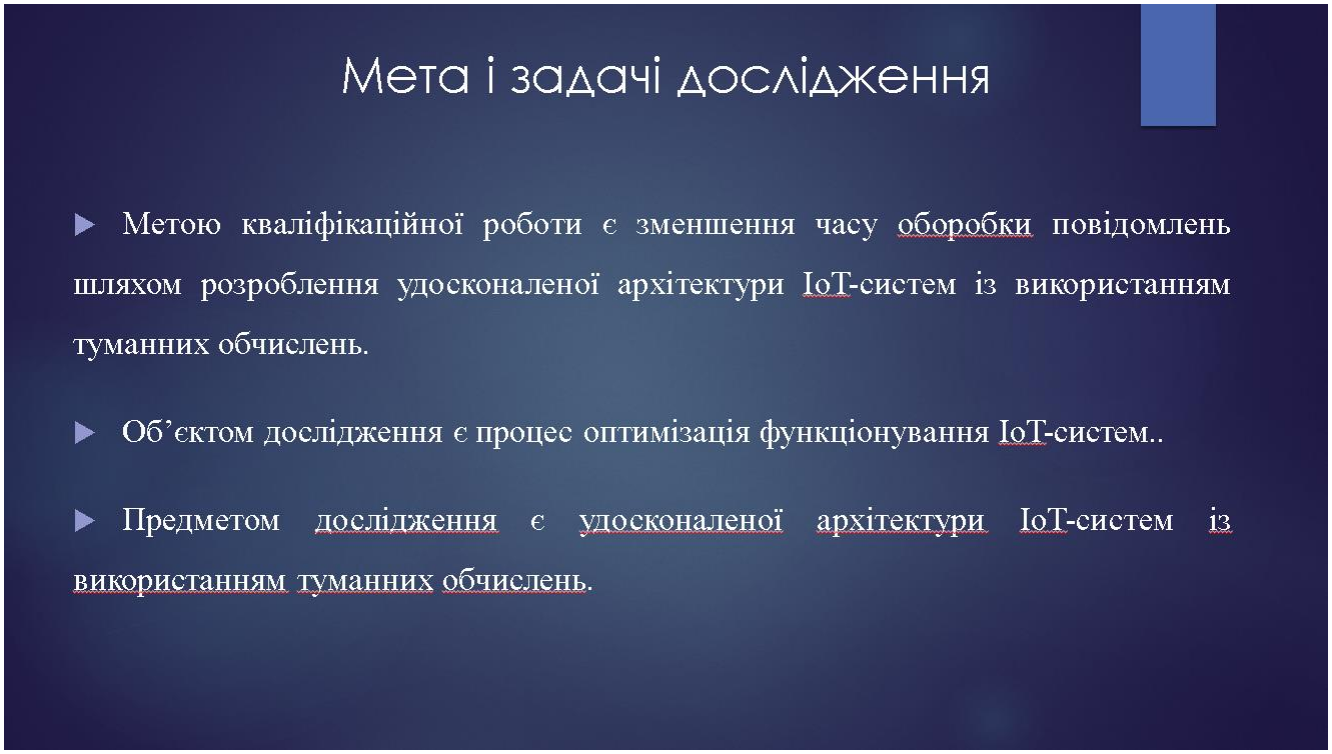
Традиційна централізована модель IoT передбачає обробку даних у хмарі. Проте така схема має низку недоліків: високу затримку, загрозу перевантаження каналів зв'язку та уразливість до збоїв. У свою чергу, туманні обчислення передбачають перенесення частини обробки ближче до пристроїв користувачів, що суттєво знижує затримку та покращує швидкість прийняття рішень.

Варіанти архітектур туманних IoT систем можуть бути централізованими, децентралізованими та гібридними. Централізовані моделі спрощують управління, однак мають проблеми з масштабуванням. Децентралізовані забезпечують вищу відмовостійкість і гнучкість. Гібридні поєднують обидва підходи, дозволяючи балансувати між централізованим контролем і локальною автономією.

Для ефективної побудови туманної IoT архітектури доцільно використовувати адаптивні моделі управління ресурсами, що реагують на зміни навантаження та умов середовища. Також важливо розробити стандарти взаємодії для сумісності пристроїв і масштабування систем без суттєвих змін.

ДОДАТОК В
(обов'язковий)

ПРЕЗЕНТАЦІЯ РОБОТИ



Мета і задачі дослідження

Поставлена мета досягається розв'язанням таких основних задач:

- дослідити архітектуру IoT туманних обчислень на основі контролю передачі даних та туманних обчислень;
- проаналізувати сучасні архітектури IoT туманних обчислень;
- дослідити методи оптимізації передачі даних у розподілених системах;
- удосконалити архітектуру IoT туманних обчислень на основі контролю передачі даних та туманних обчислень;
- реалізувати засоби удосконаленої архітектури IoT-систем із використанням туманних обчислень.

Наукова новизна та практична цінність отриманих результатів

Наукова новизна отриманих результатів:

- Набула подальшого архітектура IoT-систем із використанням туманних обчислень, яка на відміну від відомих для оптимізація часу оборобки повідомлень застосовує туманні обчислення.
- Набули подальшого розвитку програмно-технічні засоби удосконаленої архітектури IoT-систем із використанням туманних обчислень

Актуальність дослідження

Застосування IoT-fog-інфраструктур в цифровому світі є дуже актуальним:

- Мінімізація затримок.
- Зменшення навантаження на хмару.
- Забезпечення безпеки і конфіденційності.
- Підвищення надійності.
- Сприяння сталому розвитку.

Тому IoT-fog-інфраструктури залишаються ключовими елементами для забезпечення автономності, надійності та ефективної обробки даних.

АРХІТЕКТУРА IOT ТУМАННИХ ОБЧИСЛЕНЬ

Загальна концепція архітектури IoT туманних обчислень включає в себе інтеграцію туманних шлюзів для обробки даних, зниження затримок передачі інформації та забезпечення конфіденційності.

Така архітектура створюється для того, щоб оптимізувати ефективність IoT-систем за рахунок застосування туманних обчислень, де обчислення і зберігання даних частково виконуються на локальних шлюзах, що зменшує навантаження на хмарну інфраструктуру і знижує затримки.

Основною метою цієї архітектури є надання гнучкої і масштабованої системи, що здатна адаптуватися до змінних умов роботи мережі, обсягів трафіку та вимог до конфіденційності даних.

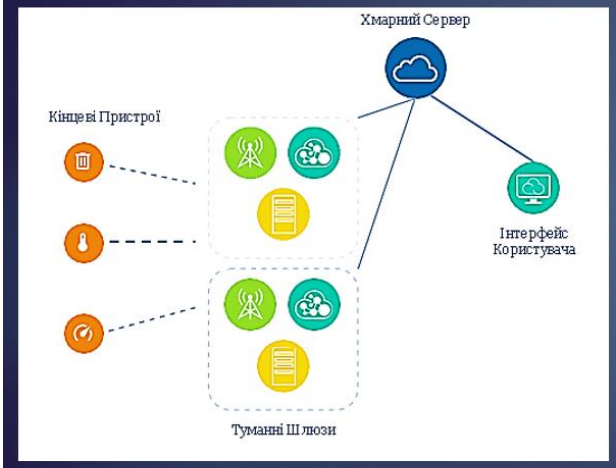
АРХІТЕКТУРА IOT ТУМАННИХ ОБЧИСЛЕНЬ

Основними компоненти архітектури IoT туманних обчислень є:

1. Кінцеві пристрої – це основні елементи, що відповідають за збір даних з навколишнього середовища через різноманітні сенсори та акутори.
2. Туманний шлюз приймає дані від кінцевих пристроїв, проводить попередню обробку і може зберігати деякі з цих даних локально, щоб знизити затримки та обсяг трафіку, що передається до хмари.
3. Хмара виконує функції зберігання великих обсягів даних і більш складної аналітики.
4. Механізм контролю передачі даних відповідає за динамічне управління потоком даних між кінцевими пристроями, туманним шлюзом і хмарною інфраструктурою.
5. Механізм забезпечення конфіденційності відповідає за обробку приватних і публічних даних, забезпечуючи їх відповідне зберігання та обробку.

АРХІТЕКТУРА IOT ТУМАННИХ ОБЧИСЛЕНЬ

Архітектура А



Запропонована архітектура А ґрунтується на ідеї інтеграції всіх стандартних елементів мережі LoRaWAN в один пристрій, який називається туманним шлюзом. Таким чином, кожен шлюз, окрім приймання, перетворення та ретрансляції пакетів, виконує також функції приватних мережевого та прикладного серверів. Такий підхід може забезпечити швидшу реакцію на поточні умови завдяки обробці актуальних даних на місці.

Рисунок 1 – Діаграма архітектури А

АРХІТЕКТУРА IOT ТУМАННИХ ОБЧИСЛЕНЬ

Архітектура B

Для коректної роботи цієї архітектури необхідно застосувати певну модель комунікації. Оскільки один складний вузол керує кількома вузлами з обмеженою логікою, застосовується модель master/slave.

Модель Майстер/Підлеглий – це модель комунікації, за якої один вузол (Майстер) керує кількома підлеглими вузлами та опосередковує комунікацію між ними.

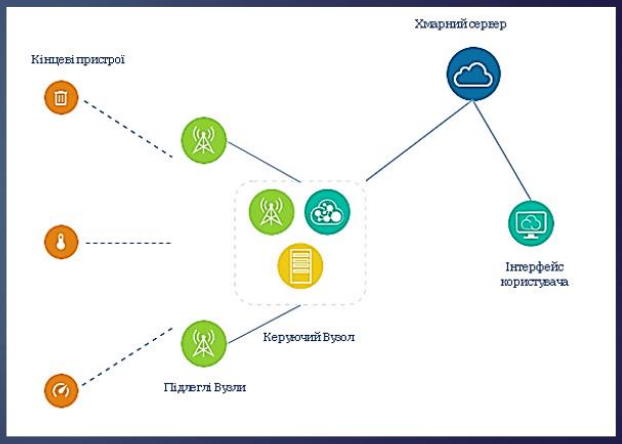


Рисунок 2 – Діаграма архітектури B

АРХІТЕКТУРА IOT ТУМАННИХ ОБЧИСЛЕНЬ

Архітектура C

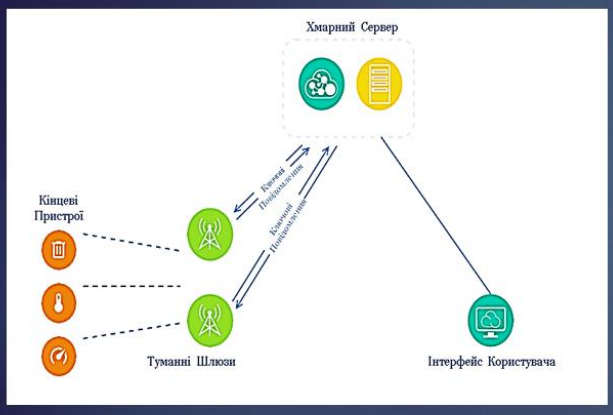


Рисунок 3 – Діаграма архітектури C

Архітектура C ґрунтується на принципово іншому підході до оптимізації, ніж попередні архітектури. Замість повторного розміщення окремих компонентів стандартної архітектури LoRaWAN, головна відмінність полягає у способі комунікації між шлюзом і сервером додатків. Таким чином, з точки зору структури, архітектура C не відрізняється від стандартної архітектури LoRaWAN.

Проєктування, реалізація та перевірка IoT-фог архітектури

Для запропонованих архітектур A і B час виконання шлюзу (форвардера пакетів) (Tpktfwd) вимірювався до моменту пересилання повідомлення на мережевий сервер, який далі здійснював обробку повідомлення. Час виконання мережевого сервера вимірювався з моменту отримання ним повідомлення до моменту його обробки та пересилання на сервер застосунку. Час виконання на сервері застосунку вимірювався з моменту отримання повідомлення до моменту додавання розшифрованого корисного навантаження до бази даних. Експериментальні вимірювання були проведені для 1000 повідомлень. За результатами вимірювань були обчислені середні значення часу та занесені у таблицю 1.

Таблиця 1 – Час виконання компонентів для архітектур A та B

Компонент	Середній час виконання (мс)
Форвардер пакетів (шлюз)	0.16
Мережевий сервер	194.62
Сервер застосунку	15.16

Проєктування, реалізація та перевірка IoT-фог архітектури

У запропонованій Архітектурі C використовується інший підхід, оскільки форвардер пакетів спеціально модифікований для виконання функцій туманних обчислень. У цьому випадку форвардер пакетів не лише пересилає дані на мережевий сервер, але й самостійно декодує повідомлення та розшифровує корисне навантаження. Тому до експериментально виміряного часу виконання форвардера необхідно додати час виконання окремих функцій, таких як декодування повідомлення (T_{decode}), розшифрування корисного навантаження ($T_{decrypt}$) та подальшого збереження цього навантаження в базі даних ($T_{database}$):

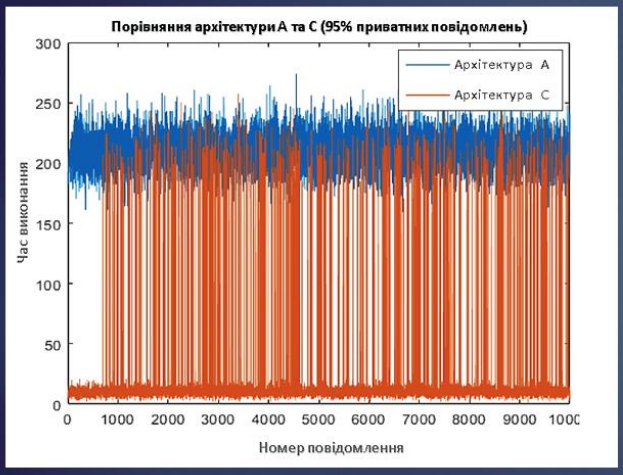
$$T_{fog} = T_{decode} + T_{decrypt} + T_{database},$$

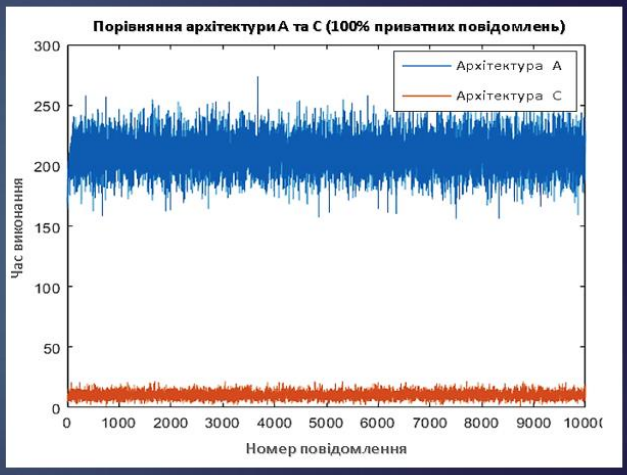
$$T_{total} = T_{pktfwd} + T_{fog}.$$

Таблиця 2 – Час виконання компонентів для архітектури C

Компонент часу	Середній час виконання (мс)
T_{decode}	1.95
$T_{decrypt}$	1.19
$T_{database}$	7.05
T_{fog}	10.19
T_{total}	10.35

Проєктування, реалізація та перевірка IoT-фог архітектури





Проєктування, реалізація та перевірка IoT-фог архітектури

Таблиця 3 – Підсумок характеристик архітектур

Характеристика	Архітектура A	Архітектура B	Архітектура C
Середній час виконання	209.96 мс	Подібний до арх. A	19.44 мс
BCET (найкращий сценарій)	209.81 мс	Подібний до арх. A	10.28 мс
WCET (найгірший сценарій)	217.78 мс	Подібний до арх. A	210.41 мс
Можливість впровадження в LoRaWAN	Ні	Ні	Так
Автономна робота	Так	Так	Так
Обчислювальні вимоги	Високі	Високі	Низькі

В усіх запропонованих архітектурах туманні шлюзи можуть працювати автономно, якщо зв'язок з хмарним сервером втрачено. Інтеграція серверів у туманний шлюз значно підвищує обчислювальні вимоги до нього.

Підсумовуючи основні характеристики кожної з архітектур, із таблиці 4 видно, що найкращі результати показала архітектура C. Тому для подальшої реалізації прототипу було обрано архітектуру C.

Проєктування, реалізація та перевірка IoT-фог архітектури!!!

Для реалізації було вирішено використати програмне забеспечення ChirpStack, яке забезпечує гнучкість у створенні приватних LoRaWAN-мереж без залежності від загальної інфраструктури.

Поточна архітектура мережі на кампусі VSB-TUO включає кінцеві пристрої, шлюзи, сервери The Things Network (TTN), сервер KoIoT і веб-додаток для користувачів. Шлюзи LoRaWAN виступають містком між бездротовими і проводовими компонентами мережі. Кожен шлюз підключений до Інтернету через Ethernet для забезпечення з'єднання з бекендом TTN. TTN виконує функції маршрутизації IoT даних між кінцевими пристроями та додатками.

Проєктування. Експерименти.

Для порівняння часу обробки між оптимізованою архітектурою та іншими двома рішеннями – поточною архітектурою на кампусі і мережею на базі ChirpStack – було виміряно загальний час обробки повідомлення, від його надходження на шлюз до зберігання розшифрованого payload у базі даних.

В оптимізованій архітектурі обробка payload відбувається на fog gateway, що значно знижує час обробки.

Для проведення тестування були використані два кінцеві пристрої, які надсилали повідомлення кожні 10 секунд. Один пристрій оброблявся через fog gateway, а інший – через хмару.

Середній час обробки в оптимізованій архітектурі становив 121.74 мс, коли деякі повідомлення оброблялися в хмарі.

При обробці усіх повідомлень на fog gateway середній час становив 10.44 мс.

Для мереж на базі ChirpStack середній час обробки становив 231.92 мс, а для поточної мережі на кампусі з TTN – 297.93 мс.

У порівнянні з поточною мережею на кампусі оптимізована архітектура дозволяє зменшити час обробки більш ніж удвічі.

Проєктування, реалізація та перевірка IoT-фог архітектури

Таблиця 5 – Результати тесту конфіденційності

Місце зберігання БД	Приватний кінцевий пристрій	Публічний кінцевий пристрій
Fog gateway	99	0
Cloud	1	100
Total	100	100

У таблиці 5 зазначені результати тесту на конфіденційність. Приватні повідомлення обробляються і зберігаються локально на fog gateway, а непри приватні – відправляються в хмару.

На рисунку 12 видно, що при порівнянні часу обробки, оптимізована архітектура забезпечує значно нижчу затримку в обробці повідомлень, оскільки fog gateway може обробляти повідомлення локально.

Рисунок 11 – Результати порівняння часу обробки

публікації

- За темою кваліфікаційної роботи магістра опубліковані тези у матеріалах XXVI Міжнародної студентської науково-технічної конференції «Перспективні мережі та комп'ютерні технології» ПерСиК 2025, 17 квітня 2025 р., Харків, Україна.

Висновки

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено архітектуру IoT-систем із використанням туманних обчислень.

У першому розділі описано сучасні підходи до побудови IoT-архітектур, обґрунтовано вибір fog computing як більш ефективної моделі для обробки даних у реальному часі.

У другому розділі проаналізовано методи оптимізації та схеми маршрутизації даних, що можуть бути використані в туманних IoT-мережах.

У третьому розділі запропоновано та описано детальну архітектуру системи на основі fog computing з підтримкою конфіденційності, маршрутизації, автономності.

У четвермоу розділі реалізовано програмний прототип системи з використанням реального обладнання та open-source платформ. А також проведено експериментальну перевірку – порівняння з TTN/ChirpStack, тести на затримку і обробку приватних даних. Підтверджено переваги запропонованого рішення.

Висновки

У процесі експериментального тестування підтверджено ключові властивості системи:

- зменшення часу обробки повідомлень у понад 20 разів у режимі локальної обробки (до 10.44 мс проти 297.93 мс у TTN);
- коректне функціонування механізму приватності з повним збереженням приватних даних на шлюзі;
- автоматичне перемикання у режим офлайн з локальним зберіганням даних при втраті з'єднання з хмарою;
- сталість часу обробки у мережах ChirpStack та TTN підтверджує об'єктивність результатів;
- забезпечення масштабованості та стабільності при змінних умовах мережі.

Отримані результати підтвердили гіпотези, сформульовані в теоретичних розділах, та довели доцільність використання туманних обчислень у сучасних IoT-мережах.

Протокол аналізу звіту подібності експертом

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

Автор: Валентин МОСІЙЧУК

Співавтор:

Назва: Мосійчук_Архітектура IoT туманних обчислень на основі контролю передачі даних та туманних обчислень

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:9.6%

Коефіцієнт подібності 2:2.4%

Мікропробіли: 0

Заміна букв: 1

Інтервали: 0

Білі знаки: 1

Дата створення звіту: 2025-05-06 21:16:23.0

Після аналізу Звіту подібності констатую наступне:

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

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

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

Обгрунтування:

2025-05-07 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism v-15.274 Educational

The maximum coincidence with one document 2.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 8%**

	Document		Sum coincidence on the DB	
ID: 240918 Title: МКР Архітектура IoT туманних обчислень на основі контролю передачі даних та туманних обчислень Added in a DB: 2025-05-06 Authors: Валентин МОСІЙЧУК Heads: Сергій ЛИСЕНКО Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	139510	1036	3912 (3%)	46 (4%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Валентин МОСІЙЧУК

Тема: Архітектура IoT на основі контролю передачі даних та туманних обчислень

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень – ; кількість сторінок записки 83

1. Короткий зміст роботи та прийнятих рішень Було розроблено архітектуру з можливістю гнучкого вибору обробки даних та механізмами динамічного реагування на статус з'єднання і політику конфіденційності.

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

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено огляд сучасних підходів до побудови IoT-архітектур. Досліджено відомі рішення та засоби в цій сфері. У другому розділі проаналізовано методи оптимізації та схеми маршрутизації даних. У третьому розділі запропоновано та описано детальну архітектуру системи на основі fog computing з підтримкою конфіденційності, маршрутизації, автономності. У четвертому розділі реалізовано програмний прототип системи з використанням реального обладнання та open-source платформ

4. Позитивні сторони роботи: Запропонована архітектура IoT туманних обчислень дозволяє зменшити час обробки повідомлень і підвищити надійність системи при втраті зв'язку з хмарою.

5. Негативні сторони роботи: В роботі присутні певні логічні помилки щодо опису методів оптимізації передачі даних та алгоритмів оцінки моделей.

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

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

8. Інші зауваження: –

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «задовільно» 3.00 (E)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) д.т.н., професор, Мартинюк В.В., завідувач кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

"5 травня" 2025р.

Завідувачу кафедри КІІС
доктору філософії, доценту
Ользі ПАВЛОВІЙ

Валентина МОСІЙЧУКА
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-23-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (StrikePlagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

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

6 травня 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Архітектура IoT на основі контролю передачі даних та туманних обчислень

Автор: Валентин МОСІЙЧУК

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Сергій ЛИСЕНКО, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

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

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 9.63% і адресується до 69 першоджерела; та системою Anti-Plagiarism складає 2.0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Сергій ЛИСЕНКО

Гарант ОП _____ Олег САВЕНКО

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА