Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи та засоби моделювання розподілених систем туманних обчислень

(тема)

Виконав:
студент II курсу, групи СПм-23 – 2
Журавель Д.С.
(прізвище, ініціали)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: проф. Волк М.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Журавлю Денису Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи та засоби моделювання розподілених систем туманних обчислень

затверджена наказом по університету від "22" листопада 2024 р. № 1236ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 січня 2025 р.

3. Вхідні дані до роботи

1. Технології моделювання хмарних систем
2. Туманні обчислення (Fog Computing)
3. Існуючі системи моделювання туманних обчислень
4. Методи та системи моделювання енергоспоживання

4. Перелік питань, що потрібно опрацювати в роботі

1 Аналіз предметної області
2 Моделі та методи моделювання розподілених систем туманних обчислень
3 Експериментальні дослідження
4 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Демонстраційні матеріали. Плакати – 12 арк. ф. А4

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	25.11.24-27.11.24	
2	Розробка моделей	28.11.24-04.12.24	
3	Реалізація алгоритмів	05.12.24-15.12/24	
4	Розробка структури програмних засобів	16.12.24-25.12.24	
5	Розробка програмних модулів	25.12.24-28.12.24	
6	Оформлення матеріалів кваліфікаційної роботи	29.12.24-03.01.25	
7	Подання кваліфікаційної роботи керівникові та її попередній захист	04.01.25-07.01.25	
8	Подання кваліфікаційної роботи на рецензування	08.01.25-20.01.25	

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Волк М.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 56 с., 18 рис., 3 табл, 1 дод., 36 джерел.

ТУМАННІ ОБЧИСЛЕННЯ, ІОТ, УПРАВЛІННЯ РЕСУРСАМИ, СИМУЛЯТОР, МЕРЕЖЕВІ ФУНКЦІЇ, МОБІЛЬНІСТЬ, АЛГОРИТМИ ПЛАНУВАННЯ, ЕНЕРГОЕФЕКТИВНІСТЬ, ПЕРЕДАЧА ЗАВДАНЬ

У роботі представлено розробку симулятора туманних обчислень, що забезпечує моделювання та аналіз сценаріїв у мережах із залученням IoT-пристроїв. Запропонована система дозволяє реалізовувати алгоритми управління ресурсами, передачі завдань, а також тестувати моделі планування у розподілених системах. FogGRASS забезпечує підтримку різноманітних мережевих функцій, моделей мобільності та протоколів зв’язку, що дозволяє дослідникам створювати реалістичні мережеві сценарії. Проведено експериментальне дослідження масштабованості, використання ресурсів та енергоефективності. Результати роботи демонструють високу продуктивність запропонованої системи, що відкриває нові можливості для подальших досліджень у сфері туманних обчислень.

ABSTRACT

Master’s thesis: 56 pages, 18 figures, 3 tables, 1 appendice, 36 sources.

FOG COMPUTING, IOT, RESOURCE MANAGEMENT, SIMULATOR, NETWORK FUNCTIONS, MOBILITY, SCHEDULING ALGORITHMS, ENERGY EFFICIENCY, TASK HANDOVER

The paper presents the development of a fog computing simulator that provides modelling and analysis of scenarios in networks involving IoT devices. The proposed system allows the implementation of resource management algorithms, task transfer, and testing scheduling models in distributed systems. FogGRASS provides support for various network functions, mobility models, and communication protocols, which allows researchers to create realistic network scenarios. An experimental study of scalability, resource utilization, and energy efficiency was conducted. The results demonstrate the high performance of the proposed system, which opens up new opportunities for further research in the field of fog computing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ВМ – віртуальна машина

ОС – операційна система

ПЗ – програмне забезпечення

AKS – Azure Kubernetes Service

API – Application Programming Interfase

DIS – Distributed Interactive Simulation

DEVS – Discrete Event System Specification

IaaS – Infrastructure as a Service

EKS – Elastic Kubernetes Service

ID – identity

FCFS – First-Come First-Served

FPGA – Field-Programmable Gate Array

GKE – Google Kubernetes Engine

HAL – Hardware Abstraction Layer

HLA – High Level Architecture

HPC – High Performance Computing

HPF – Highest Priority First

M&S – Modelling and Simulation

SaaS – Simulation as a Service

VM – Virtual Machine

VPN – Virtual Private Network

ВСТУП

За минулі роки в інформаційно-комунікаційних технологіях відбулися певні зміни парадигми від ізольованих і обмежених обчислювальних середовищ до хмарних обчислень. Розвиток телекомунікаційної та виробничої промисловості призвів до розробки потужних інтелектуальних пристроїв, які можуть вільно приєднуватися до доступної мережі в будь-який час. В останні роки ми спостерігаємо еволюцію 5G. Включення технології 5G не тільки підтримує неоднорідне підключення до бездротової мережі, але й надає широкий спектр різноманітним додаткам, які можуть скористатися перевагами 5G, щоб покращити свою продуктивність з точки зору, наприклад, затримки, часу відгуку та споживання енергії. З розвитком технологій пристрої можуть стати частиною мережі та генерувати значну кількість даних, які зазвичай надсилаються в хмару для обробки.

Мільярди пристроїв підключені до Інтернету. Необроблені дані, створені різними датчиками, вимагають інтелектуальної обробки, щоб зменшити використання пропускної здатності та покращити затримку. Тому потреба наблизити ресурси до кінцевого споживача зростає.

Fog (туман) є однією з нових парадигм периферійних обчислень. Термін туман вперше ввів Cisco [1]. У порівнянні з хмарними центрами обробки даних, fog забезпечує віртуалізоване обчислювальне середовище, розгорнуте ближче до кінцевого користувача [2]. Туман знаходиться між хмарою та пристроями кінцевого користувача. Хмара та туман надають подібні послуги кінцевим користувачам, але туман розгортається для сприяння певному географічному регіону. Туман не може існувати автономно, він доповнює хмарні обчислення. Він розроблений для підтримки програм, чутливих до затримки задач, тоді як хмара, яка знаходиться далеко від користувача, може демонструвати більшу затримку. Fog надає послуги для додатків Інтернету речей як із периферійної мережі, так і з таких

пристроїв, як маршрутизатори, точки доступу, придорожні пристрої (RSU) та інші пристрої користувача.

Завдяки розгортанню туманних вузлів можна легко керувати надійністю, відмовостійкістю та масштабованістю пристроїв. Таким чином, це також зменшує використання пропускної здатності між туманним вузлом і серверними хмарними центрами обробки даних.

Перебуваючи в стані розвитку, туманні обчислення не мають стандартизації з точки зору архітектури та платформ моделювання. На сьогоднішній день доступна низка симуляторів туманних систем, деякі з яких є відкритими, тоді як решта є комерційно доступними. Існуючі симулятори туману в основному зосереджені на ряді пристроїв, які можна моделювати. Як правило, існуючі симулятори більш схильні до конфігурацій датчиків, де датчики генерують необроблені дані, а вузли туману використовуються для інтелектуальної обробки даних перед надсиланням у серверну хмару або інші вузли. Таким чином, ці симулятори не мають мережевих властивостей і передбачають надійну та безпомилкову доставку на кожен запит служби. Крім того, жоден симулятор не дозволяє дослідникам включати власні алгоритми керування туманними вузлами, такі як планування. В даній роботі передача пристрою також не підтримується.

У кваліфікаційній роботі пропонується новий симулятор туманних систем під назвою FogGRASS, який надає користувачам детальні параметри конфігурації для імітації великої мережі туманних пристроїв. Це дозволяє дослідникам включати індивідуальні моделі мобільних пристроїв, алгоритми планування вузлів туману та керувати механізмами передачі даних. Проводиться оцінювання системи керування трафіком, щоб продемонструвати масштабованість і ефективність запропонованого симулятора з точки зору використання центрального процесора та пам’яті. В моделях також використовуються параметри мережі, такі як затримка виконання та частота помилок пакетів передачі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Туманні обчислення (Fog Computing)

Туманні обчислення – це нова парадигма, але існує кілька проблем, з якими вони стикаються, наприклад, керування гетерогенними пристроями, архітектура вузлів туману, безпека, мобільність пристроїв і конфіденційність. Крім того, сумісність різнорідних пристроїв також є складним завданням. Дані, що надходять з різних пристроїв, необхідно аналізувати та пересилати на інші пристрої для своєчасного прийняття рішень. Типова парадигма туману складається з обмежених вузлів туману, які надають віртуалізовані обчислювальні послуги. Тому для ефективного управління ресурсами потрібні складні алгоритми планування. На рисунку 1.1 зображено загальний сценарій, у якому ефективність досягається шляхом впровадження шару туману між локальною мережею та хмарою. Подібно до хмари, належне керування використанням ресурсів і виставлення рахунків є обов'язковою вимогою туманної парадигми.

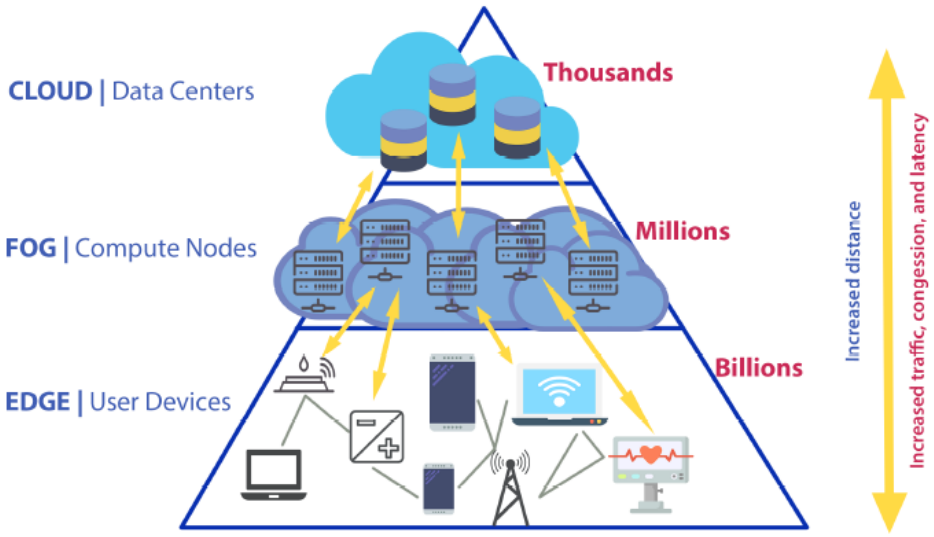

Рисунок 1.1 – Піраміда обчислювальних архітектур

Fog забезпечує гнучке середовище для кінцевих пристроїв, але ним важко керувати через його децентралізовану парадигму. Туманні вузли можна розгорнути в будь-якому місці між пристроєм і хмарною інфраструктурою. Оскільки туман розроблено для підтримки програм, чутливих до затримки, розумні шлюзи (SG) є найкращим місцем для надання обчислень туману [3].

Fog computing базується на дорогих пристроях і мережевому обладнанні. Важливо провести тестування туманної платформи перед розгортанням за допомогою засобів моделювання. Однак на даний момент немає такого стандартного інструменту/фреймворку моделювання для обчислення туману, що робить моделювання туману також відкритою проблемою досліджень.

Крім того, існуючі симулятори туману не мають основних функцій, Які обговорюються нижче. Існуючі симулятори в основному орієнтовані на однорідні пристрої. Пристрої надсилають дані до централізованого вузла, де дані обробляються перед надсиланням у хмару. Існують симулятори на основі Java, які не враховують властивості мережі і, таким чином, моделюють лише ідеальне середовище без частоти падіння пакетів/помилок, перевантаження мережі та конфлікту каналів. Симулятори надають лише обмежені або прості моделі мобільності та не мають таких функцій, як передача. Симулятори не є відкритими або пропрієтарними і не підтримують модифікації. Таким чином, дослідники не можуть використовувати власні алгоритми керування ресурсами для тестування та модифікацій. Крім того, моделі обчислення споживання енергії та витрат також не підтримуються в існуючих симуляторах.

1.2 Існуючі системи моделювання туманних обчислень

Розглянемо деякі з часто використовуваних симуляторів у порівнянні із запропонованим FogGRASS. Хан та ін. [4] запропонував DPWSim,

розроблений для додатків IoT. Він підтримує сервіс-орієнтовану модель, керовану подіями. Сотіріадіс та ін. [5] запропонував SimIoT, який розроблено як розширення системи моделювання SimIC. Він надає кілька механізмів зв'язку для датчиків IoT і хмарних центрів обробки даних. Подібним чином пропонується інструмент моделювання EdgeCloudSim для зменшення бар'єрів у звичайних хмарних симуляторах для моделювання сценаріїв периферійних обчислень. Він розроблений як розширення CloudSim [6].

Хан та ін. [7] запропонували техніку координації для роботи з великою кількістю пристроїв IoT. Фреймворк реалізований на CloudSim і підтримує програми домашньої автоматизації. Zeng та ін. запропонував IoTSim, який дозволяє обробляти великі дані в середовищі IoT. Автори прийняли модель MapReduce і представили практичне дослідження, щоб продемонструвати ефективність запропонованого симулятора. Pflanzner та ін. [8] представили мобільний симулятор IoT під назвою MobIoTSim. Основна мета – надати дослідникам платформу для навчання роботі з пристроями Інтернету речей без покупки додаткових датчиків і пристроїв. За допомогою MobIoTSim користувачі можуть ознайомитися з детальною роботою систем IoT і допомогти в формуванні початкового розуміння повної підключеної системи IoT.

Клаудіо та ін. [9] запропонував CrowdSenSim, який розроблений для імітації додатків джерела натовпу, таких як розумні міста. Щоб оцінити запропонований симулятор, автори змоделювали сценарій вуличного освітлення. Однак симулятор можна використовувати для інших додатків, які потребують збору даних із різноманітних джерел. CrowdSenSim є у вільному доступі для дослідників і користувачів.

SimpleIoTSimualtor [10] є широко використовуваним інструментом для моделювання середовища IoT за допомогою датчиків і пристроїв. Він також підтримує низку специфічних протоколів IoT, таких як MQTT, CoA тощо. Однак SimpleIoTSimulator потребує 64-розрядного середовища RedHat Linux для успішного встановлення та виконання всіх пакетів. Подібним чином IBM

запропонувала симулятор IoT із підтримкою PaaS під назвою IBM Bluemix [11]. Він надає веб-інтерфейс для швидкого розгортання хмарних програм, які можуть збирати дані з різних датчиків і пристроїв. Він забезпечує підтримку апаратних пристроїв ARM, Intel і Texas Instrument. Дані надсилаються в хмарний центр обробки даних через протокол MQTT. Facebook запускає симулятор із підтримкою платформи як послуги - Parse [12]. Він підтримує пристрої IoT. Він забезпечує просте середовище розробки програм, а також підтримує різні мобільні пристрої. Фреймворк Google Cloud включає рішення Google IoT, яке надає різні сервіси Google [13]. Пропонований фреймворк має високу масштабованість і дозволяє використовувати велику кількість пристроїв, збирає дані та забезпечує платформу візуалізації. Зокрема, надходять дані з пристроїв у Load Balancer, а звідси він поширюється на наступний рівень, тобто AppEngine.

Харшит та ін. [14] запропонували симулятор IoT і туману під назвою iFogSim. Він розроблений як розширення CloudSim з використанням технології Java. Автори представили вплив методів управління ресурсами з точки зору затримки, перевантаження та вартості. Проте є кілька недоліків: (а) він заснований на Java, тому параметри основної мережі не підтримуються, (б) він несумісний з різними версіями java, крім того, документація невелика для підтримки розробки. У запропонованому нами симуляторі доступний повний мережевий інструментарій для симуляції затримки, затримки, перевантаження, відкидання пакетів, таким чином, він дає гнучкість для моделювання різноманітних мереж зі змінними характеристиками.

Запропоновано тестовий стенд моделювання IoT на основі рішення Cisco – називається FIT IoT Lab [15]. Він також підтримує рішення з відкритим кодом. Це дозволяє використовувати бездротові вузли, розгорнуті по всій Франції. Доступ до датчиків і шлюзів здійснюється через віддалений інтерфейс. Однак симулятор не проводить моделювання в контрольованому середовищі, тому моделювання не можна повторити. WSNet — симулятор,

призначений для імітації бездротових мереж [16]. Це симулятор дискретних подій. Його можна використовувати для моделювання мережі IoT. Він уже включає такі модулі, як мобільність, енергія, радіоінтерфейси та протоколи маршрутизації. Крім того, він також підтримує моделювання катастроф, таких як землетрус і пожежа.

Конті та ін. [23] запропонували схему, яка допомагає обчислювальним вузлам туману за допомогою батареї накопичувати енергію, яку можна використовувати для виконання інтенсивних обчислювальних завдань. Накопичувач живлення може допомогти затуманити вузол під час пікового навантаження навіть при низькому рівні генерації електроенергії. Автори запропонували методику навчання на основі підкріплення, щоб мінімізувати втрату роботи.

Харуна та ін. [24] обговорювали розподіл ресурсів і мобільність користувачів. Автори запропонували алгоритм для оптимізації розподілу ресурсів і обробки хендовера для мобільних користувачів у середовищі туману.

Мен та ін. [25] вивчали модель обмеження затримки для розвантаження обчислень на серверах хмарних або туманних обчислень. Основною метою є мінімізація енергоспоживання з точки зору обчислень і зв’язку, враховуючи обмеження затримки кожного завдання. Запропонований гібридний підхід зменшив загальне споживання енергії.

У туманних обчисленнях ресурси зазвичай обмежені порівняно з хмарними; тому типові алгоритми розподілу хмарних ресурсів не можуть задовольнити потреби користувача, що швидко змінюються. У [26] Ni та ін. запропонував алгоритм розподілу ресурсів для середовища туману з використанням тарифікованих у часі мереж Петрі. Мережа Петрі з тарифікацією в часі — це варіант мереж Петрі, які визначають систему на основі ціни та часу. Він використовується для характеристики динамічної поведінки системи, а також для ефективного аналізу продуктивності з точки зору часу та вартості. Запропонована схема враховує ціну та час виконання

завдання. Крім того, в цій роботі також пропонується алгоритм, який прогнозує час, необхідний для конкретного завдання.

Fog забезпечує проміжне програмне забезпечення для додатків IoT, де датчики/пристрої генерують значну кількість даних [27]. Подібним чином відбувається управління енергією периферійних пристроїв також дуже важливо. Jie Xu та ін. [28] запропонували структуру динамічного управління енергією, яка навчається на льоту та динамічно керує механізмом розвантаження робочого навантаження.

Ananthanarayanan та ін. [29] запропонував географічно розподілену структуру для великомасштабної відеоаналітики. Для підтримки відеоаналітики в реальному часі вузли шлюзу використовуються для виконання обчислювальних вимог. Ren та ін. [30] обговорили масштабовану структуру для периферійних обчислень.

Фреймворк розроблений для підтримки додатків розумного дому, зібрані дані від різних датчиків надсилаються до вузлів прийняття рішень, зв’язок може бути зменшений за допомогою встановлення сусідніх обчислювальних вузлів [31].

Модель передачі відео запропонована в [32] Xu et al. де сервер Mobile Edge Computing (MEC) використовується для керування механізмом кешування відео. Багато відомих постачальників контенту, як-от Youtube, NetFlix тощо, кешують вміст на межі, таким чином зменшуючи затримку. Чен та ін. в [33] запропонував мережевий фреймворк для MEC з використанням технології 5G. У технології 5G пристрій може обмінюватися даними з іншими пристроями. Таким чином, велика кількість пристроїв може вимагати обчислювальних ресурсів на краю.

Сонмез та ін. [6] запропонував структуру, яка визначає стратегії для кращого спільного використання ресурсів і виконання завдань у спільній та енергоефективний спосіб.

У [34] Тан та ін. запропонував розподілену архітектуру туману для інтеграції різних інфраструктур, послуг і компонентів для розумних міст.

Автори стверджують, що з розгортанням туманного вузла час відгуку значно скоротився.

Тінг та ін. [35] запропонував кооперативну структуру для підвищення якості збору даних у мережах оптоволоконних каналів. Обмеження затримки виконуються через концепцію розвантаження у вузлах динамічного туману. Мен та ін. [36] запропонували гібридну структуру розвантаження, яка може розвантажувати дані та обчислення у вузлах туману та хмарних центрах обробки даних. Основна мета — зменшити загальне споживання енергії для обчислень і зв'язку.

Мукерджі та ін. [37] обговорювали питання безпеки в середовищі IoT. Подібним чином Hu et al. [38] обговорили безпеку та конфіденційність програм виявлення облич, коли обчислення виконуються в шарі туману. Зазвичай використовувані протоколи для парадигми туману засновані на публікації/підписці.

Для вирішення питань конфіденційності пропонується схема підписки на публікацію на основі вмісту [39]. Аналогічно, архітектура туману також корисна для транспортних засобів, які вимагають швидкого обчислення для прийняття рішень у реальному часі.

Архітектура туману для транспортних обчислень запропонована в [2] Бономі та ін. Більшість симуляторів, розглянутих вище, не є гнучкими, щоб полегшити дослідникам включення власних алгоритмів. Крім того, існуючі симулятори в основному зосереджені на пристроях, які вони можуть підтримувати; однак мережеві протоколи та комунікаційний стек не враховуються повністю. Щоб перевірити та проаналізувати алгоритми, існує потреба в гнучкому, широкому та швидкому розгортанні симуляторі туману.

Детальне порівняння показано в таблиці 1.1 У наступному розділі обговорюються деталі запропонованого симулятора для вирішення вищезгаданих проблем.

Таблиця 1.1 – Порівняння існуючих систем моделювання туманних обчислень

Симулятори	Мова програмування	ОС	Мережа	Відкритий код	Мобільні пристрої	Моделі користувача	Планування	Споживання енергії
MobileFog	NA	–	ні	ні	так	ні	ні	ні
Edge-Fog cloud	Python	All	ні	так	обмежено	ні	ні	ні
EdgeCloudSim	Java	All	ні	так	ні	ні	так	ні
EmuFog	Python	All	так	так	ні	ні	ні	ні
RECAP simulator	NA	–	обмежено	NA	ні	ні	ні	NA
FogTorch	Java	All	ні	так	ні	ні	ні	ні
Cooja	C	Linux	ні	так	обмежено	ні	ні	ні
iFogSim	Java	All	ні	ні	так	ні	так	ні
Google IoT Sim	NA	Cloud	ні	ні	так	ні	ні	ні
IBM BlueMax	Java/Python	Cloud	ні	ні	так	ні	ні	ні
SimpleIoTSimulato	Java	Unix	ні	ні	так	ні	ні	ні
MobIoTSim	Java	Linux	ні	так	так	ні	ні	ні
FogGRASS	C++	All	так	так	так	так	так	так

1.3 Методи та системи моделювання енергоспоживання

Для ефективного аналізу впливу розподілених енергетичних ресурсів (DER) у мережах середньої напруги (MV) і низької напруги (LV) підприємства повинні спочатку мати дані, необхідні для побудови детальних моделей розподільної мережі (DN). Компоненти, а також розширені інструменти моделювання, які дозволяють проводити комплексний аналіз для оцінки продуктивності системи та виявлення потенційних проблем. У цьому розділі представлено опис програмного пакета ГІС з відкритим вихідним кодом, у якому було створено плагін (QGIS), програмного забезпечення, для якого створено моделі (OpenDSS), а також короткий виклад найважливіших аспектів архітектури та функцій плагіна QGIS2OpenDSS.

Географічні інформаційні системи (ГІС) є потужним інструментом, який можна визначити як інтегрований набір даних, обладнання, програмного забезпечення та процесів, розроблених як комп'ютерна система для збору, керування, картографування та аналізу просторових даних [28]. ГІС зараз широко використовуються розподільними енергокомпаніями для керування та зберігання інформації про характеристики своїх активів, що, у свою чергу, дозволяє енергетикам використовувати ці дані (відстань повітряних ліній, тип провідника, дані про трансформатор, місцезнаходження клієнта та споживання тощо) для створення детальних мережевих моделей.

QGIS — це географічна інформаційна система з відкритим кодом (GIS), ліцензована згідно з GNU General Public License та є частиною Open-Source Geospatial Foundation (OSGeo), що дозволяє розробникам створювати плагіни для додавання додаткових функцій, які не є частиною оригінальної програмний стек [29].

OpenDSS — це всеосяжний інструмент моделювання системи енергоспоживання, головним чином для розподілених систем. Це пакет програмного забезпечення з відкритим вихідним кодом, який підтримує

майже всі аналізи в частотній області (RMS синусоїдального стаціонарного стану), які зазвичай виконують DNO в системах розподілу задач [30].

Ключовим аспектом цього керованого сценарієм інструменту моделювання є те, що він працює з численними режимами рішення, включаючи стандартний режим, щоденний режим, режим робочого циклу, режим Монте-Карло та кілька режимів, де навантаження змінюється залежно від часу, що має вирішальне значення для кількісної оцінки впливу змінних джерел і навантажень [31]. OpenDSS відрізняється від традиційних розв'язувачів радіальних схем, оскільки він розв'язує мережеві (мережі) системи розподілу так само легко, як і радіальні системи.

На рисунку 1.2 показана внутрішня архітектура програмного забезпечення та послідовність визначень, тобто елементів живлення, перетворення та керування, які містять характеристики та конфігурацію.

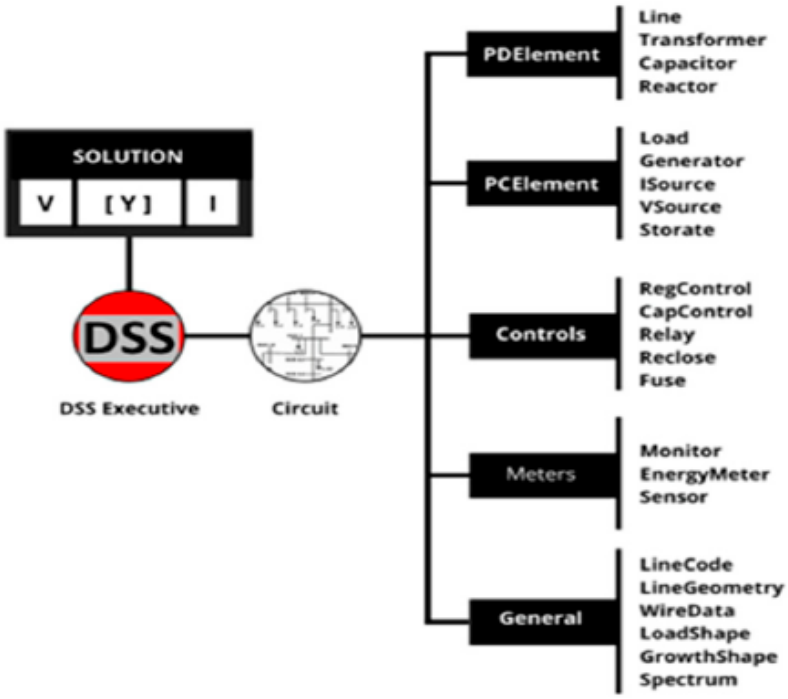

Рисунок 1.2 – Графічний інтерфейс користувача для QGIS

QGIS2OpenDSS витягує та обробляє дані ГІС для створення файлів OpenDSS (*.dss), необхідних для запуску симулятора. Плагін містить графічний інтерфейс користувача (GUI), який полегшує переклад даних у

файли, які містять інформацію, характеристики та атрибути класів об’єктів (рисунок 1.3). У GUI користувач вибирає рівень елементів DN для моделювання в OpenDSS, кожен шар має атрибути, які класифікуються як обов’язкові (суттєві) і додаткові (ці атрибути можуть бути витягнуті опосередковано з файлів, створених відділами ГІС електророзподільних компаній).

Рисунок 1.3 – Основні шари ГІС

Відповідні аспекти внутрішньої роботи плагіна, такі як зв’язок елеметів (здійснюється за допомогою Kdtrees бібліотеки Python [32]) і алгоритм розподілу профілю навантаження (де типовий профіль попиту призначається кожному завантаженню) виходять за рамки цієї роботи.

2 МОДЕЛІ ТА МЕТОДИ МОДЕЛЮВАННЯ РОЗПОДІЛЕНИХ СИСТЕМ ТУМАННИХ ОБЧИСЛЕНЬ.

2.1 Модель системи моделювання розподілених систем туманних обчислень

Пропонована система може підтримувати M мобільних пристроїв (m), F туманних вузлів та вузол брокера (B). Мобільні пристрої можуть спілкуватися з вузлом брокера через базову станцію . Мобільні пристрої можуть бути різних типів: або вони є сенсорними пристроями, які лише генерують дані для подальшої обробки у вузлі туману, або вони можуть розвантажувати завдання для обчислень. Таким чином, мобільний пристрій може виконувати програму, яка запитує обчислювальну потужність у вузлі туману.

Модель трафіку реалізується як черга M/M/1 на мобільному пристрої та чергу M/M/c на вузлах туману. Кожен мобільний пристрій може генерувати сервісний запит до вузла fog через свою базову станцію. Якщо загальна швидкість запитів менша за прийняту швидкість вузла туману, тоді всі запити будуть прийняті для обробки на вузлі туману. В іншому випадку запит далі передається в суборенду сусіднім вузлам туману. Усім цим процесом керує вузол брокера.

Вузол-посередник постійно відстежує туманні вузли, їх довжину черги та розміри кожного запиту. Крім того, припустимо, що перенесення завдання на внутрішній хмарний центр обробки даних може ще більше збільшити затримку, однак ця опція все ще доступна на вузлі посередника, яку можна використовувати в екстремальних випадках, коли швидкість запитів зростає експоненціально. Крім того, запити, згенеровані з мобільних пристроїв (m_i, $i \in M$) слідує моделі процесу Пуассона, де λ_i – середня швидкість надходження запитів.

Кожен запит, згенерований пристроєм m_i вимагає обчислень, повністю незалежний і може бути запланований на будь-якому туманному вузлі або хмарному центрі обробки даних. Крім того, існує k однорідних вузлів туману, які можуть бути використані для виконання отриманого завдання. У кожному вузлу туману q вузлів доступні для надання послуг. Швидкість виконання представлена як θ_f. Максимальне робоче навантаження туманного вузла представлено як φ_f. Причиною визначення максимального робочого навантаження є уникнення непотрібної затримки в черзі.

Виконавчий запит від m_i вперше отримається брокером. Таким чином, відповідно до процесу Пуассона ми можемо писати загальну інтенсивність прибуття даних як

$$\lambda = \sum_{i=0}^{M} \lambda_i \,. \tag{2.1}$$

Відзначимо запит, який може прийняти кожен туманний вузол як

$$\varphi = \begin{cases} 1, \text{якщо } \phi_f \geq \lambda \\ \dfrac{\phi_f}{\lambda}, \text{якщо } \phi_f < \lambda \end{cases}. \tag{2.2}$$

Таким чином, використовуючи (2.1) і (2.2), ми можемо обчислити швидкість виконання в вузлах туману як

$$S = \lambda \times \varphi = \begin{cases} \lambda, \text{якщо } \phi_f \geq \lambda \\ \phi_f, \text{якщо } \phi_f < \lambda \end{cases}. \tag{2.3}$$

Використовуючи формулу Ерланга [40] і аналіз теорії масового обслуговування, ми можемо обчислити середній час очікування кожного запиту у вузлі туману таким чином

$$T = \frac{\frac{\lambda}{k\theta_f}}{k\theta_f - \lambda} + \frac{1}{\theta_f}. \quad (2.4)$$

Вузол-посередник керує всіма вузлами туману, і якщо вузол туману не може обробити жодного нового запиту, посередник пересилає перевантажений запит сусіднім вузлам туману, які недостатньо використовуються. Крім того, в крайньому випадку завдання можна перенести в хмарний дата-центр (цей випадок ми тут не розглядаємо). Таким чином, якщо немає доступного туманного вузла, який може прийняти нове завдання, брокер може утримувати завдання у власній черзі протягом деякого часу *t*. Після закінчення часу він може переслати завдання на доступний туманний вузол. В іншому випадку після закінчення часу *t*, брокер може відмовитися від завдання та запросити повторну передачу.

Розглянемо модель енергоспоживання. Енергію даної задачі можна обчислити за допомогою виразу

$$E_t^f = \frac{Size_t}{N} Pw + \frac{Size_t}{\rho_f} Pw_{idle}, \quad (2.5)$$

де E_t являє собою спожиту енергію для завдання. Це комбінація енергії пристрою та енергії, споживаної у вузлі туману. Наприкінці пристрою енергія витрачається на завантаження завдання розміру *Size* на каналі даних пропускної здатності *N* з його потужністю передачі *Pw*. Тоді як потужність, споживана під час виконання завдання розміру *Size* у вузлі туману (*Fog*) з обчислювальною потужністю ρ_f. Pw_{idle} представляє потужність простою.

Тогда загальне енергоспоживання для *C* завдань буде оцінюватися як

$$E = \sum_{t=0}^{C} E_t^f. \quad (2.6)$$

Діаграма стану поточної енергетичної моделі представлена на рисунку 2.1.

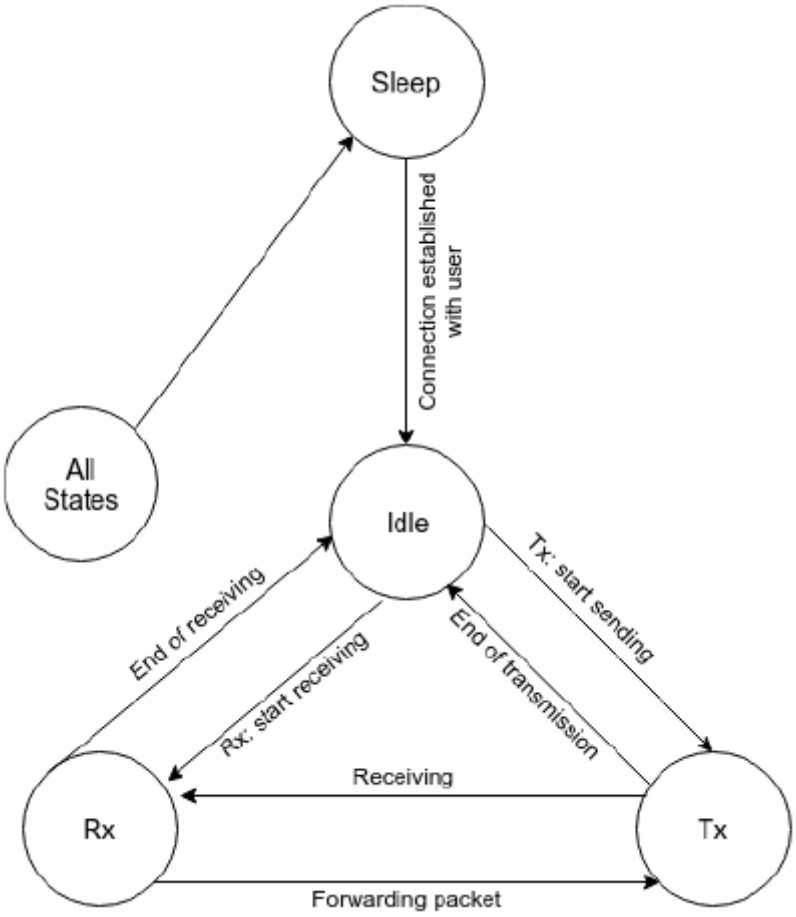

Рисунок 2.1 – Діаграма стану для моделі енергоспоживання

Модель ціноутворення у FogGRASS також пропонує різні моделі ціноутворення. Центральний брокер стежить за використанням ресурсів. Модель ціноутворення, доступна у FogGRASS, наведена в таблиці II. Для моделі ціноутворення брокер відповідає за забезпечення SLA користувача (угода про рівень обслуговування).

Пропонований FogGRASS – це великий симулятор, призначений для підтримки виконання завдань у тумані. Зазвичай ресурси, доступні на рівні туману, менші порівняно з хмарним центром обробки даних.

Тому дуже бажано ефективне використання ресурсів туману. Крім того, запропонована структура розроблена з централізованим диспетчером

завдань, тобто вузлом посередника. Брокер відповідає за управління всіма ресурсами туману. Спочатку пристрій зв'язується з брокером зі своїми обчислювальними засобами. Після успішного виділення вузла туману пристрій безпосередньо надсилає завдання призначеному вузлу. Таким чином, це зменшує накладні витрати на вузлі брокера. Крім того, запропонована робота також підтримує сенсорні пристрої, які можуть лише генерувати дані, а інші пристрої можуть отримувати ці дані через підписку. Решта технічних деталей обговорюються в наступному підрозділі.

2.2 Реалізація архітектури системи моделювання розподілених систем туманних обчислень

FogGRASS розроблено як розширення OMNeT++. OMNet++ широко використовуваний інструмент із відкритим вихідним кодом, який надає велику бібліотеку для моделювання характеристик мережі. Він надає низку вбудованих модулів, які діють як реалістичні мережеві пристрої. На рисунку 2.2 показано дизайн запропонованого фрейворку FogGRASS. У FogGRASS можна легко інтегрувати всі доступні модулі OMNeT++.

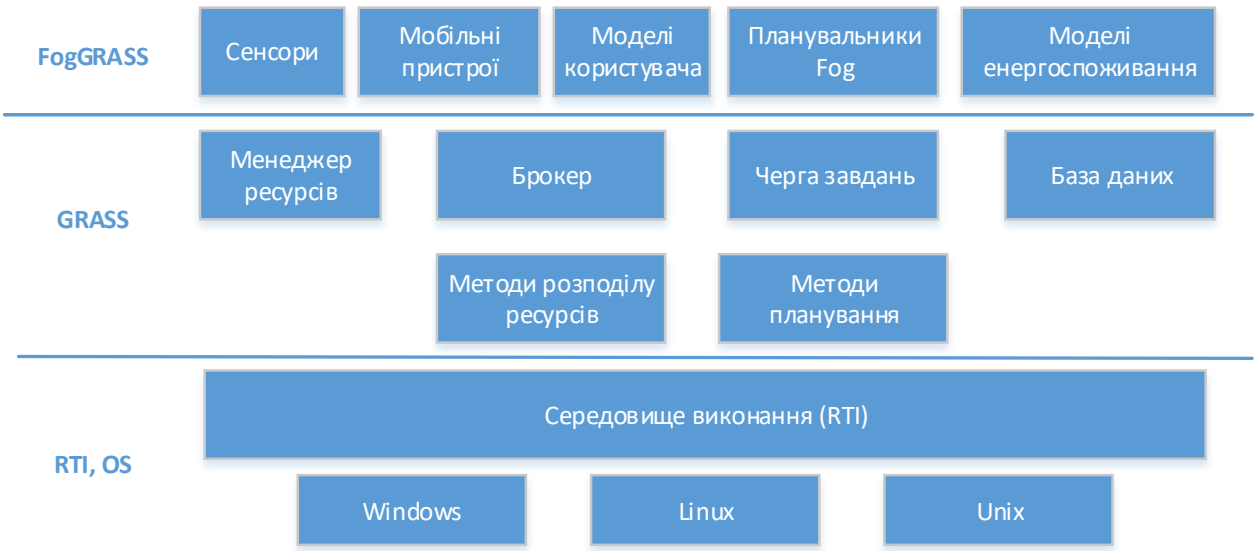

Рисунок 2.2 – Структура фрейворка FogGRASS

Дослідники можуть легко модифікувати модулі FogGRASS для моделювання своїх сценаріїв. Основна мета розробки FogGRASS полягає в тому, що існуючі фреймворки розроблені для підтримки різних датчиків [10,11]. Однак вони проігнорували такі характеристики мережі, як частота помилок і швидкість передачі даних, які можуть відігравати вирішальну роль у моделюванні програм, чутливих до затримок.

Так само дана робота не підтримує мобільність і не надає простих моделей мобільності [41]. Графічний інтерфейс FogGRASS показаний на рисунку 2.3. Запропонований фрейворк представляє повне середовище моделювання, яке включає датчики, вузли туману, вузол посередника та територіально розподілені центри обробки даних. Брокер відповідає за управління туманними вузлами, він отримує послугу, запитує та призначає вузли туману, зважаючи на використання кожного вузла туману. Це нова структура, яка підтримує як мобільні, так і статичні вузли, пристрої та шлюзи. Модульна конструкція FogGRASS показана на рисунку 2.2. FogGRASS складається з двох основних модулів, тобто кінцевих пристроїв і брокера.

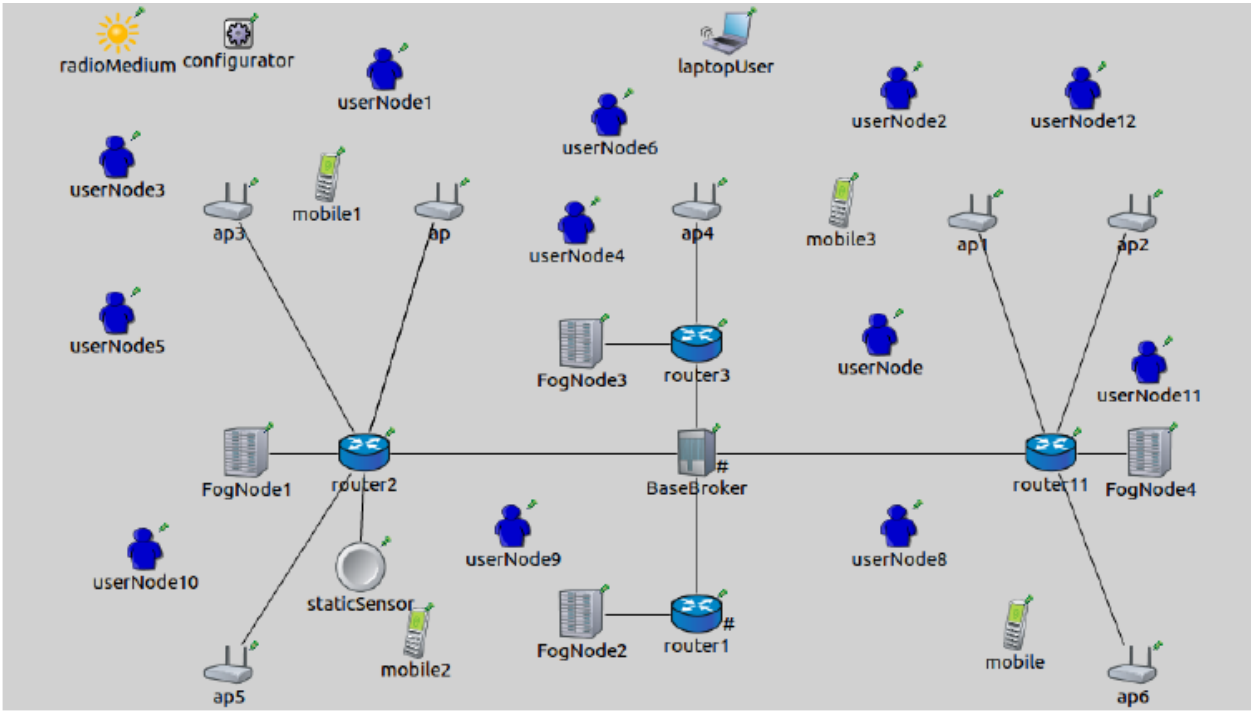

Рисунок 2.3 – Графічний інтерфейс користувача FogGRASS

Основними модулями FogGRASS є брокер, туманний вузол і кінцеві пристрої. Брокер – це централізований менеджер ресурсів, який відстежує всі туманні вузли. Туманний вузол надає послугу обчислень.

Кінцевий пристрій – це фактичний вузол/датчик, який може рухатися під час зв'язку та генерувати запити для обчислювальних вузлів. Вузол-посередник отримує запит від пристроїв на виконання запитуваного обчислення. Вузол посередника підтримує чергу, звідки запити обслуговуються за принципом «першим прийшов – першим обслужено». Таким чином, брокер відповідає за планування, виконання та передачу завдань. Модуль передачі використовується у двох випадках. Перший – коли вузол переміщується з одного регіону в інший і виконується його завдання. У цьому випадку брокер динамічно передає нерозподілене завдання своєму сусідньому вузлу fog; другий – запитане завдання передається на сусідній вузол, де вузол туману доступний або малозавантажений. Останній використовується, коли прямо підключений туманний вузол сильно завантажений і не може виконати запит через великі незавершені завдання, як показано на рисунку 2.4.

Вузли-посередники також підключені до магістральної мережі, яка додатково з'єднує його з хмарним центром обробки даних. Обчислені результати/дані можна надіслати до центру обробки даних для зберігання та проведення аналізу на наступному етапі. Зазвичай існують обмежені туманні вузли, які забезпечують обчислення. Кожен туманний вузол надає послуги великій кількості пристроїв. У типовому випадку він використовується для збору даних із датчиків, а єдина значима інформація надсилається до центру обробки даних. У запропонованому FogGRASS дослідники можуть включити власні алгоритми планування запитів на вузлі посередника для підтримки виконання завдань на вузлі туману. Крім того, складний алгоритм хэндовера також можна дуже легко включити. Це може запровадити новий напрямок досліджень у симуляторах туману. У всіх попередніх симуляторах відсутні

такі алгоритми планування, які дозволяють досліднику ефективно використовувати туманні вузли. Мережевий модуль у FogGRASS забезпечує повні властивості фізичного/бездротового з'єднання мережі, такі як скидання пакетів, повторна передача, частота бітових помилок і пропускна здатність з'єднання.

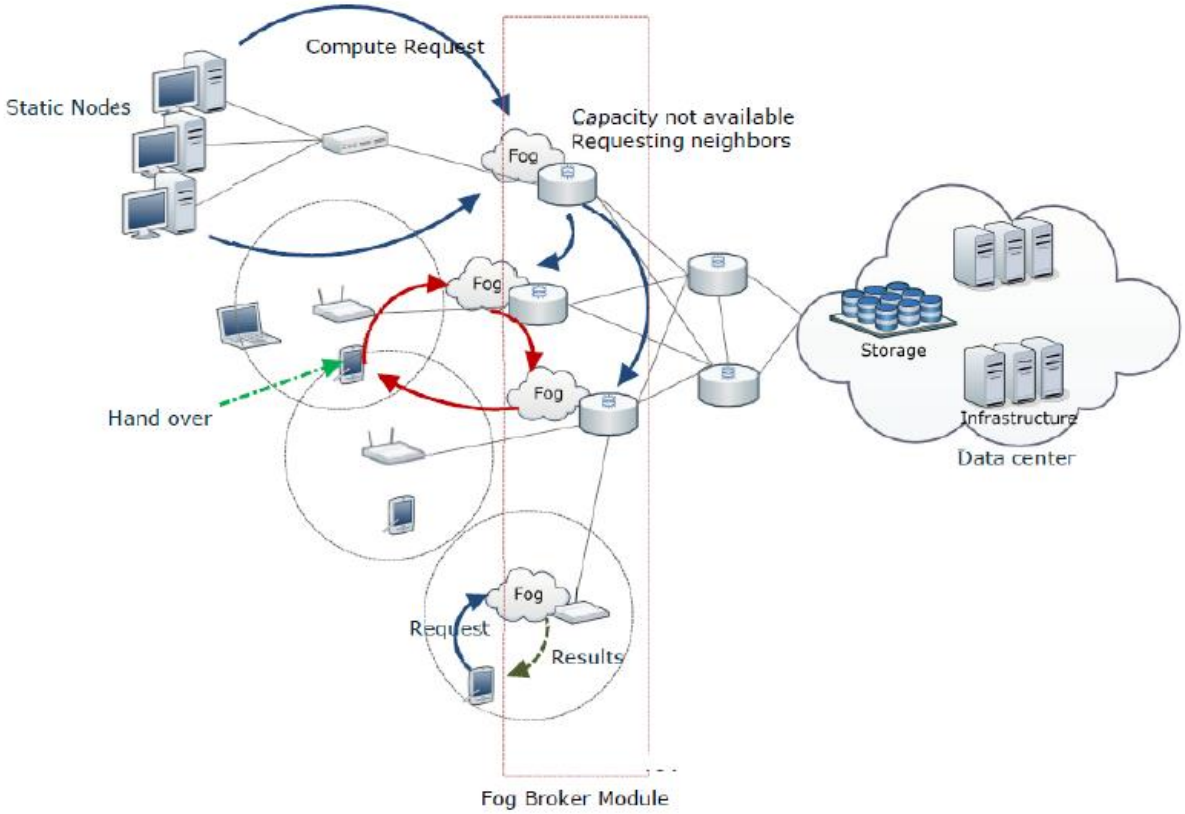

Рисунок 2.4 – Архітектурна модель FogGRASS

FogGRASS забезпечує гнучкість конфігурації мережі. Користувач може визначати параметри мережі відповідно до своїх проектних вимог. Таким чином, можна змоделювати більш реалістичну мережу. FogGRASS містить низку протоколів зв'язку, які можна використовувати для моделювання різноманітних сценаріїв. Наразі доступними протоколами є TCP, UDP, FTP, HTTP, MQTT, CoAP і AMPQ. HTTP, TCP і UDP також можна використовувати для зв'язку з центром обробки даних. FogGRASS підтримує різнорідні пристрої та туманні вузли зі змінною кількістю додатків, які можуть виконуватися на кожному вузлі одночасно. Крім того, програми

можуть використовувати різні протоколи зв'язку. Крім того, він також підтримує IPv4 і IPv6; однак обидва не можна використовувати одночасно. Брокер можна використовувати для відправки даних на запитані пристрої; отже, механізм публікації, підписки реалізується на вузлі брокера. Будь-який пристрій може зареєструватися у брокера як передплатник, видавець або обидва.

Таким чином, брокер відстежує місцезнаходження пристрою та ділиться оновленнями з усіма пристроями абонента. Цей механізм додано для підтримки ролі різних датчиків в IoT. Для цієї мети MQTT є легким протоколом обміну повідомленнями на основі публікації/підписки3, який реалізований у FogGRASS для зв’язку між пристроями та вузлами-брокерами.

Мережа Fog приладів складається з площини керування та площини даних. Наприклад, на площині даних туманне обчислення дозволяє обчислювальним службам розташовуватися на межі мережі на відміну від серверів у центрі обробки даних. Порівняно з хмарними обчисленнями, туманні обчислення підкреслюють наближеність до цілей кінцевих користувачів і клієнтів (наприклад, операційні витрати, політика безпеки, використання ресурсів), щільний географічний розподіл і усвідомлення контексту (що стосується обчислювальних ресурсів і ресурсів Інтернету речей), зменшення затримки і економія магістральної пропускної здатності для досягнення кращої якості обслуговування (QoS)[8] і периферійної аналітики/потокового аналізу, що забезпечує чудову взаємодію з користувачем[9] і резервування в разі збою, а також його можна використовувати в сценаріях.

Мережа Fog підтримує концепцію Інтернету речей (IoT), згідно з якою більшість пристроїв, що використовуються людьми щодня, будуть підключені один до одного. Приклади включають телефони, переносні пристрої для моніторингу стану здоров’я, підключені транспортні засоби та доповнену реальність за допомогою таких пристроїв, як Google Glass.

Пристрої IoT часто мають обмежені ресурси та обмежені обчислювальні можливості для виконання криптографічних обчислень. Туманний вузол може забезпечити безпеку для пристроїв IoT, виконуючи натомість ці криптографічні обчислення або шифрування/дешифрування даних, що передаються між вузлами.

2.3 Опис основних модулів системи

Модуль брокера. Як обговорювалося вище, вузол посередника відповідає за надання ресурсів на запит. FogGRASS ми класифікували вузли на три типи, тобто статичні, на основі мобільності та на основі бездротового доступу (WAB). Статичні вузли – це сервери туманних обчислень, розміщені на шлюзі для забезпечення обчислень за запитом. Мобільні вузли – це фактичні мобільні вузли, які можуть запитувати обчислення від туманних вузлів.

WAB — це точки доступу, які можна використовувати для обслуговування підключених вузлів. Внутрішні модулі вузла брокера показано на рисунку 2.5. WAB розроблено за технологіями Cisco Edge маршрутизаторів, які забезпечують обчислення разом із маршрутизацією. Інші функції посередницького вузла полягають у реєстрації/керуванні видавцями, передплатниками, плануванні туманних вузлів для запитуваних пристроїв, керування ресурсами, оптимальне використання ресурсів серед кількості запитуваних пристроїв, керування передачами, забезпечення зв’язку з центрами обробки даних та надійні дані доставка. Виконання вузла брокера та туману наведено в Алгоритмах 1 та 2 у додатку А. Була використана псевдомова програмування для того, щоб відобразити основні етами розроблених методів. Розділення алгоритмів говорить про те, що їх можливо застосовувати як паралельно, так і використовуючи розподілені технології програмування.

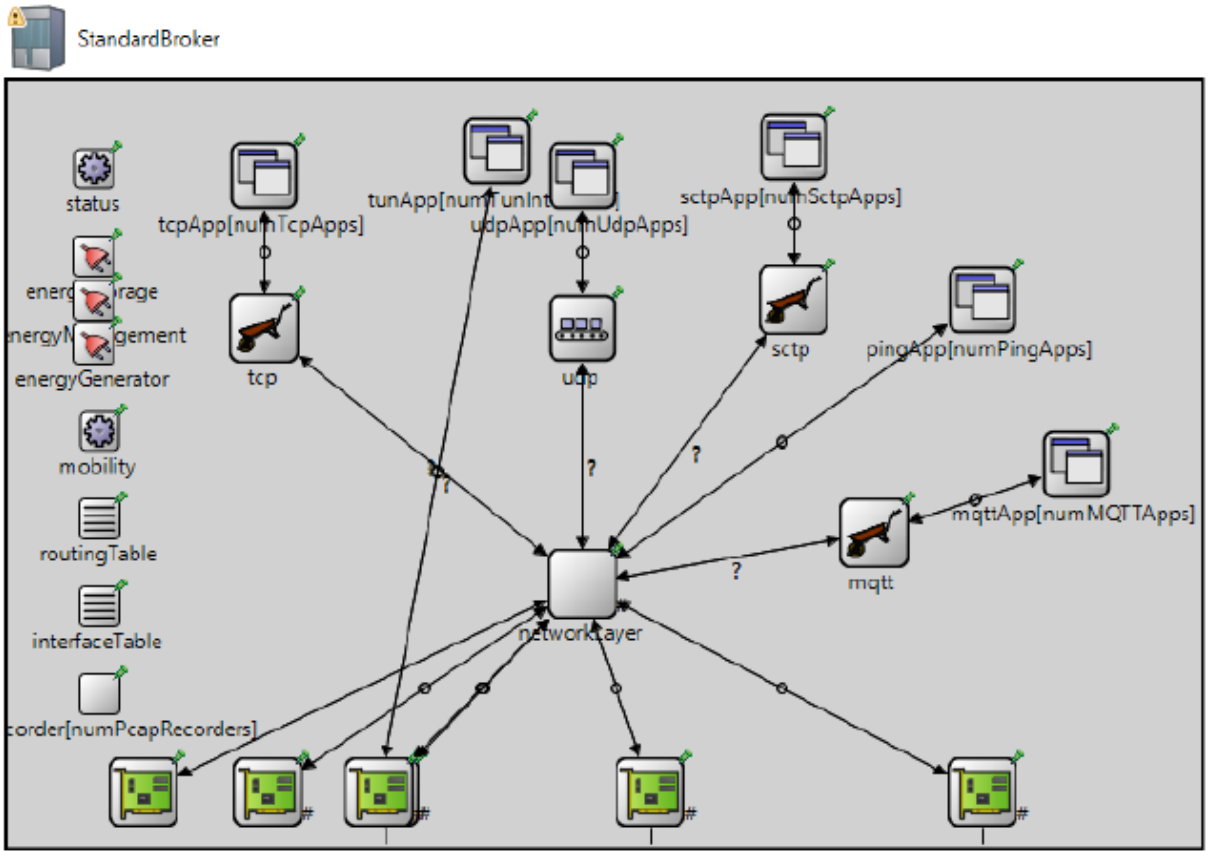

Рисунок 2.5 – Внутрішній модульний вигляд вузла брокеру

Пристрої кінцевого вузла. Іншим основним модулем FogGRASS є мобільні пристрої, які можуть відігравати важливу роль у обчисленні туману. Тут ми класифікуємо пристрої як вузли датчиків і вузли користувача. Сенсорні вузли діють як генератор даних і надсилають згенеровані дані брокеру або іншим пристроям через регулярний проміжок часу. Тоді як вузол користувача може генерувати або отримувати дані. Вузол користувача також може діяти як датчик. Більше того, ці вузли поділяються на два типи: дротові та бездротові вузли. Вузли можуть бути статичними або мобільними, обидві версії підтримуються у FogGRASS. Пристрої можуть реєструватися для публікації даних. Дані надсилаються до вузла брокера. У пропонованому симуляторі всі пристрої мають IP- протокол.

Пристрої підтримують протокол MQTT разом з іншими протоколами додатків, такими як HTTP, TCP, FTP, SMTP, SNMP і UDP. Крім того, вибір протоколу для зв'язку здійснюється під час розробки. Завдяки гнучкій

архітектурі FogGRASS дослідники можуть легко включати інші протоколи за допомогою чітко визначених класів. Кінцеві пристрої в основному використовуються для створення або споживання даних. Пристрій використовує повідомлення MQTT для зв’язку з вузлом-брокером. FogGRASS підтримує різнорідні пристрої; кожен пристрій може мати різні функції, такі як швидкість передачі даних, функціональність тощо.

Моделі мобільності. Мобільність — ще одна дуже важлива особливість кінцевих пристроїв. У FogGRASS підтримуються бездротові та дротові пристрої. Мобільність може відігравати важливу роль, вона може створити нові виклики для дослідників у формі оптимального використання ресурсів і передачі ресурсів.

Зазвичай моделі мобільності класифікуються на дві категорії, тобто модель сутності та модель групи. У моделі сутності рух кожного вузла не залежить від інших вузлів. Найпоширенішими моделями об’єктів є випадкові маршрутні точки, Гаусс-Марков та розділ міста. У груповій моделі рух одного вузла залежить від руху інших вузлів. Найбільш часто використовувані групові моделі – це мобільність колони, кочова спільнота та група контрольної точки. FogGRASS підтримує лише моделі сутностей. Вже наявні моделі є випадкова маршрутна точка, масова мобільність, мобільність Гаусса Маркова, мобільність Чейнга, CircleMobility, LinearMobility та мобільність транспортного засобу.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Опис експериментів

FogGRASS пропонує комплексну платформу для моделювання різноманітних застосувань туману. Це також допомагає зрозуміти основну концепцію туманних обчислень. FogGRASS забезпечує багату конфігурацію мережі, керовану через мережевий модуль. Це дозволяє моделювати реалістичне мережеве середовище, яке відкриває нові виклики для дослідників, такі як використання ресурсів, частота помилок даних і передача даних. Усі модулі налаштовуються за допомогою файлу конфігурації з розширенням *.cnf. Користувач може встановити значення різних параметрів, таких як кількість брокерів, туманних вузлів, кінцевих пристроїв, швидкість передачі даних, шум каналу та моделі мобільності для кожного окремого користувача або групи вузлів.

Для оцінки продуктивності симулятора моделюється сценарій комунікаційної мережі на основі інтелектуальної системи керування трафіком. Інтелектуальне керування трафіком є ідеальним випадком для туманних вузлів, де розгортається кілька статичних датчиків для збору інформації про дорожній рух. Крім того, пішоходи також можуть ділитися навколишніми умовами за допомогою своїх розумних пристроїв. Пішоходи рухливі, можуть вільно рухатися в будь-якому напрямку. Ці датчики генерують дані в окремих точках. Дані обробляються на вузлі туману для швидкого реагування, виявлення будь-яких порушень правил, таких як перевищення швидкості або будь-яка незвичайна ситуація. Крім того, є користувачі, які підписалися на ці датчики залежно від їхнього географічного розташування. Туманні вузли використовуються для виявлення порушень на основі отриманих даних і розповсюдження результатів абонентам. Крім

цього, користувачі також можуть генерувати запит на обчислювальні ресурси для здійснення пошуку чи прогнозування.

Було проаналізовано мережу на основі вищезгаданого сценарію за допомогою FogGRASS. Змодельована топологія показана на рисунку 3.1.

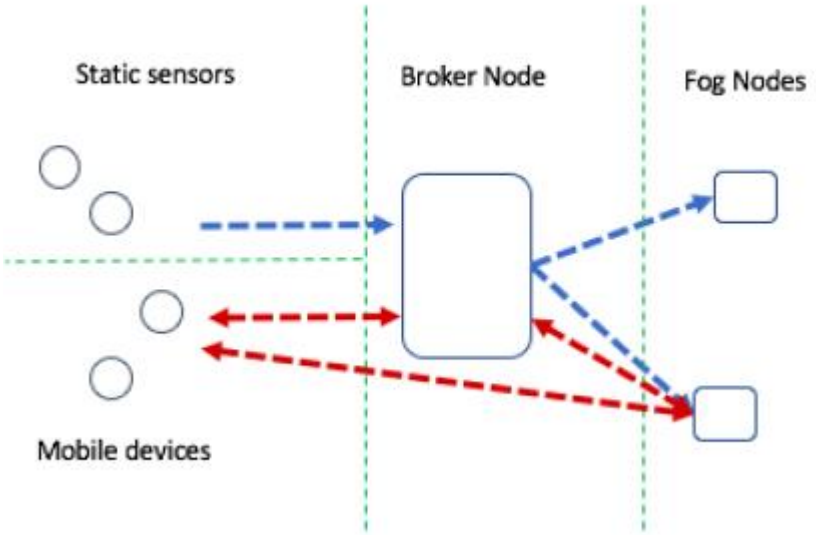

Рисунок 3.1 – Приклад змодельованої топології

Датчик генерує дані через рівні проміжки часу. Вузол- посередник отримує дані та передає їх на вузол fog для обробки, а результати розповсюджуються на передплачені пристрої. Крім того, пристрої можуть надсилати запит на обчислювальну потужність вузлу-посереднику. Посередник призначає ресурси на основі політики планування та після виконання надсилає результат на запитаний пристрій. Під час виконання завдання брокер також відстежує запитані або підписані пристрої для операцій передачі.

Масштабованість запропонованого FogGRASS вимірюється з точки зору використання пам’яті та ЦП. Тому, оскільки IoT складається з великої кількості пристроїв, важливо порівняти запропонований FogGRASS з точки зору використання пам’яті та ЦП. Крім того, інші параметри мережі, які тут перевіряються, це затримка, затримка та частота помилок. Затримка також вимірюється щодо завдання виконання. Завдання поділяються на великі

(1500 MIPS), середні (900 MIPS), малі (200 MIPS) і випадкові (випадкове число від 200 до 1500 MIPS). Зазвичай різні функції вимагають різної обчислювальної потужності. Наприклад, у системі керування трафіком ідентифікація об’єктів із моментального знімка або прогнозування руху групи можуть займати різний час виконання. Технічні характеристики системи наведено в таблиці 31. Конфігурацію можна при необхідності змінювати конфігураційним файлом.

Таблиця 3.1 – Конфігурація експериментальної системи

Параметр системи	Значення
Кількість центральних процесорів	8
Кількість ядер	4
Кількість потоків на ядро	2
Частота процесора	1300Мгц
Пам'ять	16Гб

Параметри, які використовуються в моделюванні, наведені у таблиці 3.2.

Таблиця 3.2 – Параметри моделювання

Параметр системи	Значення
Брокери	До 500
Бездротовы датчики	100-1000
Точки доступу	100-500
Пристрої	100-1500
Хмарні датацентри	1-8
Мобильні пристрої	100-1500
Туманні вузли	10-800
Швидкість мережі	10 Гбіт/с

3.2 Результати експериментів

Пропонований FogGRASS оцінюється шляхом зміни кількості пристроїв. На рисунку 3.2 показано, що використання ЦП по відношенню до загальної кількості модулів демонструє тенденцію до зростання; однак приблизно для 1320 вузлів використання ЦП становить менше 25%. Кількість вузлів на рисунку, включає обидва пристрої, туманні вузли та вузол-посередник.

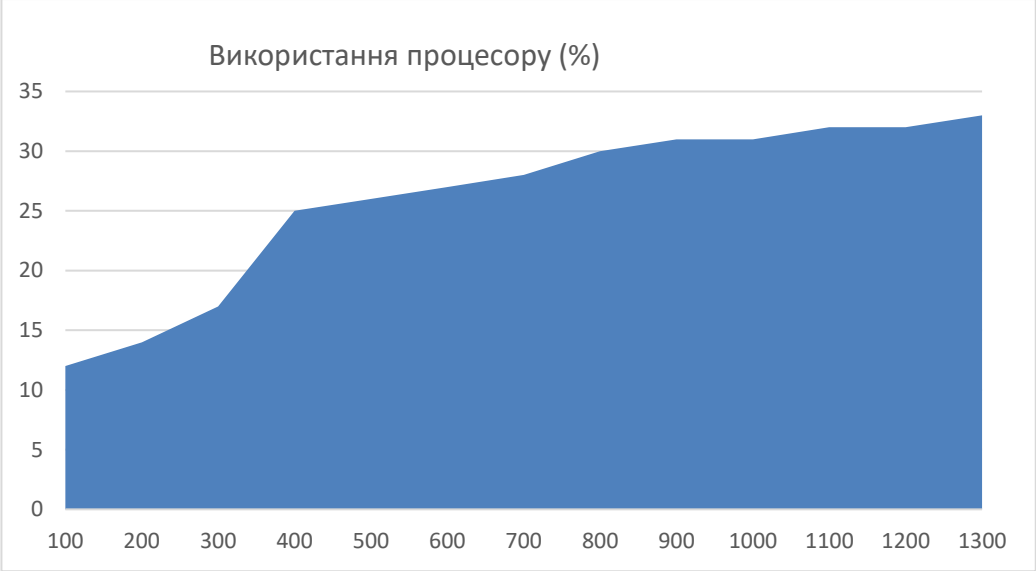

Рисунок 3.2 – Використання ЦП FogGRASS щодо кількості вузлів

Так само, зі збільшенням кількості вузлів, споживання пам’яті також демонструє тенденцію до зростання. Однак для 1320 вузлів приблизно використовується лише 15% пам’яті (рисунок 3.3). Таким чином, це чітко показує, що запропонований симулятор може легко підтримувати виконання тисяч вузлів на типовій обчислювальній машині. Це пов’язано насамперед з тієї причини, що кожен вузол зберігає менш специфічні для вузла дані. У дизайні симулятора загальні дані розподіляються між усіма вузлами.

FogGRASS розроблено на основі OMNeT++, що забезпечує багатий графічний інтерфейс користувача. Таким чином, збільшення кількості вузлів зазвичай вимагає значного часу для заповнення GUI. Отже, затримка в

побудові графічного інтерфейсу також є важливим параметром, на який слід звернути увагу.

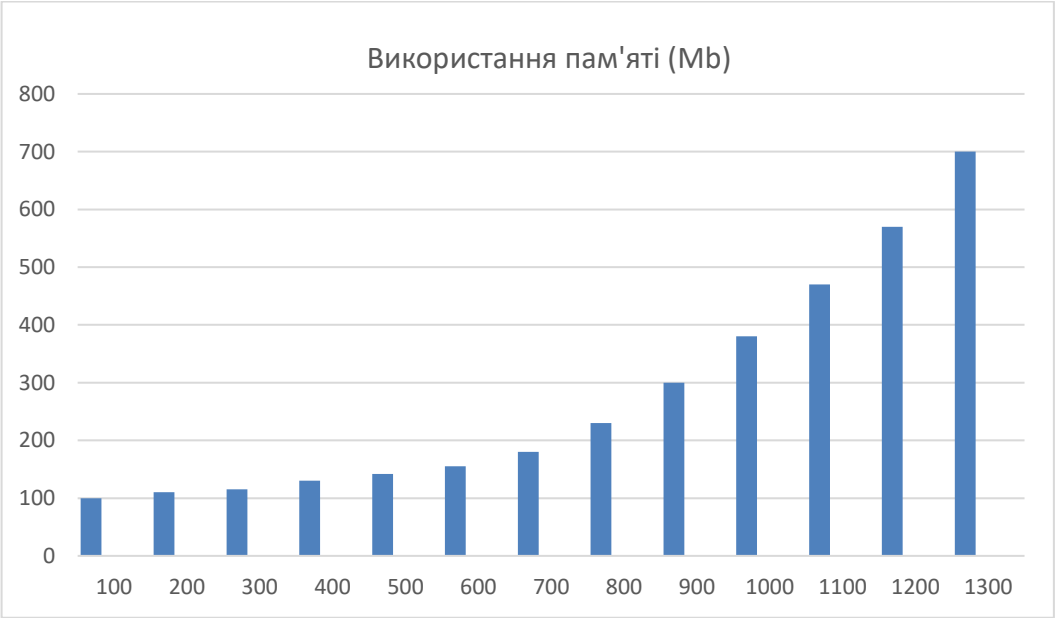

Рисунок 3.3 – Використання пам’яті FogGRASS від кількості вузлів

На рисунку 3.4 показано затримку в побудові графічного інтерфейсу користувача (GUI).

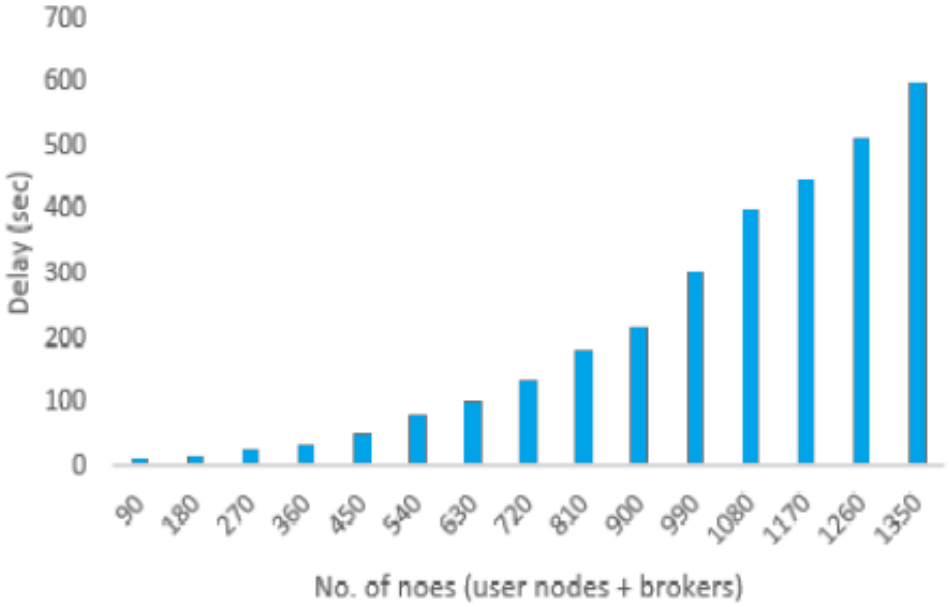

Рисунок 3.4 – Затримка у створенні розширеного графічного інтерфейсу

На 1320 вузлах максимальна затримка становить близько 600 секунд. Це завдяки багатому графічному інтерфейсу, який дозволяє користувачам перевіряти кожен модуль і параметри через графічний інтерфейс. Однак запропонований симулятор також може працювати в режимі командного рядка, щоб уникнути таких затримок.

На рисунку 3.5 показано відсоток передач, виконаних під час оцінювання. Однак можна побачити, що зі збільшеною довжиною завдання є більші шанси для операції передачі. Крім того, розташування пристрою та модель мобільності також відігравали важливу роль у передачі. Таким чином, вузол-посередник зберігає розташування запитуваного вузла, щоб після завершення завдання результати могли бути відправлені відповідним чином.

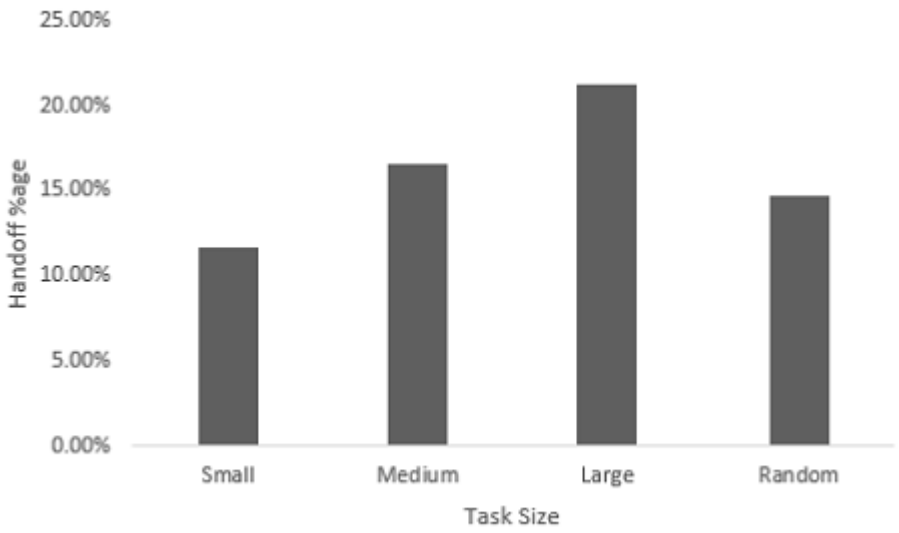

Рисунок 3.5 – Handoff, виконане залежно від розміру завдання

Наскрізна затримка збільшується зі збільшенням кількості користувальницьких вузлів, як показано на рисунку 3.6. Тут визначається наскрізна затримка як час, необхідний запиту від пристрою до вузла брокера. У типовому сценарії, коли велика кількість пристроїв генерує запити через регулярні проміжки часу, це призводить до більшої затримки, як показано на рисунку.

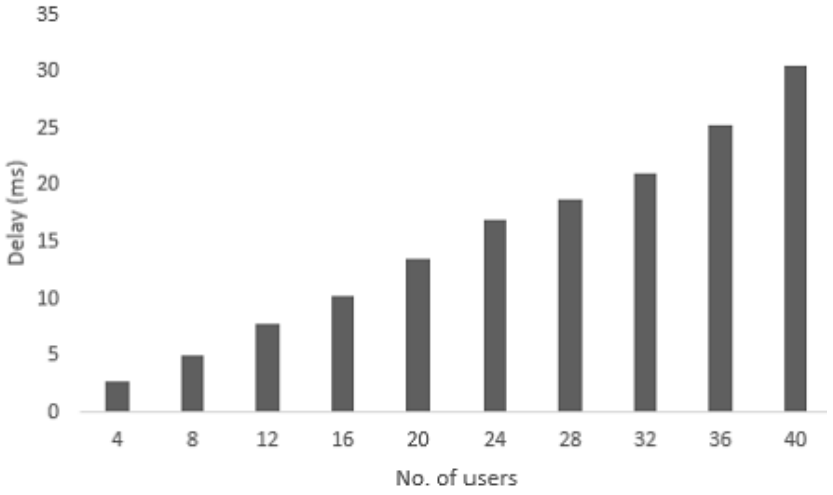

Рисунок 3.6 – Середня затримка відносно розміру завдання

В експериментах результати збираються на основі різних вимог до обчислення завдань. Завдання поділяються на малі, середні, великі та змішані. На рисунку 3.7 показано середню затримку виконання завдання в кожній категорії завдань.

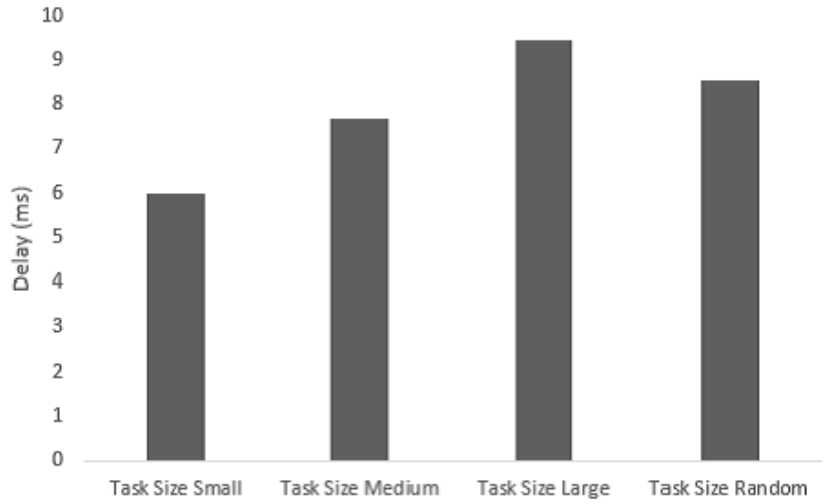

Рисунок 3.7 – Загальна затримка на основі вимог до обчислювальної потужності

На рисунку чітко видно, що виконання великого завдання займає значну кількість часу. Таким чином, невеликі завдання виконання можуть підвищити продуктивність системи та скоротити час очікування на туманних вузлах.

На рисунку 3.8 показано середній рівень помилок, повідомлений під час виконання. Пристрої спілкуються за допомогою бездротового каналу. Однак швидкість залежить від розміщення пристрою, перевантаженості мережі та моделі мобільності, яка використовується для вузла.

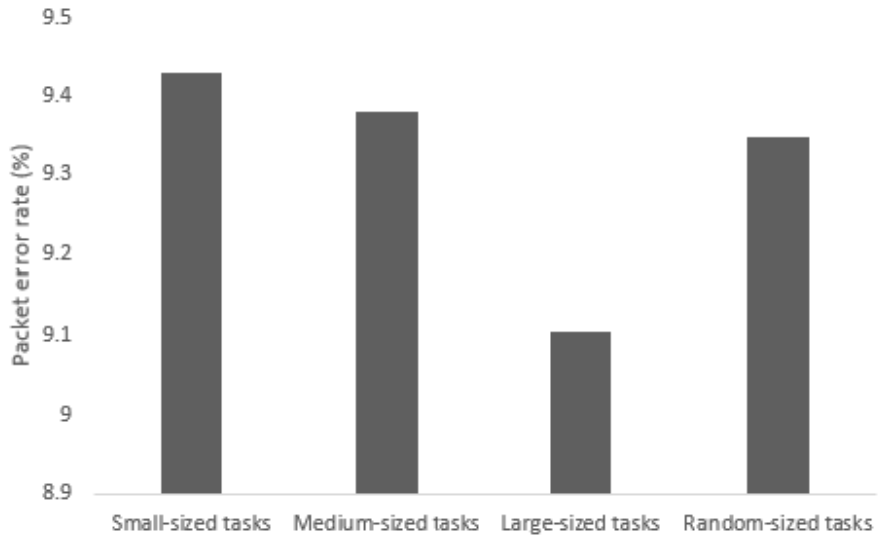

Рисунок 3.8 – Середня частота помилок бездротового зв’язку, повідомлена під час виконання

На рисунку 3.9 показано залишкову енергію трьох мобільних користувачів.

Рисунок 3.9 – Вектор залишкової енергії протягом часу моделювання

Залишкова енергія визначається як енергія, що залишилася на мобільному вузлі після зв'язку з брокером/вузлом туману. FogGRASS забезпечує розширений спосіб вимірювання енергії кожного мобільного вузла, який брав участь у моделюванні. Вузол більше не стає частиною мережі, коли його залишкова енергія досягає нуля.

На рисунку 3.10 показано споживання енергії відносно потужності передачі.

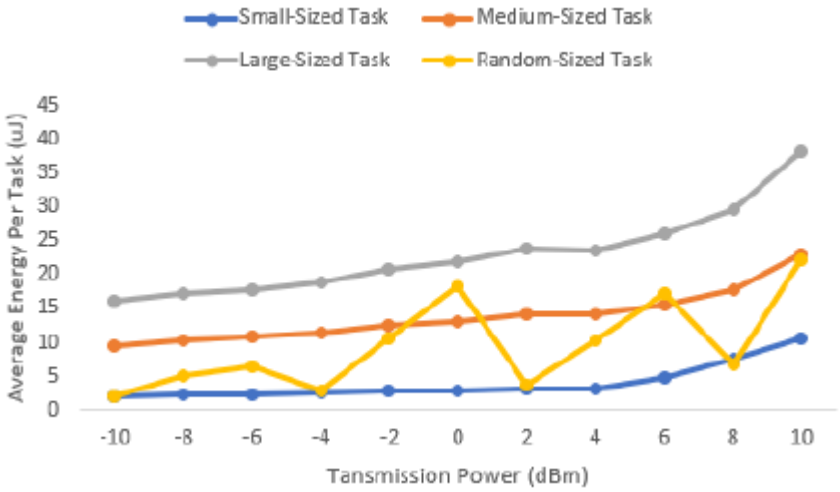

Рисунок 3.10 – Середнє споживання енергії та потужність передачі

У FogGRASS мобільні вузли можуть динамічно змінювати свою потужність передачі для зв'язку з брокером/вузлом туману. З іншого боку, зміна потужності передачі може безпосередньо вплинути на термін служби пристроїв, що працюють від батареї. При великій потужності передачі пристрій може працювати менший період часу.

ВИСНОВКИ

У зв'язку з розвитком інформаційних технологій і зростанням використання інтернету, з часом технологія Інтернет речей стала широко використовуватися в багатьох областях. Швидке зростання інтересу до технології Інтернет речей дає нам підстави вважати, що ця технологія в найближчому майбутньому стане невід'ємною частиною нашого повсякденного життя. Через те, що фізичні ресурси розумних пристроїв, що працюють на основі технології Інтернету речей, мають низку обмежень, вони були інтегровані в хмарні технології, проте в той же час було виявлено низку недоліків хмарних технологій для таких пристроїв. Ці недоліки призвели до появи технології “fog computing”. Ця технологія не заміняє хмару повністю, вона просто діє як "посередник" між хмарою і кінцевим користувачем. Прогнозується, що в найближчому майбутньому попит на цю технологію сильно зростатиме.

Поява туманних і периферійних обчислень відкриває нові можливості для програм, чутливих до затримки. Надання обчислень на межі мережі, ближче до кінцевого користувача, зменшує традиційні затримки мережевого зв’язку порівняно з хмарною моделлю виконання. У цій роботі ми запропонували симулятор туманної мережі [35].

FogGRASS надає дослідникам основу для дослідження власних методів управління ресурсами, виконання передачі ресурсів, впровадження нових алгоритмів із повним мережевим стеком. Усі існуючі симулятори не повністю забезпечують характеристики мережі, функції мобільності та модулі управління ресурсами. FogGRASS надає різноманітні мережеві функції та підтримує велику кількість моделей мобільності. Таким чином, наша мета полягає в тому, щоб полегшити дослідникам швидку розробку та тестування нових алгоритмів управління ресурсами [36].

В роботі ми порівняно FogGRASS з точки зору використання мережі та поведінки щодо збільшення трафіку. Оцінки ефективності ілюструють ефективність запропонованої системи. У майбутньому можливо розширити FogGRASS, включив міграцію VM між туманними вузлами. Крім того, сумісність між туманними федераціями є ще однією цікавою сферою, яка потребує подальшого дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Shi and S. Dustdar, “The promise of edge computing,” Computer, vol. 49, no. 5, pp. 78–81, 2016.

2. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012, pp. 13–16.

3. P. Bellavista, L. Foschini, and D. Scotece, “Converging mobile edge computing, fog computing, and IoT quality requirements,” in Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on. IEEE, 2017, pp. 313–320.

4. S. N. Han, G. M. Lee, N. Crespi, N. Van Luong, K. Heo, M. Brut, and P. Gatellier, “Dpwsim: A simulation toolkit for iot applications using devices profile for web services,” in Internet of Things (WF-IoT), 2014 IEEE World Forum on. IEEE, 2014, pp. 544–547.

5. S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee, “Towards simulating the internet of things,” in Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on. IEEE, 2014, pp. 444–448.

6. C. Sonmez, A. Ozgovde, and C. Ersoy, “Edgecloudsim: An environment for performance evaluation of edge computing systems,” in Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on. IEEE, 2017, pp. 39–44.

7. A. M. Khan, L. Navarro, L. Sharifi, and L. Veiga, “Clouds of small things: Provisioning infrastructure-as-a-service from within community networks,” in Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on. IEEE, 2013, pp. 16–21.

8. T. Pflanzner, A. Kert´esz, B. Spinnewyn, and S. Latr´e, “Mobiotsim: towards a mobile iot device simulator,” in 2016 IEEE 4th International Conference

on Future Internet of Things and Cloud Workshops (Fi- CloudW). IEEE, 2016, pp. 21–27.

9. C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. Sorger, P. Bouvry, B. Kantarci, F. Granelli, and S. Giordano, “CrowdsenSim: a simulation platform for mobile crowdsensing in realistic urban environments,” IEEE Access, vol. 5, pp. 3490–3503, 2017.

10. “Parse,” https://parse.com/products/iot, accessed: 2018-08-10.

11. “Google cloud platform,” https://cloud.google.com/solutions/iot/, accessed: 2018-08-10.

12. H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, “ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments,” Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, 2017.

13. C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele et al., “Fit iot-lab: A large scale open experimental iot testbed,” in Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. IEEE, 2015, pp. 459–464.

14. K. Dolui and S. K. Datta, “Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing,” in Global Internet of Things Summit (GIoTS), 2017. IEEE, 2017, pp. 1–6.

15. “Contiki cooja,” http://www.contiki-os.org/start.html, accessed: 2018-08-10.

16. A. Brogi and S. Forti, “Qos-aware deployment of iot applications through the fog,” IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1185–1192, Oct 2017.

17. J. Byrne, S. Svorobej, A. Gourinovitch, D. M. Elango, P. Liston, P. J. Byrne, and T. Lynn, “Recap simulator: Simulation of cloud/edge/fog computing scenarios,” in 2017 Winter Simulation Conference (WSC), Dec 2017, pp. 4568–4569.

18. R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran,

"Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures," in 2017 IEEE Fog World Congress (FWC), Oct 2017, pp. 1–6.

19. N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for internet of things computations," in 2016 Cloudification of the Internet of Things (CIoT), Nov 2016, pp. 1–6.

20. K. Hong, D. J. Lillethun, U. Ramachandran, B. Ottenw¨alder, and B. Koldehofe, "Mobile fog: a programming model for large-scale applications on the internet of things," in MCC@SIGCOMM, 2013.

21. S. Conti, G. Faraci, R. Nicolosi, S. A. Rizzo, and G. Schembra, "Battery management in a green fog-computing node: a reinforcement-learning approach," IEEE Access, vol. 5, pp. 21 126–21 138, 2017.

22. H. A. M. Name, F. O. Oladipo, and E. Ariwa, "User mobility and resource scheduling and management in fog computing to support iot devices," in Innovative Computing Technology (INTECH), 2017 Seventh International Conference on. IEEE, 2017, pp. 191–196.

23. X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," IEEE Access, vol. 5, pp. 21 355–21 367, 2017.

24. L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1216–1228, 2017.

25. A. Carrega, M. Repetto, P. Gouvas, and A. Zafeiropoulos, "A middleware for mobile edge computing," IEEE Cloud Computing, vol. 4, no. 4, pp. 26–37, 2017.

26. J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 3, pp. 361–373, 2017.

27. G. Ananthanarayanan, P. Bahl, P. Bod´ık, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," computer, vol. 50, no. 10, pp. 58–67, 2017.

28. J. Ren, H. Guo, C. Xu, and Y. Zhang, “Serving at the edge: A scalable iot architecture based on transparent computing,” IEEE Network, vol. 31, no. 5, pp. 96–105, 2017.

29. X. Gong, L. Guo, G. Shen, and G. Tian, “Virtual network embedding for collaborative edge computing in optical-wireless networks,” Journal of Lightwave Technology, vol. 35, no. 18, pp. 3980–3990, 2017.

30. X. Xu, J. Liu, and X. Tao, “Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation,”

31. IEEE Access, vol. 5, pp. 16 406–16 415, 2017.

32. X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, “Exploiting massive d2d collaboration for energy-efficient mobile edge computing,” IEEE Wireless Communications, vol. 24, no. 4, pp. 64–71, 2017.

33. B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang, “Incorporating intelligence in fog computing for big data analysis in smart cities,” IEEE Transactions on Industrial informatics, vol. 13, no. 5, pp. 2140–2150, 2017.

34. T. Li, Y. Liu, L. Gao, and A. Liu, “A cooperative-based model for smartsensing tasks in fog computing,” IEEE access, vol. 5, pp. 21 296–21 311, 2017.

35. Volk Maksym, Buhrii Andrii, Kovtun Evgenii, Zhuravel Denys, Zborovskyi Mykhailo. Simulation and management of fog computing for IoT. 7-th International Scientific and Technical Conference "COMPUTER AND INFORMATION SYSTEMS AND TECHNOLOGIES". Kharkiv: NURE. – 2024. − P. 11-12

36. Волк М.О., Бугрій А.М., Брестовицький Р.М., Соробей Б.В., Журавель Д.С. Моделі та методи ефективного управління ресурсами в системах хмарних обчислень. Проблеми інформатизації: Матеріали дванадцятої міжнародної науково-технічної конференції. –Баку – Харків – Бельсько-Бяла , 21 – 22 листопада 2024 року. С.49. doi: https://doi.org/10.32620/PI.24.t2