ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення автоматизованих систем

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: «**ЗАСТОСУВАННЯ ХМАРНОЇ МЕРЕЖЕВОЇ ВІРТУАЛІЗАЦІЯ ДЛЯ ДИНАМІЧНОГО ПІДКЛЮЧЕННЯ ПРИСТРОЇВ В МЕРЕЖІ ІОТ»**

Виконав: студент 6 курсу, групи ІСДМ-61
спеціальності 126 Інформаційні системи та технології
освітня програма «Інформаційні системи та технології»
(шифр і назва спеціальності)
Пешков І.О.
(прізвище та ініціали)
Керівник Полоневич О.В.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)
Нормоконтроль
(прізвище та ініціали)

Київ – 2023

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти - «Магістр»
Спеціальність підготовки 126 Інформаційні системи та технології
Освітня програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
_____ К.П.Сторчак
"____" _____2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Пешков Іван Олексійович
(прізвище, ім'я, по батькові)

1.Тема роботи: «Застосування хмарної мережевої віртуалізація для динамічного підключення пристроїв в мережі IoT»
Керівник роботи: Полоневич Ольга Володимирівна, к.т.н., доцент, доцент кафедри ІПЗАС.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від _____року №
2. Строк подання студентом роботи _____
3. Вхідні дані до роботи :
1. Науково-технічна література
2. Існуючі методології проектування IoT

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналіз основних положень промислового інтернету речей.
2. Дослідження архітектури мереж IoT.
3. Розробка моделі системи IoT з використанням мережевої віртуалізації

5. Перелік графічного матеріалу
1. Титульний слайд
2. Постановка завдання
3. Основні технології проектування IoT
4. Оптимізація мережі за допомогою мережевої віртуалізації
5. Розробка та моделювання системи IoT
6. Тестування запропонованого методу

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Приміт ка
1	Підбір науково-технічної літератури		
2	Вивчення матеріалів для подальшої взаємодії з ними		
3	Аналіз основних положень промислового інтернету речей.		
4	Дослідження архітектури мереж IoT.		
5	Розробка моделі системи IoT з використанням мережевої віртуалізації		
6	Тестування запропонованого методу		
7	Вступ, висновки, реферат		
8	Розробка демонстраційних матеріалів		
9	Попередній захист роботи		

Студент_____ Пешков І.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Полоневич О.В..
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи роботи 65 с., 25 рис., 28 джерел.

Мета роботи - аналіз ефективності застосування хмарної мережевої віртуалізації для динамічного підключення пристроїв в мережі IoT.

Об`єкт дослідження – оптимізація систем IoT за допомогою технології мережевої віртуалізації.

Предмет дослідження – системи IoT.

Методи дослідження – методи теорії інформації, методи багатокритеріальної оптимізації, методи оптимального управління.

У роботі досліджено концепцію побудови динамічної віртуальної мережі в хмарному середовищі серед підключених пристроїв IoT. Ключова ідея полягає в тому, щоб забезпечити механізм побудови віртуальної мережі між підключеними пристроями IoT з різних доменів через відповідні віртуальні об'єкти в хмарному середовищі. Це сприятиме спільному використанню ресурсів і швидкому розвитку різноманітних додатків на рівні віртуалізації шляхом встановлення динамічного наскрізного з'єднання між пристроями IoT.

У роботі запропоновано детальний дизайн системи для побудови віртуальної мережі IoT. Також реалізовано протоколи прикладних рівнів у OMNET++ для моделювання мережі віртуальних об'єктів, щоб провести аналіз продуктивності запропонованої мережі IoT.

Галузь використання – сучасні системи телекомунікацій України.

ІНТЕРНЕТ РЕЧЕЙ, ХМАРНІ ТЕХНОЛОГІЇ, ТЕХНОЛОГІЯ ВІРТУАЛІЗАЦІЇ, БЕЗДРОТОВІ СЕНСОРНІ МЕРЕЖІ, ВІРТУАЛІЗАЦІЯ ЗА ДОПОМОГОЮ ХМАРНИХ ТЕХНОЛОГІЙ.

ЗМІСТ

ВСТУП

Актуальність. Інтернет речей (IoT) вважається однією з революційних технологій майбутнього і привернув велику увагу. Пристрої IoT — це сенсорні або активні пристрої, прикріплені до предметів повсякденного життя, здатні надсилати дані сенсорів і отримувати команди. Технологія хмарних обчислень забезпечує величезну обчислювальну потужність і ємність для зберігання в Інтернеті, щоб подолати обмежені ресурсів IoT. Проводиться багато досліджень щодо віртуалізації пристроїв IoT у хмарному середовищі для полегшення віддаленого доступу та контролю. У майбутньому доступ до пристроїв IoT буде здійснюватися через відповідні віртуальні об’єкти. Подібно до мережі фізичних пристроїв, у кіберсвіті має існувати мережа віртуальних об’єктів.

У магістрській роботі пердставлено концепцію побудови динамічної віртуальної мережі в хмарному середовищі серед підключених пристроїв IoT. Ключова ідея полягає в тому, щоб забезпечити механізм побудови віртуальної мережі між підключеними пристроями IoT з різних доменів через відповідні віртуальні об’єкти в хмарному середовищі. Це сприятиме спільному використанню ресурсів і швидкому розвитку різноманітних додатків на рівні віртуалізації шляхом встановлення динамічного наскрізного з’єднання між пристроями IoT. У роботі пропонується детальний дизайн запропонованої системи для побудови віртуальної мережі IoT. Також реалізовано протоколи прикладних рівнів у OMNET++ для моделювання мережі віртуальних об’єктів, щоб провести аналіз продуктивності запропонованої віртуалізації мережі IoT.

Мета роботи – аналіз ефективності застосування хмарної мережевої віртуалізації для динамічного підключення пристроїв в мережі IoT.

Для досягнення поставленої мети в роботі виконано наступні завдання:

1. Проведення аналіз існуючих рішень по віртуалізації IoT
2. Дослідження можливостей віртуалізації мереж IoT
3. Проведення моделювання мереж віртуальних об'єктів для оцінки ефективності запропоновоного рішення віртуалізації.

Об'єкт дослідження – процес підключення пристроїв в мережі IoT

Предмет дослідження – хмарна мережева віртуалізація для підключення пристроїв в мережі IoT

Методи дослідження. У процесі виконання поставлених завдань використовувалися методи теорії інформації, математичні методи системного аналізу, теорії ймовірності і математичної статистики.

Наукова новизна одержаних результатів. У магістрській роботі представляємо детальний дизайн системи для побудови віртуальної мережі IoT.

Практична значущість одержаих результатів. Отримані результати можуть бути використані при побудові ефективних мереж IoT з елементами віртуалізації.

Апробація: Пешков І.О. “Віртуалізація для систем інтерету речей за допомогою хмарних технологій: рішення, підходи та проблеми”, Телекомунікаційні та інформаційні технології.

1 АНАЛІЗ РОЗГЛЯДАЄМОЇ ПРОБЛЕМИ

1.1 Постановка задачі

Інтернет речей (IoT) призначений для підключення комунікаційного пристрою з усім, що ми хочемо контролювати за допомогою Інтернету. Ядро мереж IoT складається з двох типів пристроїв: (a) датчики, які збирають контекстні дані; та (b) приводи, які отримують команди для керування середовищем. IoT в основному зосереджено на підключенні речей (об’єктів повсякденного життя з приєднаними датчиками або приводами) до Інтернету, щоб користувачі могли дистанційно контролювати та контролювати певну діяльність або пристрій. Системи IoT мають величезні можливості та застосування [1]. Нещодавно багато гігантських виробничих і розробних організацій інвестували в цю технологію, щоб реалізувати її потенціал. Багато проектів ініційовано для розробки різноманітних додатків IoT для багатьох реальних проблем і проведення досліджень у різних напрямках для вирішення різних аспектів систем на основі IoT разом із пов’язаними проблемами безпеки, автентифікації та авторизації, а також ідентифікації скомпрометованих вузлів.

Для полегшення доступу та контролю пристроїв IoT у кіберсвіті введено концепцію віртуалізації, яка забезпечує гнучкий інтерфейс для маніпулювання та взаємодії з фізичними пристроями IoT [2].

Пристрої IoT представлені віртуальними об’єктами в кіберсвіті, які забезпечують гнучкий зв’язок між датчиками та виконавчими механізмами. Окрім гнучкості, віртуалізація також сприяє безпроблемному дублюванню та спільному використанню ресурсів між різними доменами. Багато існуючих систем надають послуги віртуалізації для фізичних пристроїв шляхом створення свого віртуального об'єкта (ВО). Пізніше ВО об'єднуються для створення сервісів, які можна використовувати для створення різних додатків. Віртуальний світ також відомий як кіберсвіт, а вся система тоді називається кіберфізичною системою (CPS), призначеною для спостереження за фізичним світом і контролю за допомогою віртуального світу в середовищах IoT.

Зазвичай зв’язок між пристроями підтримується в мережі за допомогою таблиць маршрутизації. Таблиці маршрутизації забезпечують ефективний і швидкий спосіб пересилання пакетів. Вносити зміни на вимогу в таблиці маршрутизації непросто, і тому концепція програмно визначеної мережі (SDN) введена для розділення мережевого керування та планів даних. Однак SDN вимагає від мережевих пристроїв виконання певних складних операцій і протоколів, які не підтримуються обмеженими пристроями IoT. Наприклад, протокол OpenFlow зазвичай використовується контролерами SDN для оновлення записів потоку в мережевих пристроях для керування трафіком на вимогу. У літературі можна знайти кілька досліджень щодо розробки програмованої інфраструктури SDN для забезпечення екосистеми IoT. Однак пристрої IoT мають обмежені ресурси з точки зору пам’яті та обчислень, і надання підтримки для протоколів, подібних до OpenFlow, призведе до зниження продуктивності.

Технології хмарних обчислень спрямовані на забезпечення стабільних і спільних обчислювальних ресурсів і ресурсів зберігання для різноманітних користувачів і програм. З появою цієї технології кінцеві користувачі можуть мати повну комп’ютерну систему в Інтернеті відповідно до своїх бажаних специфікацій і бюджетних обмежень у формі віртуальної машини. У багатьох прикладних сценаріях екземпляри віртуальних машин, запущені на хмарній платформі, повинні спілкуватися одна з одною, і незабаром віртуальні мережі між віртуальними машинами будуть створені за допомогою віртуальних комутаторів і віртуальних маршрутизаторів. Після створення віртуальних мереж у кіберсвіті функції інших важливих мережевих пристроїв, таких як пристрої виявлення вторгнень, балансувальники навантаження, брандмауери тощо, також були віртуалізовані у формі віртуалізації мережевих функцій (NFV). Структура NFV вимагає трьох компонентів: (a) програмна реалізація мережевої функції; (b) інфраструктура для розгортання функції віртуальної мережі; та (c) структуру управління для функції віртуальної мережі.

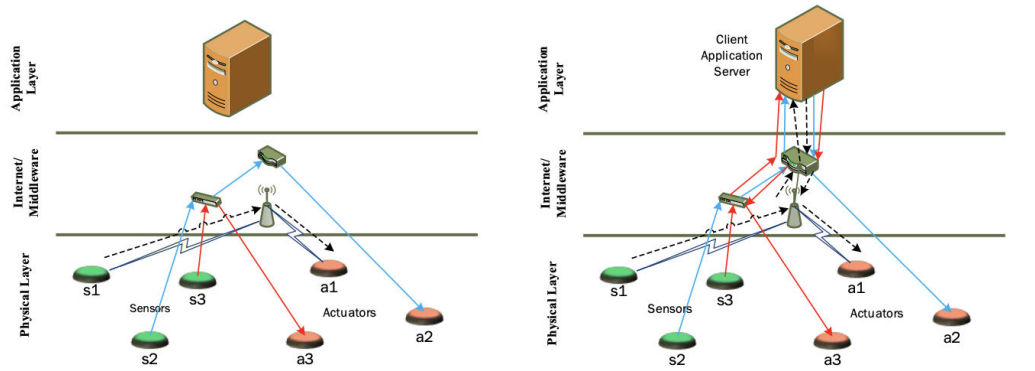
Звичайний життєвий цикл розробки продукту вимагає ретельного проектування, тестування та стандартизації перед випуском, що було дуже дорогим і трудомістким. NFV привніс революцію в розробку продукту, оскільки

його можна швидко розгорнути, протестувати в реальному середовищі та легко оновити, якщо це необхідно.

Раніше більшість досліджень були зосереджені на віртуалізації датчиків у хмарному середовищі. Усі операції зі збору та обробки даних на підключених віртуальних об’єктах оброблялися логікою додатків, і не було такої потреби у створенні віртуальної мережі. Однак із появою IoT з’єднано два типи пристроїв, тобто датчики та виконавчі механізми. Основна відмінність між додатками на основі чисто сенсорної мережі та додатками на основі IoT полягає в тому, що додатки на основі датчиків зазвичай використовуються лише для моніторингу та аналізу, тоді як додатки на основі IoT використовуються для автоматизації процесів. У додатках на основі Інтернету речей після ретельного аналізу даних датчиків за допомогою простих схем на основі правил або розширених алгоритмів машинного навчання команди надсилаються на відповідні виконавчі пристрої для виконання бажаних операцій. Іншими словами, існує певний зв’язок або мережа між сенсорними та виконавчими пристроями. Традиційно конфігурацію для створення цієї мережі виконували або на фізичних пристроях IoT, або на прикладному рівні, як показано на рисунку 1a,b відповідно.

Дуже складно змінити параметри мережі в пристроях IoT, і нам потрібно отримати фізичний або віддалений доступ до кожного пристрою, а потім внести потрібні зміни. З ростом кількості підключених пристроїв IoT стало неможливо вносити зміни в окремі пристрої IoT. Налаштування мережі на прикладному рівні вимагає повного контролю доступу до відповідних пристроїв IoT, що зазвичай можливо, якщо пристрої та програма належать одному власнику. Однак цей підхід не є сприятливим для спільного використання базової фізичної мережевої інфраструктури IoT. Після віртуалізації пристроїв IoT у хмарному середовищі ми можемо легко оновлювати та тиражувати віртуальні об’єкти, щоб безперешкодно спільно використовувати один пристрій IoT між багатьма залежними програмами. Оскільки віртуальні об’єкти знаходяться в хмарній інфраструктурі, віртуальні IoT мережі можуть бути встановлені серед пов’язаних віртуальних об’єктів у хмарному середовищі. У цій роботі представлено концепцію створення віртуалізованої мережі IoT у хмарному середовищі серед підключених пристроїв

IoT. Останні розробки в різних мережевих і хмарних інфраструктурних технологіях дозволили нам вивчити концепцію віртуалізації мережі IoT. На рисунку 2 показано три найбільш релевантні сфери, які призвели до розвитку віртуалізованих мереж IoT.



(a) IoT devices with preconfigured connectivity. **(b)** IoT devices connectivity established through application server.

Рисунок 1.1 – Традиційний підхід до створення мереж IoT

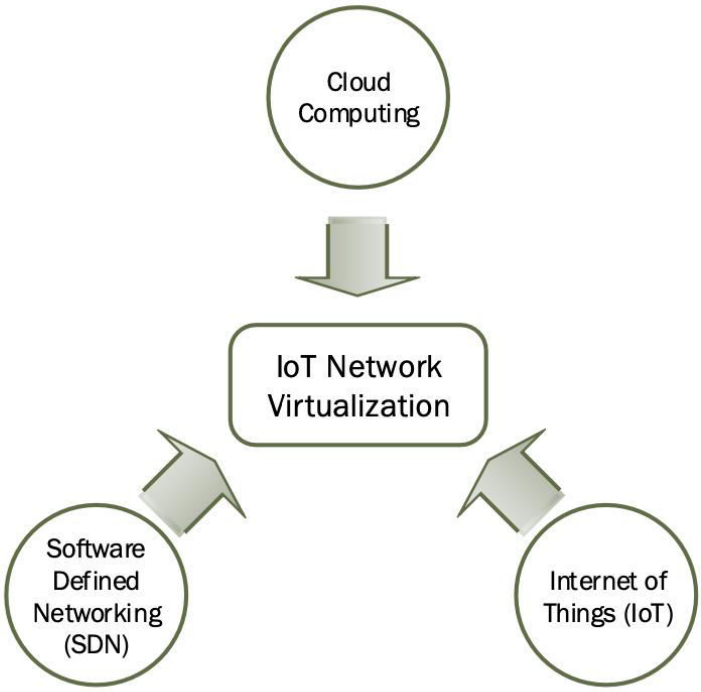


Рисунок 1.2 – Три найбільш релевантні сфери, які призвели до розвитку віртуалізованих мереж IoT

1.2 Аналіз наукових робіт

З моменту створення Інтернет речей привернув багато уваги дослідників, і його вважають однією з революційних технологій майбутнього, які можуть змінити кожен аспект людського життя. Різноманітні додатки на основі IoT розроблені для створення розумного середовища в різних секторах послуг розумних міст, наприклад, розумні будинки, розумне здоров'я, розумна освіта, безпека, розваги та промисловість, і т. д. У минулому передбачалося, що WSN функціонуватиме всередині певного домену, і дослідники були проти використання стека протоколів (через пристрої з обмеженими ресурсами), і глобальний унікальний ідентифікатор для кожного пристрою не був необхідним. Удосконалення інформаційно-комунікаційних технологій (ІКТ) і мікроелектроніки дозволило мільярдам пристроїв Інтернету речей підключитися до Інтернету за допомогою глобального унікального ідентифікатора. Пізніше цю концепцію віртуальної сенсорної мережі (VSN) використовували для позначення підмножини сенсорних вузлів, призначених для конкретного завдання. VSN забезпечує логічне підключення між вибраними вузлами датчиків, хоча ці вибрані вузли можуть не бути фізично підключеними, як показано на рисунку 3. У цьому прикладі сценарію існує дві віртуальні мережі датчиків (a) для моніторингу температури (b) для моніторингу середовища проживання. Сенсорні вузли в обох мережах забезпечують логічний зв'язок між сенсорними вузлами кожної мережі.

Рисунок 1.3 – Приклад сценарію двох сенсорних мереж для моніторингу температури та середовища існування

Для використання потенціалу та потужності існуючих фізичних мереж застосовуються різні підходи. Серед початкових зусиль у цьому напрямку можна виділити концепцію накладної сенсорної мережі (OSN). Накладена мережа встановлюється поверх іншої фізичної мережі, яка забезпечує інфраструктуру для зв'язку. Зазвичай під час накладання мереж базова мережа базується на зовсім інших технологіях, ніж та, яка накладається поверх неї. Пакети з однієї мережі інкапсулюються в пакети допоміжних мереж, щоб увімкнути послуги, які не надаються існуючою мережею. Часто використовувані розподілені системи, такі як клієнт-серверні програми, однорангові мережі та хмарні обчислення, є хорошими прикладами накладених мереж, оскільки окремі вузли в цих типах мереж працюють у всесвітньому Інтернеті. Сам Інтернет є накладеною мережею, спочатку побудованою поверх існуючих телефонних ліній. Накладені мережі також обрамляються поверх бездротових сенсорних мереж, у яких віртуальна топологія створюється поверх фізичної топології в WSN. Такі накладні мережі утворюються підмножиною вузлів у WSN, і ці вузли не обов'язково повинні знаходитися поруч один з одним або мати канал зв'язку з одним стрибком. Припускається, що між вузлами накладеної мережі існує віртуальний зв'язок, який може відповідати повному шляху, що охоплює кілька вузлів базової фізичної мережі. Для бездоганної інтеграції різноманітних мереж реалізація для формування накладної мережі виконується на прикладному рівні за допомогою інкапсуляції пакетів. Однак для деяких накладених мереж може знадобитися невелика модифікація базових рівнів стеку протоколів.

Ще одну архітектуру доступу для надання QoS у бездротових сенсорних мережах шляхом побудови накладеної мережі поверх WLAN 802.11. Накладена мережа діє як контрольна площина для забезпечення зв'язку між активними потоками з максимальним використанням пропускної здатності.

Пристрої IoT — це невеликі пристрої з живленням від батареї з обмеженими можливостями зберігання, обчислення та зв'язку. Щоб подолати проблеми обмеження ресурсів у сенсорних вузлах і пристроях IoT, було розроблено різні

типи мереж. Для вирішення цієї проблеми зазвичай використовується хмарна архітектура. Мадрія та ін. представили концепцію сенсорної хмари для з'єднання географічно диверсифікованих мереж [2]. Вони використовували віртуальні датчики для вирішення проблем обмежених ресурсів фізичних датчиків. Тестовий стенд був розроблений у сенсорній хмарі для запуску різних додатків з доступом у великого кола користувачів. Запропоновано хмарну архітектуру «Cloud4sens» для забезпечення віддаленого інтерфейсу для моніторингу та керування датчиками. «The Things Network» надає набір інструментів, які більше зосереджені на швидкій розробці додатків IoT з мінімальними/нульовими зусиллями кодування через операції «drag and drop»[3]. Ефективний розподіл ресурсів у середовищі віртуальної сенсорної мережі є складною проблемою, і в літературі пропонуються різні рішення для вирішення цієї проблеми.

Концепція віртуалізації в WSN не нова Бхаттачарія та ін. представили ідею об'єднання сенсорних мереж для віртуалізації та спільного використання сенсорних ресурсів між доменами [4]. Кілька досліджень представляють прототип реалізації віртуалізації датчиків, наприклад, модель віртуалізації датчиків у мережах сенсорів тіла. Вони також реалізували запропоновану модель за допомогою архітектури «SPINE2», яка є фреймворком з відкритим вихідним кодом для підтримки доменно-спеціальної обробки сигналів і роботи зондування бездротових сенсорних мереж. Левіс розробив крихітну віртуальну машину «Mate» для сенсорних мереж. «Mate» надає інтерфейси для кодування складних програм у дуже малому розмірі (менше 100 байт), що дозволяє уникнути проблем із передачею та зберіганням у сенсорних мережах. Леонтіадіс розробив платформу SenShare для відокремлення інфраструктури сенсорних мереж від додатків, що працюють на ній. Таким чином, кілька додатків можуть спільно використовувати одну базову інфраструктуру для виконання різних операцій. Крім того, віртуалізація в сенсорних мережах може допомогти у розділенні проблем у звичайних WSN. Раніше мережами WSN керувала та володіла одна організація, і та сама організація відповідала за спільне використання своєї мережі для надання послуг різним клієнтським програмам. У майбутньому можна створити логічну мережу серед серверів віртуалізації для різних доменів для

підтримки спільного використання ресурсів для надання скоординованих послуг клієнтським програмам для майбутнього розумного світу, як показано на рисунку 4. Для кожного домену ми можемо мати сервер віртуалізації, який містить інформацію про пов’язані пристрої IoT у формі віртуальних датчиків або віртуальних об’єктів. Ці сервери віртуалізації можуть бути підключені для спільного використання інформації про ресурси між доменами.

Більшість існуючих досліджень зосереджено на віртуалізації пристроїв IoT у хмарному середовищі, щоб їх можна було легко тиражувати та спільно використовувати між різними програмами. Однак технології хмарних обчислень забезпечують централізований надійний доступ до великого обсягу даних разом із величезною обчислювальною потужністю, але вони не підходять для чутливих до затримок програм через значну затримку під час надсилання даних на централізований хмарний сервер для обробки. Щоб вирішити цю проблему, можна використовувати технології периферійних обчислень, де обчислення наближаються до джерела даних, щоб уникнути непотрібних затримок зв’язку.

Рішення віртуалізації на основі SDN для пристроїв IoT зосереджені на дистанційному керуванні пристроями IoT для внесення необхідних змін у конфігурацію, але цей підхід потребує ресурсів обчислення та зберігання, що може призвести до зниження продуктивності. Хмарна віртуалізація сервісів забезпечує централізоване спільне сховище віртуальних об’єктів для зареєстрованих пристроїв IoT, щоб їх можна було легко надавати спільного доступу та тиражувати. Однак усі операції зі збору та обробки даних на підключених віртуальних об’єктах оброблялися логікою додатків, і не було як такої потреби у створенні віртуальної мережі. З появою IoT з’єднано два типи пристроїв, тобто датчики та виконавчі механізми. У додатках на основі IoT певним чином існує зв’язок або мережа між сенсорними та керуючими пристроями, які можуть бути реалізовані у формі створення віртуального IoT у хмарному середовищі серед пов’язаних віртуальних пристроїв IoT.

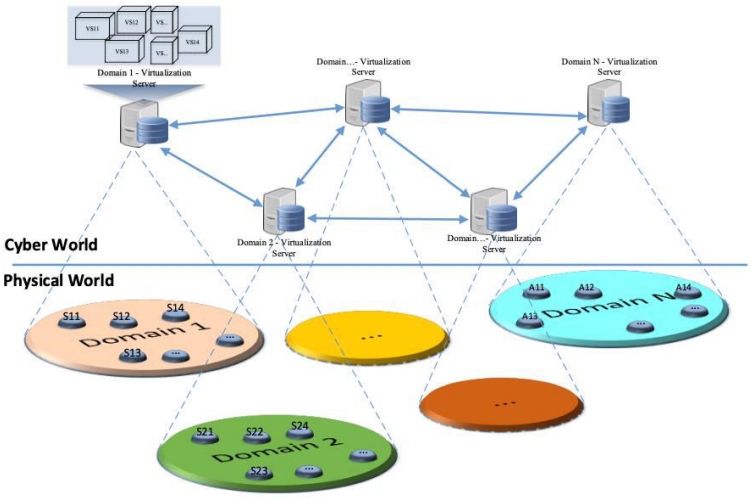


Рисунок 1.4 – Мережа віртуалізації серед для різних доменів для підтримки спільного використання ресурсів.

2 АНАЛІЗ МЕТОДІВ ВІРТУАЛІЗАЦІЇ МЕРЕЖ IoT

Бездротові сенсорні мережі (WSN) є ключовою парадигмою нових мереж IoT і використовуються в широкому діапазоні областей застосування. Вони розгортаються для підтримки певних додатків. Для багатьох комерційних систем IoT підключення до хмарної або туманної платформи можуть забезпечити вільний доступ та дає змогу підключити різноманітні додатки. Віртуалізація — це технологія, яка потенційно може забезпечити такий обмін.

2.1 Визначення та поняття

Контекст Хмари сягає початку 1960-х років, коли Джон Маккарті уявив, що обчислювальні ресурси будуть надаватися громаді як допоміжний сервіс. У 1990-х роках слово «хмара» використовувалося для визначення кількох понять, наприклад, зображення великих мереж банкоматів [5].

Крім того, генеральний директор Google Ерік Шмідт у 2006 році застосував цей термін, щоб описати бізнес-парадигму надання комп'ютерних послуг через Інтернет. Відтоді термінологія хмарних обчислень застосовувалася в основному як бізнес-вираз для визначення у багатьох різноманітних контекстах [6]. Однак багато експертів з бізнес-маркетингу та наукових досліджень прагнуть пояснити визначення хмарних обчислень і які особливі характеристики вони надають.

Хмара — це паралельна та розподілена обчислювальна система, що складається з набору взаємопов'язаних та віртуалізованих комп'ютерів, які динамічно надаються та представлені як один або декілька уніфікованих обчислювальних ресурсів на основі рівня обслуговування.

Крім того, Вакуеро[7] зазначив, що «хмари — це великий пул віртуалізованих ресурсів, які легко використовувати та доступні (таких як апаратне забезпечення, платформи розробки та/або служби). Ці ресурси можна динамічно реконфігурувати для адаптації до змінного навантаження (масштабу), забезпечуючи також оптимальне використання ресурсів. Цей пул ресурсів

зазвичай використовується за моделлю оплати за використання, у якій гарантії надаються Постачальником інфраструктури за допомогою затверджених угод про рівень обслуговування».

NIST заявив, що хмарні обчислення мають п’ять основних характеристик («Самообслуговування на вимогу», широкий доступ до мережі, об’єднання ресурсів, швидка еластичність, вимірюване обслуговування»), три моделі обслуговування («Програмне забезпечення як послуга» (SaaS).), «Платформа як послуга» (PaaS), «Інфраструктура як послуга» (IaaS)) і чотири моделі розгортання («Приватна хмара, хмара спільноти, публічна хмара, гібридна хмара»).

Загалом, Сатомойер (2009) зазначив, що «хмара частіше використовується для позначення IT-ресурсів, які розгортаються в інфраструктурі як послуги».

Стандарт IEEE 1934 визначив туманні обчислення як «горизонтальну архітектуру системного рівня, яка розподіляє ресурси та послуги обчислень, зберігання, керування та мереж у будь-якій точці континууму «хмара-речі». Він підтримує галузеві вертикалі та домени додатків, дозволяє розповсюджувати служби та додатки ближче до джерел, що створюють дані, і поширюється від речей за межі мережі через хмару та через кілька рівнів протоколу».

Головним чином, туманні обчислення мають кілька характеристик, таких як:

• Низька затримка: вузли «туману» розташовані поблизу пристроїв IoT, що допомагає зменшити затримку обробки запиту.

• Широкий географічний розподіл: «туманний шар» включає набір гетерогенних вузлів туману, розподілених у великих географічних зонах, щоб задовольнити різноманітні вимоги додатків і послуг, які потребують розпорошеного розгортання.

• Неоднорідність: «туманний шар» включає різні типи крайових пристроїв, які мають різні ресурси.

• Підтримка мобільності: необхідна для багатьох застосувань туману гарантувати прямий зв’язок із мобільними пристроями за допомогою стандартних протоколів.

• Взаємодія в реальному часі: туманні обчислення підтримують взаємодію в режимі реального часу між пристроями «fog», що необхідно для послуг чутливих до часу.

• Масштабованість: туманні обчислення пропонують розподілену обробку та ресурси, необхідні для обробки величезних кількість даних, створених пристроями IoT.

2.2 Віртуалізація систем IoT

Революція Інтернету речей глибоко вплинула на мережеву інфраструктуру. IoT трансформує взаємодію між кібернетичним і фізичним простором з величезним впливом на повсякденне життя. Ефективність цих технологій потребуватиме нових методологічних та інженерних підходів через вражаючий масштаб проблеми та нові складні вимоги щодо продуктивності, безпеки та надійності. Віртуалізація мережі одночасно йде своїм шляхом, наприклад, віртуалізація мережевих функцій і програмно-визначені мережі. Очікується, що програмно-визначені мережі в поєднанні з мережевими функціями віртуалізації та хмарними обчисленнями стануть критично важливою технологією, яка радикально революціонізуватиме спосіб, у який мережеві оператори створюватимуть свою інфраструктуру.

Віртуалізація відноситься до процесу абстрагування фізичних ресурсів і приховує складність конструкції основного апаратного забезпечення та демонструє важливі деталі, які потрібні розробникам для розгортання їхньої програми. Віртуалізація роз’єднує тісний зв’язок, який зв’язує мережі з їхніми програмами під час розгортання. Таким чином, техніка віртуалізації забезпечує механізм ізоляції між фізичними ресурсами та додатками, що працюють на вершині, отже, фізичні ресурси не знають, яку програму обслуговуватимуть, і програма працює незалежно від платформи, яка залежить від фізичної інфраструктури.

Морабіто дослідив доцільність запуску віртуалізованих екземплярів на широкому діапазоні малопотужних вузлів. Аналіз показав, що використання

технологій віртуалізації контейнерів на одноплатних комп'ютерах (SBC) дає майже незначний ефект щодо продуктивності порівняно з нативними виконаннями. Крім того, враховуючи компроміс між продуктивністю та енергоспоживанням за великого набору робочих навантажень, вони емпірично підтвердили енергоефективність SBC. Загалом аналіз продуктивності мережі показав, що 64-розрядні пристрої центрального процесора (ЦП) не створюють жодних відчутних витрат порівняно з 32-розрядними аналогами. Це дослідження має деякі переваги, такі як покращення продуктивності та ефективності, гнучкості, високої потужності та низького енергоспоживання.

Кархула та Валта зосередилися на проблемах підключення до Інтернету речей, розробляючи та оцінюючи шлюз Інтернету речей, який використовує методи віртуалізації функціональності. Функціональними можливостями шлюзу керують як контейнерами, які можна надавати, оновлювати, запускати та швидко видаляти. Крім того, шлюз підтримує найсучасніші протоколи IoT і забезпечує підключення до Інтернету для пристроїв, які за замовчуванням не підтримують стек Інтернет-протоколів. Архітектура шлюзу підтримує граничні обчислення для зменшення затримки та навантаження на мережу. Крім того, запропонована реалізація забезпечує відповідний спосіб керування функціональністю шлюзу IoT.

Алам, Чоудрі та Нол представили фреймворк віртуалізації IoT для підтримки відповідних датчиків, пропонуючи рішення з базовою хмарою IoT. Фреймворк використовує дані датчика, щоб відобразити функціональні особливості пов'язаних об'єктів хмари IoT. Вони застосували покращену політики доступу, щоб підтвердити, що лише юридичні особи можуть отримати доступ до послуг IoT, що призводить до підвищення загальної безпеки інфраструктури. Крім того, модель сервіс-орієнтованої архітектури, керованої подіями (e-SOA), допомагає структурі керувати процесом моніторингу, виявляючи різноманітні підключені об'єкти та реагуючи на них.

Запропонований фреймворк демонструє, як послуги на основі датчиків можуть бути включені в сценарії зв'язку з обмеженими ресурсами. Вони запропонували принципи проектування та продемонстрували розширення застосування IoT зі здатністю міркувати, використовуючи тематичне

дослідження «Green School MotorCycle» (GSMC). Результати продемонстрували, що поєднання e-SOA, семантичних веб-технологій і віртуалізації охоплює підхід до вирішення проблем безпеки, підключення та моніторингу IoT. Незважаючи на великі переваги запропонованого фреймворку, такі як динамічність та сумісність, з точки зору складності, легкості та гнучкості, аналіз продуктивності фреймворку в реальному часі не проводився.

Також Келайдоніс представив фреймворк для когнітивного управління об'єктами в IoT. Фреймворк має три рівні функціональності, і кожен рівень включає когнітивні сутності, які забезпечують засоби для самостійного керування розумними об'єктами. Фреймворк дозволяє абстрагувати неоднорідність, яка є результатом величезної кількості різноманітних об'єктів, одночасно покращуючи надійність і цілісність системи. Також було введено поняття віртуальних об'єктів (ВО) з метою зв'язку віртуального світу з фізичним. Вони також представили відповідний прототип, який був прийнятий для доказу запропонованої структури.

Для досягнення принципів IoT WSN має віртуальну роль, щоб забезпечити платформу для надання доступу до ресурсів. Таким чином, Лукас Мартінес, і Ернандес Діас представили метод віртуальної ілюстрації джерел подій у WSN. У запропонованому методі була обрана концепція оркестровки, оскільки вона не потребує додаткових змін у раніше розгорнутих простих датчиках. Було запропоновано базовий набір правил для створення сервісів і нових додатків. Також було запропоновано додаткове правило для налаштування служби для генерації події, коли швидкість зміни вимірювань перевищує певний поріг. Мета статті полягає в тому, щоб продемонструвати, що навіть пристрої з обмеженнями можуть включати деякі базові додатки, але розширення цих правил, що дозволяє формувати більш складні додатки за допомогою будь-яких джерел подій. Незважаючи на те, що модель була протестована в реальній реалізації, це дослідження потребує додаткового тестування, щоб розширити та покращити її можливості.

Дар, Тахеркорді та Еліассен досліджували сервіси IoT на рівні хмари. Вони запропонували структуру, яка базується на загальній моделі, що може включати різні шаблони резервування та використовувати метод віртуалізації для

організації цих шаблонів під час виконання у формі віртуальних служб. Віртуальний сервіс у структурі також підсумовує логіку обробки даних датчика та функціональність виклику служб IoT. У цьому підході відповідність функціональних потреб очікуваному рівню надійності зменшує витрати на обслуговування сервісів і запобігає потенційним хвильовим ефектам від розриву сервісів пізніше в реальному середовищі. Щоб перевірити можливість використання хмарних послуг IoT, вони реалізували мережу на основі «Contiki» реальних вузлів «Tmote Sky» у хмарній платформі «SicsthSense». На основі цієї реалізації він зауважив, що запропонована структура має незначні накладні витрати на обробку порівняно із загальною затримкою обробки та зв'язку з використанням протоколу обмеженого застосування «CoAP» для доступу до даних датчиків. Крім того, фреймворк дуже легко налаштовується з точки зору введення більшої кількості моделей.

З іншого боку, Мелоні, Пегораро, Атзорі, Беніньї та Суліс обговорювали архітектуру IoT для ефективного моніторингу електромережі та запропонували практичну реалізацію точної системи. Функція віртуалізації IoT, продуктивність і гнучкість хмари були використані для отримання гнучкої системи моніторингу, побудованої на системі глобальних вимірювань на основі блоку вимірювання вектора (PMU). Аналіз продуктивності архітектури було виконано на активній мережі, що є результатом фактичної системи в Арізоні, змодельованої як тестовий фідер IEEE 34, за допомогою симулятора «Opal-RT». Результати показали, що вимоги до якості обслуговування (QoS) щодо затримки та точності процесу прогнозування можуть бути гарантовані значним зменшенням необхідної пропускної здатності. Крім того, результати показали, що запропонована архітектура може виконувати поставлені операційні завдання, задовольняючи при цьому необхідний рівень якості як з точки зору зв'язку, так і з точки зору мережі з важливим ступенем гнучкості та адаптивності. Це дослідження, незважаючи на переваги масштабованості та гнучкості, зосередилося лише на одній сфері віртуалізації.

У польовій мережі (FAN), де пов'язано багато пристроїв IoT з різними протоколами та форматами даних, можуть виникати збої через невизначеність

FAN. Нова платформа для керування складним FAN, до якої підключено багато пристроїв із різноманітними комунікаційними технологіями та протоколами, була запропонована в Нішигучі, Яно, Охтані, Мацукара та Какута. Платформа полегшує важливі функції для стабільної роботи FAN: збір інформації про стан пристрою та мережі, формування топології та усунення проблем шляхом віртуалізації пристроїв у FAN. Незважаючи на переваги гнучкості та зниження частоти помилок, це дослідження має деякі недоліки. Наприклад, запити від платформи до пристроїв можуть бути невдалими, оскільки комунікаційні середовища FAN є незахищеними.

Крім того, Батала представив кількісне порівняння двох провідних гіпервізорів з відкритим кодом, XEN (гіпервізори з відкритим кодом) і віртуальної машини на основі ядра (KVM), зосередившись на пропускній здатності та затримці, які є ключовими факторами промислового Інтернету речей. У статті розглядається методологія обчислення пропускної здатності та затримки на обчислювальному пристрої, який містить кілька підсистем IoT. Найважливішим результатом стало те, що гіпервізор KVM забезпечує високу продуктивність. Значення пропускної здатності в KVM було приблизно в п'ять разів вище, ніж у XEN. KVM з паравіртуалізацією може стати в нагоді в більш складних випадках. У випадку затримки мережі два гіпервізори схожі. Однак, на відміну від XEN, гіпервізор KVM має сильну залежність між затримкою та прийнятим навантаженням. Загальний висновок полягає в тому, що гіпервізор KVM є кращим вибором для віртуалізації IoT у більшості випадків. Хоча це дослідження не реалізовано в реальному середовищі, воно покращило продуктивність, затримку та пропускну здатність.

Маттос, Веллосо та Дуарте запропонували метод інтеграції складних мережевих послуг із пристроїв Інтернету речей через гнучку та ефективну інфраструктуру віртуалізації мережевих функцій ізольованих доменів Інтернету речей. Таким чином, метод розробляє простий вузол доступу до шлюзу, який віртуалізує домени, до яких підключаються пристрої. У кожному домені використовуються ізольовані та адаптовані мережеві функції, щоб відповідати вимогам безпеки та продуктивності кожного додатоку IoT. Було реалізовано

прототип QoS, і його перевірка показує, що віртуалізація вузла доступу не впливає на продуктивність віртуальної мережі. Результати показали, що віртуалізований вузол шлюзу доступу не має втрат продуктивності. Оскільки метод не застосовує протоколи, які виконують керування потоком, зміна затримки зв'язку між шлюзом та інфраструктурою віртуалізації мережевої функції (NFVI) не залежить від швидкості отримання пакетів. Іншим важливим моментом є те, що метод базується на консолідованих і широко доступних інструментах. Незважаючи на те, що дослідження не приділяє достатньо уваги витратам, воно містить багато переваг, включаючи високий рівень безпеки для пристроїв IoT, розпізнавання зловмисного трафіку з високою точністю, уникнення відмови в основних послугах і гарантування якості обслуговування.

Вердоу, Волферт, Бюленс і Райленд дослідили ідею віртуальних ланцюгів постачання їжі з точки зору IoT і запропонували нову хмарну архітектуру інформаційних систем. Крім того, архітектура призначена для реалізації інформаційних систем. Вони стверджували, що віртуалізація може відігравати важливу роль у вирішенні конкретних проблем ланцюгів постачання харчових продуктів, харчових продуктів, що швидко псуються, непередбачуваних коливань поставок і суворих потреб у безпеці харчових продуктів. Додатки, засновані на цих віртуалізаціях, також дозволяють інвесторам діяти безпосередньо у разі відхилень. Крім того, запропонована архітектура інформаційних систем показала, як цей новий підхід до віртуалізації може бути реалізований за допомогою загальних технологічних механізмів, що містять IoT і хмару. Незважаючи на те, що ця структура має багато переваг, зокрема спрощення управління ланцюгом постачання та дистанційну оптимізацію бізнес-процесів, необхідні подальші дослідження для систематичної кількісної оцінки впливу віртуалізації на ефективність ланцюга постачання.

Крім того, Сера, Санабріо-Русо, Пубіль і Верікоукіс представили спільну архітектуру периферійних хмарних обчислень, щоб відповідати складним вимогам аналітики даних IoT. Таким чином, можна задовольнити потреби як у додатках IoT із низькою затримкою, так і в додатках, стійких до затримок. Крім того, платформа зіткнулася зі складними неоднорідними характеристиками даних

IoT, тобто їх високою розмірністю або георозподіленою природою. З одного боку, нещодавні досягнення в методах машинного навчання (ML) використовуються для визначення того, як можна виконувати аналіз даних IoT на поточній платформі. З іншого боку, віртуалізація, централізоване управління, глобальний огляд і програмованість обчислювальних і мережевих ресурсів вважаються такими, що відповідають вимогам підходів ML. Було визначено три різноманітні сценарії для спільної крайової хмарної платформи. Результати сценарію показали, що розподілені методи ML вимагають низької затримки та надійного зв'язку. Крім того, отримано глобальне уявлення про комунікаційні та обчислювальні ресурси для зовнішніх додатків через північні інтерфейси (NBI). Незважаючи на такі переваги цього дослідження, як зменшення затримки, надійний зв'язок і підвищення продуктивності, воно зосереджувалося лише на обмеженому діапазоні проблем.

Саманєго, Еспана та Детерс представили архітектуру, яка поєднує розширені обчислення з доставкою віртуальних ресурсів на периферійному рівні. Крім того, протокол блокчейну на основі дозволів поєднанню для досягнення безпеки та надійності в периферійних мережах. Архітектура пропонує інтелектуальні віртуальні ресурси, розміщені на периферійних пристроях, щоб надавати перегляд даних і послуги Інтернету речей клієнтам. Архітектура, представлена в цій роботі, розроблена на основі особливостей кожної рекомендованої технології та результатів попередніх оцінок цих технологій. Цей підхід виходить за рамки віртуальної віртуалізації Інтернету речей і формує розподілені віртуальні системи, які містять переваги хмарних і туманних обчислень, щоб надавати послуги безпосередньо на периферійному рівні. Однак архітектура не реалізована в реальному середовищі.

2.3 Рівні віртуалізації

Віртуалізація є новою областю досліджень, і в цьому контексті було запропоновано багато підходів і рішень. Однак ці рішення реалізують методи віртуалізації на різних рівнях абстракції (можливо, на апаратному рівні, рівні

мережі або рівні даних). Тому вкрай необхідно класифікувати ці рівні, щоб усунути неоднозначність у тому, як можна реалізувати методи віртуалізації.

Переважно техніка віртуалізації має три рівні.

1. Віртуалізація на апаратному рівні

2. Віртуалізація на рівні мережі

Віртуалізація на апаратному рівні – це абстракція обчислювальних ресурсів від програмного забезпечення, що працює на вершині [8]. Це означає, що віртуальну машину можна створити як нову версію реального комп'ютера з такою операційною системою, як Windows VMware Workstation і Oracle VM VirtualBox. Мета полягає в тому, щоб відокремити програми від базових ресурсів і забезпечити спільне використання ресурсів між різними програмами. Віртуалізація на рівні мережі може бути досягнута шляхом створення віртуальних вузлів через підключення віртуальних каналів через фізичну мережу [9], таку як віртуальна приватна мережа (VPN). Мета полягає в тому, щоб дозволити багатьом віртуальним мережам співжити в одній фізичній мережі. По суті, віртуальна мережа також відома як підмножина фізичної мережі, де кожна підмножина обслуговує різні програми. Віртуалізація на рівні даних стосується процесу керування, абстрагування та інкапсуляції даних таким чином, щоб користувачі додатків надавали доступ до збережених даних для отримання, запиту та маніпулювання даними, не знаючи технічних деталей даних. джерело, фізичне розташування або спосіб його форматування. Віртуалізація даних покращує гнучкість управління, масштабованість, ефективне використання ресурсів та енергоефективність.

Проте один очевидний спільний зв'язок між цими рівнями полягає в тому, що техніка віртуалізації копіює оригінальні ресурси (апаратне забезпечення, мережі або дані) і надає користувачам віртуальний перегляд. Таким чином, реалізація методів віртуалізації на якому рівні в основному залежить від вимог і специфікацій програми.

Однак характеристики рівнів віртуалізації підсумовані в таблиці 2.1.

Таблиця 2.1

Характеристики рівнів віртуалізації

Virtualization level	Virtualization method	Virtualization tools
Hardware level	Isolates running applications from the underline hardware using encapsulation mechanism	Operating system and Virtual machine
Network level	Can be achieved by forming of virtual networks above the physical networks using virtual nodes and virtual links	Cluster-based formation and Overlay-based formation
Data level	Grants applications an access to data in such away that they can retrieve and manipulate data without any changes to the original resource	Docker container and Database partitioning

Техніка віртуалізації є багатообіцяючим рішенням для усунення недоліку спільного використання ресурсів, коли вона дозволяє більшій кількості програм співіснувати та виконувати свої завдання одночасно в одній мережі. Важливість віртуалізації WSN виникла в основному через те, що деякі сенсорні вузли залишалися бездіяльними та не виконували жодних завдань. Віртуалізація є важливою технікою, яка ефективно використовує неутратні ресурси сенсорних мереж. Техніка віртуалізації вважається ключем до Інтернету майбутнього [9], тому вкрай важливо впровадити цю техніку в сенсорних мережах, щоб задовольнити зростаючі вимоги технології IoT.

Проте віртуалізація в сенсорних мережах досліджувалась у сфері академічних досліджень. Наприклад, Мерентітіс [10] розглядав віртуалізацію як ключовий елемент обміну інформацією в парадигмі IoT. А ту саму інфраструктуру WSN можна віртуалізувати для підтримки кількох додатків одночасно в хмарі та надання таких послуг, як «мережа як послуга» (NaaS) різним користувачам.

Цао [11] стверджував, що платформи IoT можуть використовуватися кількома додатками в розумних містах, а техніку віртуалізації WSN у розумному

місті для ефективного використання розгорнутої інфраструктури. Він важав віртуалізацію в WSN як значну інновацію для створення великомасштабної сенсорної інфраструктури, яка використовується для ефективного використання ресурсів та запропонував архітектуру для віртуалізації WSN на основі кількох рівнів: фізичного рівня, віртуального рівня та шару накладання, щоб дозволити розгорнутій мережі спільно використовувати різні програми.

2.4 Хмарні обчислення

Нещодавній прогрес у доступності та масштабованості IT та Інтернету зробив обчислювальні ресурси ефективнішими та доступними повсюдно для громадськості у формі послуг на вимогу . Така технологічна орієнтація призвела до появи сучасної концепції, відомої як «Cloud Computing». За допомогою цієї моделі обслуговування сервери, сховища, мережі тощо надаються як комунальні послуги (надають обчислювальні ресурси як послугу, а не як продукт) різним користувачам через Інтернет за принципом «Pay-as-you-go».

Окрім цих ресурсів, хмара надає різноманітні типи програмних послуг, як-от API (інтерфейси прикладних програм) та інструменти розробки, щоб підтримувати користувачів у розробці інноваційних програм у режимі реального часу швидким та ефективним способом. Крім того, хмарні обчислення забезпечать автентифікацію користувачів і цілісність даних у спільному середовищі.

Основною метою надання таких API є сприяння розгортанню програм на хмарній платформі. Однак хмарні обчислення були розроблені на основі різних технологій, таких як «паралельні обчислення», «розподілені обчислення», «грід-обчислення» та еволюції техніки віртуалізації.

Хмарні та туманні системи IoT

IoT — це технологічна революція, яка показує майбутнє підключення та доступності. Поява як хмарних обчислень, так і IoT змінює спосіб розгляду інформаційних і комунікаційних систем. IoT стимулює еволюцію хмарних технологій до розподілу ресурсів між багатохмарності та включення

різноманітних різнорідних пристроїв. Хмарні обчислення та Інтернет речей сприяли новому режиму логістичного обслуговування, тобто режиму хмарної логістики. Спеціальні програми IoT можуть вимагати розгортання шлюзів на межі мережі, щоб увімкнути її взаємодію з фізичними датчиками, попередню обробку даних із цих датчиків і синхронізацію їх із хмарою. У цьому відношенні віртуалізація може підвищити безпеку хмарних обчислень.

Хмарне рішення віртуалізації

Спочатку бажано висвітлити базову концепцію пов’язаних технологій і представити зв’язок між ними, а також те, як запропоноване рішення з’явилося на світлі.

По-перше, хмарні обчислення виграють від методів віртуалізації, віртуалізуючи апаратне забезпечення, дозволяючи спільне використання доступу та ефективне використання ресурсів. Тому хмарні обчислення максимізують свої прибутки та обслуговуватимуть велику кількість різноманітних програм. Впровадження методів віртуалізації в хмарних обчисленнях з’явиться в новому контексті під назвою хмарне середовище віртуалізації, яке еквівалентно зростаючим вимогам програм IoT.

По-друге, системи зондування IoT використовуватимуть чудові можливості, які надає хмарне обчислення, і виграш, досягнутий від методів віртуалізації, щоб підвищити продуктивність і широке поширення в різних географічних областях. У таблиці 1 описуються переваги та недоліки пов’язаних технологій, які відзначили Рашид і Чатурведі [12]. Крім того, інтеграція сенсорних мереж із хмарним середовищем віртуалізації призведе до нової повсюдної обчислювальної послуги, відомої як парадигма сенсорної хмарної віртуалізації. Однак налаштоване рішення зображено на рис. 1. Де ця конфігурація ґрунтується на висновках попереднього провадження. Загалом WSN складається з чотирьох основних частин: сенсорні вузли, зона моніторингу, базова станція та програми користувача [13]. Датчики будуть збирати інформацію з моніторингу місцевості

та застосовувати початкову попередню обробку та агрегацію, а потім передавати дані на базову станцію, де можна застосувати методи обробки та аналізу.

Віртуалізація в реальному часі широко визнана як один із ключових факторів, що сприяють розвитку туманному та хмарному IoT. Нарешті, Ру і Штайнер досліджували віртуалізацію в реальному часі та вбудовану віртуалізацію. У них є вимоги, яким повинен відповідати будь-який сенсор, що кваліфікується як детерміноване рішення віртуалізації для IoT. Вони охарактеризували існуючу роботу в галузі віртуалізації в реальному часі, щоб проілюструвати компроміс між гнучкістю та детермінованим виконанням. Крім того, вони вказали на відсутність сенсорів, які відповідають усім вимогам детермінованої віртуалізації. Розглянуті сенсори показують або високі максимальні затримки, або високу середню затримку системи. Їх дослідження, незважаючи на слабкість реалізації в реальному середовищі, зменшило затримку системи та підвищило гнучкість.

Як ми бачили в цій частині, основні фактори, які були проаналізовані в обговорюваних документах, включають гнучкість, оптимізацію, затримку, безпеку, продуктивність і надійність, на затримку приділяють найбільшу увагу дослідники. Таким чином, віртуалізація може покращити безпеку та затримку, а також може підвищити загальну продуктивність IoT на основі Cloud і Fog.

Інтеграція IoT-WSN із хмарною віртуалізацією

З технічної точки зору сенсорна мережа розглядається як ключовий елемент для еволюції парадигми IoT, де вона забезпечує взаємодію з фізичним світом і витягує дані зондування.

Незважаючи на переваги, які система зондування надає різноманітним популярним програмам (наприклад, розумне місто, розумний дім, система охорони здоров’я, моніторинг навколишнього середовища, виявлення об’єктів, відстеження цілей, військові програми тощо), величезна кількість даних, зібраних сенсором вузли, вимагають належного зберігання, динамічних обчислювальних ресурсів, ефективних методів обробки та аналізу, і їх потрібно спільно використовувати та обслуговувати для програм, які потребують інформації. Крім того, ці величезні дані не використовуються ефективно через дефіцит досвіду,

грошей і часу, які передбачають, щоб дані були ідеально оброблені та збережені для майбутнього використання.

У цьому контексті залучення технології з високою продуктивністю та потужними обчислювальними ресурсами є важливим. Хмарна інфраструктура віртуалізації є багатообіцяючим рішенням для підтримки обмежених мереж і систем розповсюдження. Таке перенаправлення даних, отриманих із сенсорних мереж, у достатні ресурси для обробки, спільного використання та зберігання дозволить використати можливості, які надає це ефективне рішення.

Однак процес інтеграції WSN з хмарою передбачив появу парадигми Sensor-Cloud, яка забезпечує гнучке об’єднання WSN, динамічне управління ресурсами та економічно ефективне надання послуг.

Сенсорно-хмарний підхід

Сучасне вдосконалення технології IoT призвело до з’єднання мільйонів маленьких пристроїв і фізичних об’єктів по всьому світу, щоб вони могли автоматично спілкуватися один з одним без допомоги людини. Сучасні інфраструктури, такі як розумні будинки, розумні міста, розумна мережа, розумний транспорт, розумні водопровідні системи тощо, дозволяють інтелектуально та плавно взаємодіяти, контролюючи нашу повсякденну діяльність і полегшуючи життя [14]. Це відіграє важливу роль у різних аспектах технічного життя та дає змогу використовувати багато додатків, таких як виявлення пожежі, безпека будівель, контроль інвентарю, нагляд за кордоном, відстеження пацієнтів, моніторинг руху тощо. Повсюдний характер сенсорних пристроїв, легше розгортання та гнучкий зв’язок підвищують їхню популярність та здатність інтегруватися з іншими новими технологіями [15].

Однак головним завданням сенсорних мереж є моніторинг фізичних явищ у нашому середовищі (таких як температура, тиск, тепло, світло тощо) і передача відповідної інформації системам моніторингу, які потім формують інформацію. база даних для надійного управління даними та прийняття правильних рішень [14]. У традиційних WSN мережеві моделі мають деякі обмеження, які обмежують їх використання в сучасних інноваційних програмах. Кожен WSN призначається одній програмі, у якій користувач/власник є єдиним

відповідальним за видалення мережі, розподіл ресурсів, програмування та обслуговування.

Сьогодні, зі зміною поколінь у WSN на основі IoT, підключені машини та датчики можуть приймати автономні рішення. Використовуючи інтелектуальні датчики та штучний інтелект, машина може інформувати іншу машину, коли відбувається певна подія, тому відповідні дії можуть бути прийняті незалежно, а потім власник може бути повідомлений. Відношення між правом власності та використанням зібраних даних нещодавно виникло в таких кіберфізичних системах, де ці дані збираються з різнорідних датчиків, розташованих у різних місцях, консолідуються, обробляються та аналізуються для подальшого використання.

«Sensor-Cloud» — це нова парадигма WSN, яка вирішує цю проблему та відокремлює власників фізичних датчиків від користувачів мережі. Багато сенсорних мереж, які можуть належати різним організаціям і розгорнуті в різних географічних областях, об’єднані за допомогою хмари, що дозволяє їм взаємодіяти одна з одною одночасно для кількох програм.

Це призвело до появи так званого SenaS (Sensing as a Service) і відкрило нові можливості для бізнесу з даними. Фізичні обмежені ресурси ефективно віртуалізуються в хмарі, а потім можуть бути запропоновані як послуга багатьом клієнтам через Інтернет у різних географічних регіонах відповідно до вимог їхніх програм.

Він також дозволяє інтегрувати численні мережі датчиків від різних постачальників послуг і передавати зібрані дані в хмарну інфраструктуру [16]. «Sensor-Cloud» віртуалізує фізичні датчики в хмарних ресурсах і створює шаблон емуляції, відомий як віртуальні датчики, які користувачі можуть автоматично надавати та відключати відповідно до вимог додатків.

Однак сучасні технології дали нові визначення для «Sensor-Cloud». Sensor-Cloud — це інфраструктура, яка дозволяє справді всеосяжне обчислення, використовуючи датчики як інтерфейс між фізичним і кіберсвітом, кластери даних і обчислення як кібермагістраль та Інтернет як середовище зв'язку.

Ідея віртуалізації мережі IoT запозичена з концепції NFV в контексті SDN і хмарних обчислень. NFV дозволяє швидко розвивати функціональні можливості реальних пристроїв у вигляді програмного пакету та зручно розгортати його в хмарному середовищі. Крім того, за допомогою NFV ці віртуальні пристрої можна легко оновлювати, покращувати, тиражувати та спільно використовувати.

IoT дозволяє підключати невеликі сенсорні та комунікаційні пристрої до Інтернету та приєднувати їх до об’єктів повсякденного життя для віддаленого моніторингу та контролю. Очікується, що найближчим часом до Інтернету будуть підключені мільярди пристроїв IoT, які генеруватимуть величезну кількість даних. Багато досліджень пропонують і віддають перевагу підключенню пристроїв IoT до хмарного середовища для зручного зберігання та обробки зібраних даних за допомогою розширеної аналітики даних і схем обробки великих даних, які підтримуються хмарною платформою. Провідні постачальники хмарних послуг, такі як Google, Amazon, Microsoft тощо, також надають підтримку та API для підключення пристроїв IoT безпосередньо до їхніх платформ, щоб їхні дані могли легко передаватися та оброблятися відповідними клієнтськими програмами. Подібно до віртуальних машин і віртуалізованих мережевих функцій, ми можемо мати віртуальне представлення пристроїв IoT у хмарному середовищі, яке зазвичай називають віртуальними об’єктами. Основна відмінність між NFV і віртуальними об’єктами полягає в тому, що NFV не мають фізичного пристрою, який підтримує їх, тоді як віртуальні об’єкти підтримуються реальними пристроями IoT. Після підключення пристроїв IoT і створення відповідних віртуальних об’єктів у хмарному середовищі можна створити віртуальну мережу IoT серед пов’язаних віртуальних об’єктів.

На рисунку 2.1 показано концептуальний вигляд формування мережі віртуального IoT серед віртуальних пристроїв IoT (віртуальних об’єктів). На рисунку 2.1 також представлено короткий підсумок еволюції типів мереж. Мережа типу 1 відповідає звичайним фізичним мережам, де інформація про маршрутизацію зберігається на проміжних маршрутизаторах. Для будь-яких змін у налаштуваннях мережі до кожного маршрутизатора потрібно отримати індивідуальний доступ. Коли розмір мережі зростає, індивідуальне

обслуговування мережевих пристроїв стає виснажливою роботою. Щоб подолати ці проблеми, були запроваджені мережі типу 2, тобто SDN. У SDN плани даних і керування розділені. Адміністратор мережі може використовувати централізовані контролери для налаштування та обслуговування мережі. Контролер взаємодіє з окремими мережевими пристроями, щоб відобразити необхідні зміни в їхніх внутрішніх таблицях маршрутизації. Мережеві пристрої лише пересилають дані відповідно до правил, визначених контролером у їхніх таблицях маршрутизації. Концепцію віртуалізованих мереж IoT можна розглядати як мережі типу 3, де мережа серед підключених пристроїв IoT підтримується в хмарному середовищі через відповідні віртуальні об’єкти.

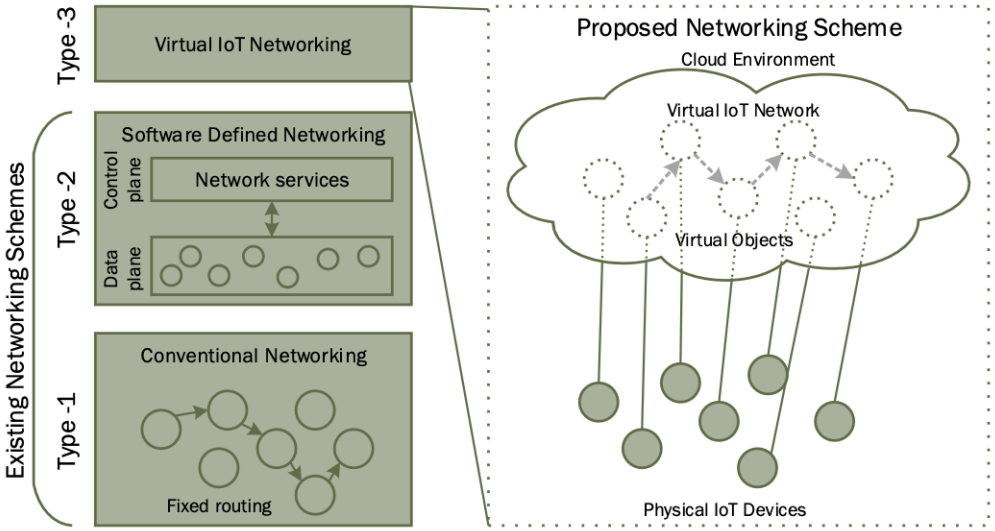


Рисунок 2.1 – Концептуальний вид віртуалізованої мережі IoT

На рисунку 2.2 показано більш детальне уявлення про створення віртуалізованої мережі IoT для різноманітних програм у хмарному середовищі. У цій запропонованій архітектурі клієнтські програми мають спілкуватися лише з централізованим контролером, щоб визначити необхідну конфігурацію мережі. Як і контролер у мережах SDN, модуль контролера є централізованим і відповідає за формування та обслуговування мережі. На рисунку 6 показано налаштування трьох віртуалізованих мереж IoT за запитом для різних клієнтських програм.

Запропонована архітектура підтримує спільний доступ до тих самих віртуальних об’єктів у різних мережах.

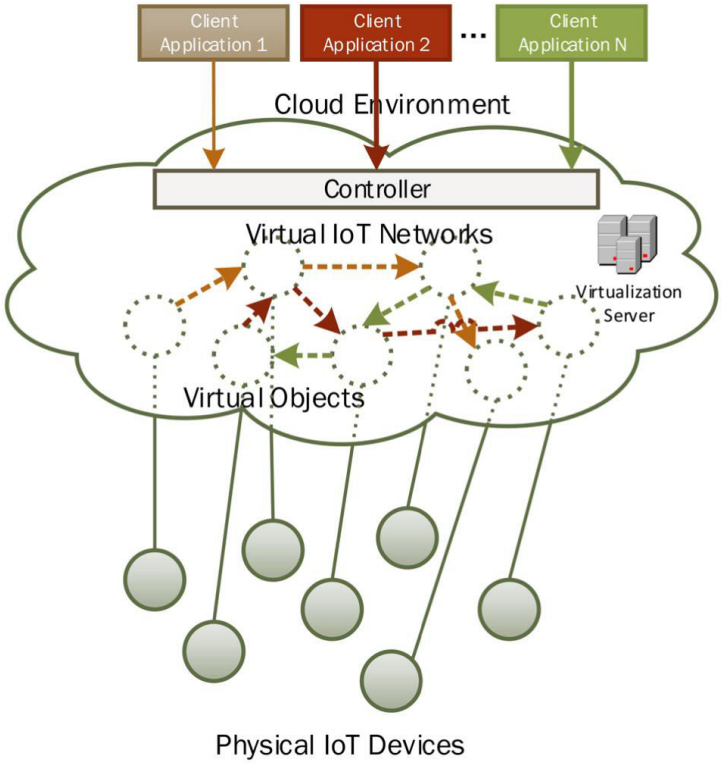


Рисунок 2.2 – Створення віртуалізованої мережі IoT для різноманітних програм у хмарному середовищі

Багаторівнева архітектура запропонованої системи для віртуалізації мережі IoT наведена на рисунку 2.3. Вона має три рівні: (a) Фізичний рівень складається з фактичних пристроїв IoT, які можуть бути сенсорними та керуючими пристроями, підключеними до системи. Пристрої IoT можуть бути розподілені між різними доменами з однаковими цілями та завданнями; (b) рівень віртуалізації діє як проміжне програмне забезпечення між клієнтськими програмами та фізичними пристроями IoT. Віртуальні об’єкти створюються на цьому рівні для кожного під’єднаного пристрою IoT, і ці віртуальні об’єкти використовуються різними клієнтськими програмами та спільно використовуються ними. Віртуалізована мережа IoT встановлюється серед пов’язаних пристроїв IoT відповідно до вимог клієнтської програми та бажаних налаштувань. Віртуальні об’єкти надають інтерфейс фактичному пристрою IoT для використання його послуг у різних

програмах. (c) Різні додатки розміщені на прикладному рівні для споживання послуг базового рівня віртуалізації. Програми з будь-якого домену можуть мати доступ до спільних ресурсів через рівень віртуалізації. На рисунку 2.3 також показано формування віртуальної мережі IoT на рівні віртуалізації для двох різних клієнтів на рівнях додатків. Додатки виражають необхідну кількість пристроїв IoT із бажаними параметрами підключення через спеціальний інтерфейс, а логіка відображається у формі віртуальної мережі серед пов'язаних віртуальних об'єктів.

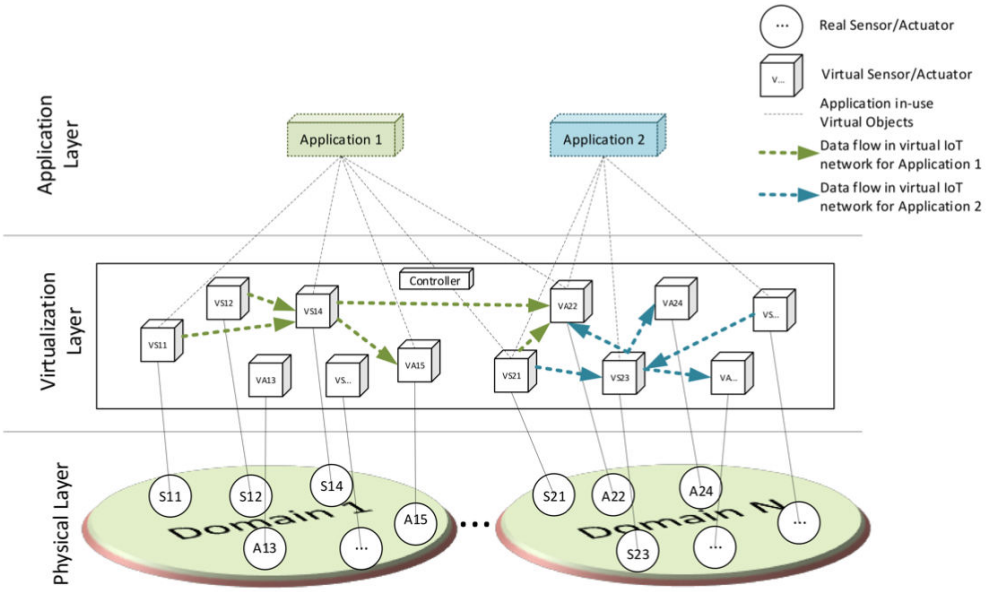


Рисунок 2.3 – Архітектура віртуалізації мережі IoT для двох різних клієнтів.

Основні компоненти запропонованої системи для віртуалізації мережі IoT та її функціонування показано на рисунку 2.4 за допомогою діаграми послідовності. Процес починається з реєстрації пристрою IoT, коли кожен пристрій IoT публікує інформацію свого профілю на попередньо налаштований сервер віртуалізації, надсилаючи запит на реєстрацію. Типовий профіль віртуального об'єкта для IP-камери містить інформацію, як показано у форматі XML на рисунку 2.5. Після необхідної перевірки інформації про профіль пристрою створюється віртуальний об'єкт для відповідного пристрою IoT, і на відповідний пристрій надсилається повідомлення з підтвердженням. Віртуальний об'єкт використовується сервером віртуалізації для подальшого зв'язку з відповідним пристроєм IoT. Після цього

користувач ініціює запит через клієнтську програму на потрібний тип і кількість пристроїв IoT. Контролер обробляє запит і отримує профілі віртуальних об'єктів відповідних пристроїв IoT відповідно до визначених користувачем критеріїв. Отримавши списки віртуальних об'єктів у клієнтській програмі, користувач вказує та налаштовує потрібні параметри. Після цього бажані користувачем налаштування розгортаються через клієнтську програму, а запит надсилається до контролера. Контролер відповідає за встановлення бажаних параметрів мережі шляхом маніпулювання віртуальними об'єктами на сервері віртуалізації. Динамічні зв'язки створюються між пов'язаними віртуальними об'єктами відповідно до бажаних користувачем налаштувань шляхом оновлення списку зіставлення. Після цього на відповідні датчики IoT надсилається команда активації для ініціювання передачі даних. Дані датчиків отримують відповідні віртуальні об'єкти, а потім команда активації пересилається до наступного підключеного віртуального об'єкта після вибору зіставленого пристрою зі списку зіставлення. Виконавчий пристрій виконав потрібну операцію, і повідомлення про підтвердження надсилається клієнтській програмі. Процес триває до закінчення часу активації пристрою. Інформація про підключення для певної віртуалізованої мережі IoT видаляється зі списку зіставлення, коли час її активації закінчився.

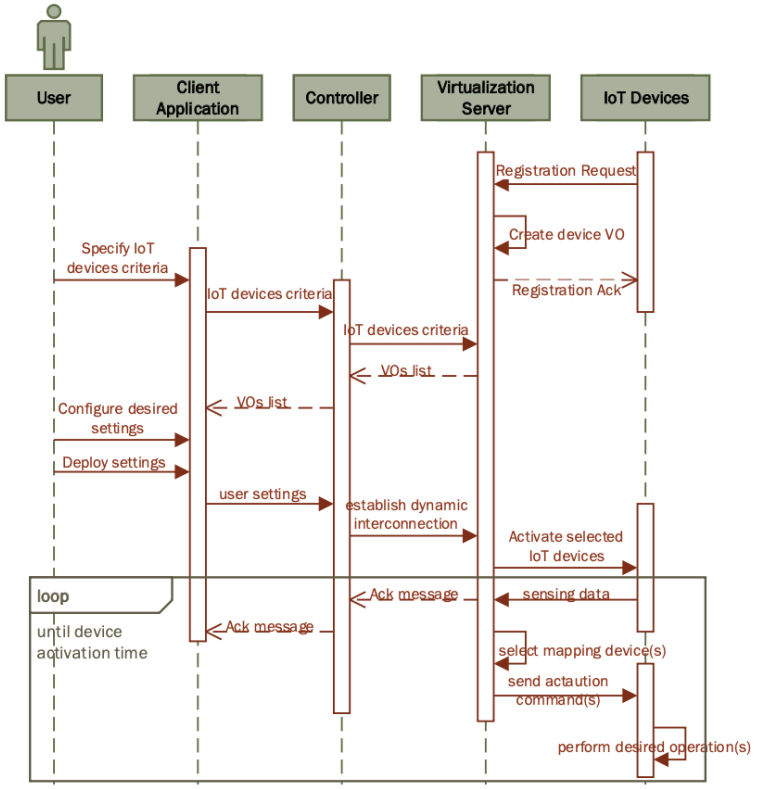


Рисунок 2.4 – Основні компоненти системи для віртуалізації мережі IoT та її функціонування

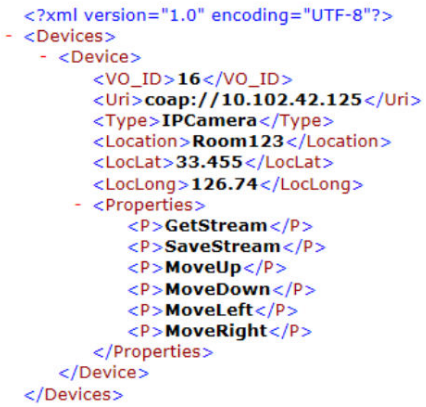


Рисунок 2.5 – Типовий профіль віртуального об’єкта для IP-камери

Головна таблиця відображення та таблиці потоків

У запропонованому налаштуванні сервер віртуалізації зберігає віртуальні об’єкти всіх підключених пристроїв IoT. Контролер можна розмістити на одному сервері або за потреби розгорнути на окремому сервері. Сервер віртуалізації забезпечує віртуальне представлення для зареєстрованих пристроїв IoT у формі віртуальних об’єктів, тоді як модуль контролера взаємодіє з клієнтами, щоб маніпулювати підключенням між віртуальними об’єктами за бажанням клієнтів. Таблиця 2.2 — це таблиця, модуля контролера та містить всю інформацію про віртуальні мережі IoT, створені в хмарі для типового сценарію, наведеного на рисунку 2.3. Дві програми використовують віртуальні об’єкти на рівні віртуалізації та потік зв’язку для кожної віртуальної мережі. Мережа IoT позначається іншим кольором. Як показано на рисунку 2.3, деякі віртуальні об’єкти спільно використовуються двома програмами, наприклад, VS21 і VA22 використовуються обома програмами. Для VS21 у таблиці 2.2 зроблено два записи (тобто ідентифікатори запису 1010 і 1011) для обробки та генерації пакетів для обох програм.

Таблиця 2.2

Entry ID	Network ID	VO ID	Source ID	Target ID	Received From	Set Source ID	Set Target ID	Send To	Entry Type
1001	vIoTNet-1	VS11	-	-	S11	S11	A22	VS14	Event
1002	vIoTNet-1	VS14	S12	A15	VS12	-	-	VA15	Event
…	…	…	…	…	…	…	…	…	…
1010	vIoTNet-1	VS21	-	-	S21	S21	A22	VA22	Event
1011	vIoTNet-2	VS21	-	-	S21	S21	A24	VS23	Periodic
1020	vIoTNet-2	VS23	S21	A24	VS21	-	-	VA24	Periodic
1021	vIoTNet-2	VA24	S21	A24	VS23	-	-	A24	Periodic
…	…	…	…	…	…	…	…	…	…

Кожен запис унікально ідентифікується ідентифікатором, тобто ідентифікатором запису. Кожен запис у головній таблиці представляє зв'язок між двома реальними/віртуальними об'єктами та належатиме до певної віртуальної мережі IoT з унікальним ідентифікатором мережі. Ця таблиця містить основну інформацію про з'єднання між віртуальними об'єктами, а інформація про з'єднання окремого віртуального об'єкта ідентифікується полем «VO ID». Кожен пакет, згенерований у цій віртуальній мережевій архітектурі IoT, може бути призначений для одного чи кількох цільових вузлів. Ідентифікатор джерела та цілі вказує на джерело та призначення для певного потоку зв'язку. Коли віртуальний об'єкт отримує пакет від реального пристрою, він встановлює вихідний і цільовий ідентифікатори відповідно до правил, визначених головною таблицею відображення. Поле «отримано від» використовується для ідентифікації попереднього переходу пакета, щоб він міг переслати пакет до наступного вузла, визначеного полем «надіслано». Поле типу запису вказує на тип потоку даних, наприклад, на основі подій, періодичний тощо. Термін дії вказує на термін дії віртуальної мережі IoT, і всі записи буде видалено, коли закінчиться термін дії.

Приклади записів у таблиці потоків для VS21, показаній на рисунку 2.3, наведені на рисунку 2.6. Цей віртуальний об'єкт спільно використовується двома програмами. Для кожної програми робиться окремий запис, щоб указати правила потоку для пересилання даних на рівні віртуального об'єкта. Припустімо, що програма 1 створює систему на основі подій, і її правило буде запущено, якщо значення даних датчика перевищує певний поріг. Для другого застосування ми припускаємо систему періодичного керування, яка в цьому прикладі надсилає команду приводу через регулярний інтервал 0,1 с. Один пакет, отриманий віртуальним об'єктом, може бути перенаправлений до кількох цільових вузлів за допомогою кількох записів у таблиці потоків. Крім того, якщо запис потоку має

тип події, тоді поле умови перевіряється, якщо воно істинне, а потім виконується відповідна дія. Якщо тип запису потоку є періодичним, то поле інтервалу використовується для встановлення таймера для наступної передачі пакету.

Entry ID	Network ID	Source ID	Target ID	Received From	Entry Type	Condition	Action	Interval	Set Source ID	Set Target ID	Send To
101	vIoTNet-1	-	-	S21	Event	If data > threshold1	Send message (command= Action1)	-	S21	A22	VA22
102	vIoTNet-2	-	-	S21	Periodic	-	Send message (command= Action2)	0.1	S21	A24	VS23

Рисунок 2.6 - Приклади записів у таблиці потоків для VS21

3 ПРАКТИЧНІ ПРИКЛАДИ ВІРТУАЛІЗАЦІЇ МЕРЕЖ

У наступних підрозділах ми представляємо два різні сценарії використання, щоб проілюструвати роботу та корисність запропонованої концепції віртуалізації мережі IoT. У кожному сценарії ми представляємо певний прикладний сценарій, який потребує певного типу підключення між пов'язаними пристроями IoT. За допомогою цих прикладів ми пояснимо, як віртуалізована мережа IoT встановлюється серед відповідних віртуальних об'єктів для досягнення бажаної мети програми.

3.1 Система смарт відкриття дверей

У цьому прикладі ми розглядаємо простий сценарій, коли користувач хоче розробити програму автоматичного відкривання дверей на основі Інтернету речей у невеликій житловій зоні, яка складається з трьох кімнат. Для спрощення ми розглядаємо два типи пристроїв IoT у цій мережі, тобто датчик руху та дверний актуатор, як показано на рисунку 3.1. У нас є датчик руху, розгорнутий біля дверей кожної кімнати, а відкриття та закриття дверей контролюється відповідним приводом . Загалом на рівні віртуалізації необхідно створити шість віртуальних об'єктів. Для цього сценарію потрібне пряме з'єднання «один-до-одного» між пристроями IoT через віртуальні об'єкти на основі простих правил із використанням умови/дії. Користувач може отримати віртуальний список через інтерфейс клієнтської програми та вказати бажане підключення між відповідним датчиком і приводом. Користувачі також можуть вказати прості правила, що вказують рівень чутливості датчика руху до активації приводу відкриття дверей. Після розгортання контролер оновить налаштування кожного віртуального об'єкта, щоб встановити бажані параметри з'єднання між пов'язаними

віртуальними об’єктами, а команди активації будуть надіслані на датчики руху. Датчик руху почне надсилати дані датчика руху через визначений інтервал до відповідного віртуального об’єкта, де будуть застосовані прості правила, щоб визначити, надсилати чи ні команди активації приводам відкривання дверей.

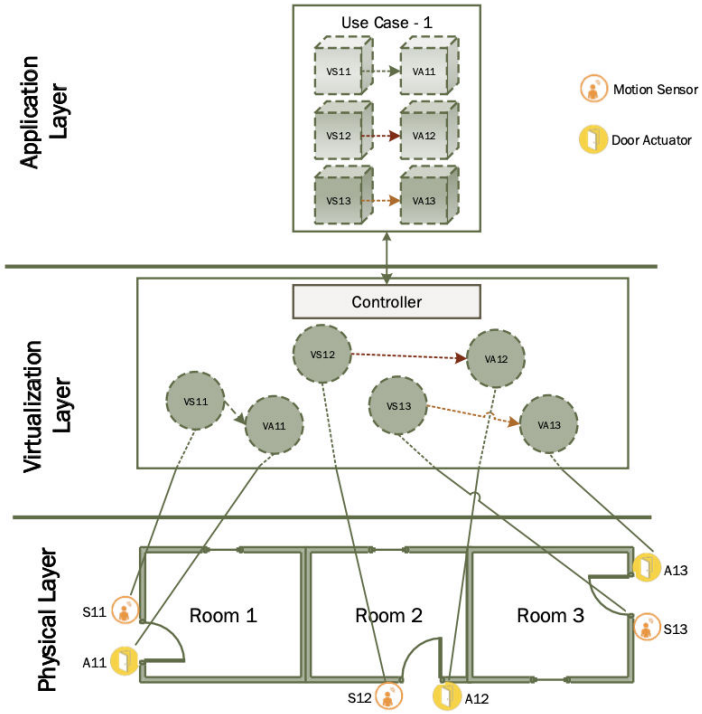


Рисунок 3.1 – Приклад застосування віртуалізованої мережі IoT з типом підключення один-до-одного.

3.2 Системи IoT з віртуалізацією для великого житлового приміщення

Цей сценарій вимагає складного взаємозв’язку між пристроями IoT через віртуальні об’єкти шляхом виконання різноманітних операцій об’єднання, інтеграції та агрегації даних. Цей сценарій відрізняється від двох інших випадків, оскільки тут певні віртуальні об’єкти мають здатність як отримувати, так і надсилати дані. Приклад сценарію застосування наведено на рисунку 3.2, щоб краще пояснити цей сценарій. У цьому випадку клієнт хоче розробити додаток для внутрішнього середовища на основі IoT у невеликій житловій зоні, яка складається з великого холу. Користувач хоче контролювати температуру в приміщенні, освітлення та якість повітря за допомогою цього розумного додатка.

Для спрощення ми розглядаємо шість різних типів пристроїв IoT у цій мережі, тобто три датчики — датчик температури, датчик освітленості та датчики CO2, а також три приводи, тобто кондиціонер/нагрівач, освітлення та два вентилятори. Користувачеві необхідно вказати бажаний діапазон температури, освітленості та якості повітря в приміщенні. Кондиціонер буде працювати, якщо температура в приміщенні буде вище бажаного діапазону, щоб охолодити довкілля в приміщенні, а обігрівач працюватиме, якщо температура в приміщенні буде нижчою від бажаного діапазону, щоб підняти температуру в приміщенні. Подібним чином, якщо рівень освітлення в приміщенні нижчий від бажаного користувачем діапазону, електричні лампочки будуть увімкнені. Так само, якщо якість повітря в приміщенні погіршується (концентрація CO2 перевищує вказаний рівень), вентилятор працюватиме для підтримки якості повітря.

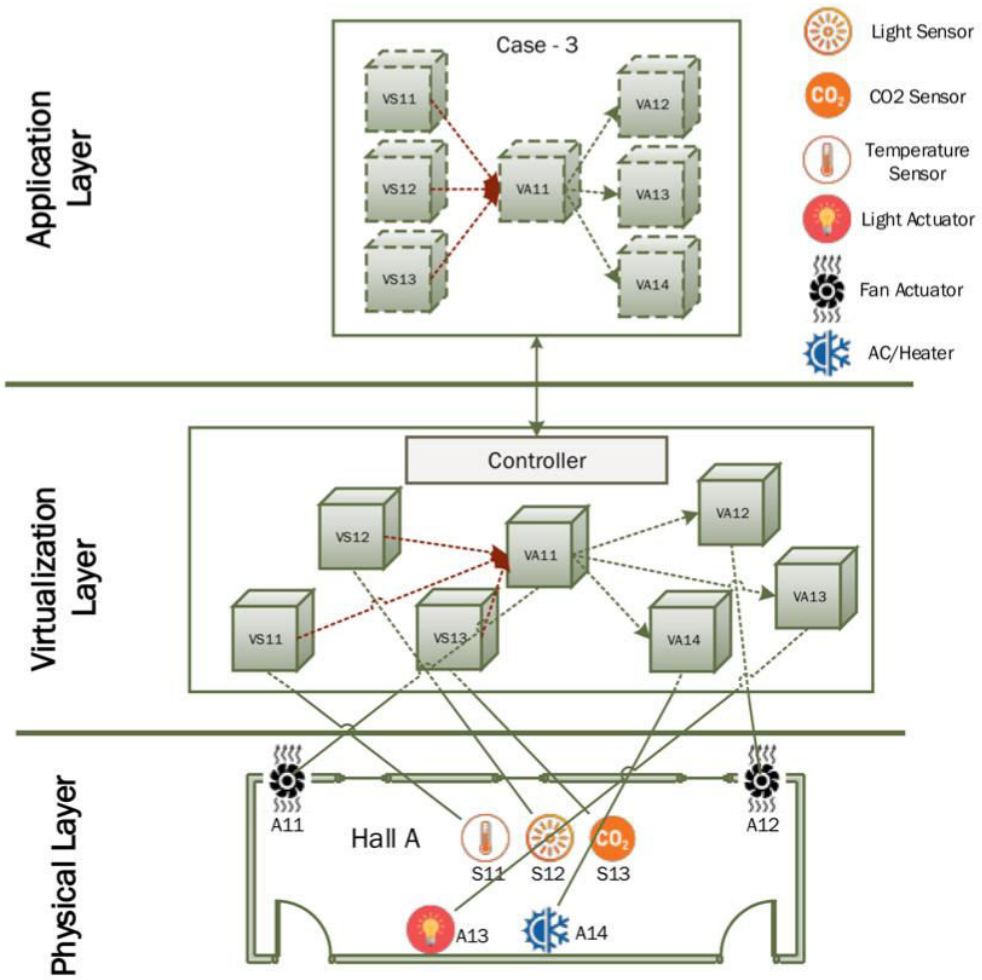


Рисунок 3.2 – Приклад використання віртуалізованої мережі IoT з складним взаємозв’язком між пристроями.

Для цього сценарію необхідно створити загалом сім віртуальних об’єктів на рівні віртуалізації, як показано на рисунку 3.2. Користувач може отримати список віртуальних об’єктів через інтерфейс клієнтської програми та вказати з’єднання між відповідними датчиками та виконавчими механізмами для досягнення бажаної мети. . Користувач також може вказати прості правила відповідно до бажаного діапазону для кожного внутрішнього параметра для активації відповідних виконавчих пристроїв. Після розгортання контролер оновить налаштування кожного віртуального об’єкта, щоб встановити бажані параметри з’єднання між пов’язаними віртуальними об’єктами, як показано на рисунку 3.2, і команди активації будуть надіслані датчикам. Датчики почнуть надсилати дані про оточення в приміщенні через визначений інтервал до відповідного віртуального об’єкта датчика. Відповідно до конфігурації віртуальної мережі, усі віртуальні об’єкти, що сприймають, пересилають дані на призначений віртуальний датчик вентилятора, де застосовуватимуться прості правила, щоб визначити, надсилати чи ні команду активації кожному приводу. Цей сценарій вимагає встановлення з’єднання «багато-до-одного» та «один-до-багатьох» між сенсорами та віртуальними об’єктами приводу. Логіка програми для цього сценарію виражається на прикладному рівні, де її реалізація реалізується на рівні віртуалізації через створення віртуалізованої мережі IoT серед пов’язаних віртуальних об’єктів.

3.3 Моделювання сценаріїв

Ми налаштували гібридну мережу в OMNeT++ для моделювання мережі віртуальних об’єктів, як показано на рисунку 3.3. OMNeT++ означає Objective Modular Network Testbed у C++, і це дуже популярний симулятор для моделювання дискретних подій. Ми також можемо виконувати симуляцію мережі в OMNeT++ за допомогою фреймворку INET (з відкритим кодом). Цей пакет включає різні протоколи для мереж зв'язку (провідних, бездротових і мобільних мереж). У цьому документі ми використовували OMNeT++ 5.2 із фреймворком INET версії 3.6.

Ми розробили три протоколи прикладного рівня в OMNeT++ для клієнтів, серверів і пристроїв Інтернету речей у нашій мережі. Структура каталогів цих протоколів у середовищі OMNeT++ показана на рисунку 14. Серверна система має прикладний протокол віртуалізації VirtualizationServerApp, IoT-пристрій має протокол IoTDeviceApp на прикладному рівні. Після цього ми розробили протокол прикладного рівня для клієнта IoTClientApp для надсилання запиту на сервер віртуальних об'єктів для конфігурації бажаної топології серед зареєстрованих віртуальних об'єктів для виконання потрібної операції.

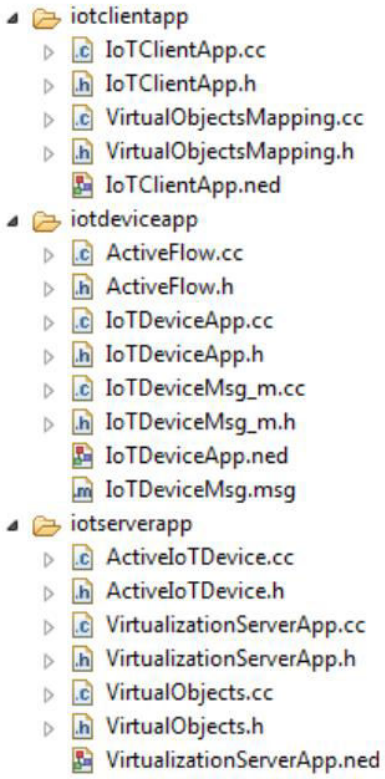


Рисунок 3.3 – Структура трьох протоколів у OMNeT++

Структура повідомлень IoT

Перш ніж представити детальну внутрішню структуру цих протоколів, ми спочатку обговоримо структуру повідомлень, яка використовується в цих експериментах. У таблиці 3.1 наведено короткий опис різних полів повідомлень. Кожне повідомлення має унікальний ідентифікатор для його ідентифікації в середовищі моделювання. У цих експериментах генеруються різні типи повідомлень, наприклад, повідомлення із запитом на реєстрацію, повідомлення з

підтвердженням, командне повідомлення тощо. Тип повідомлення – це ціле число, що вказує на тип повідомлення, а в таблиці 3 наведено короткий опис різних типів повідомлень. використовувалися в цих експериментах. Поле даних у структурі повідомлення є найважливішою частиною структури повідомлення. Залежно від типу повідомлення, пов'язаний вміст повідомлення та інформація кодуються в цьому полі. Синтаксичний аналізатор використовується для отримання інформації з даних на стороні приймача. Розмір поля даних є змінним, оскільки для різних типів повідомлень потрібна різна інформація. Загальний розмір повідомлення в основному залежить від розміру цього поля даних. Далі ми представляємо короткий опис трьох протоколів прикладного рівня, використаних у цьому дослідженні.

Таблиця 3.1

Опис протоколів

Message Field	Description
messageID	Unique ID of message for its identification
type	Indicate the type of message (various message types are given in Table 2)
from	Address of the originator/source of the message
to	Address of the target/destination of the message
sendingTime	The time stamp at which the message was sent/generated from source.
data	Contains the actual content of message in encoded format.
messageLength	The total size (bytes) of message includes header and data contents.

Message Type	Description
1	Registration request message from IoT device
2	Registration acknowledgment message from server
3	Command message received from server
4	Sensing data message from IoT device
5	Command data packet from client to actuator IoT device
10	Data request message from client end
11	Acknowledgement message to data request packet from server to client device

Протокол IoTDeviceApp

IoTDeviceApp — це протокол прикладного рівня, розроблений як для визначення, так і для активації пристроїв IoTIoTDeviceMsg.msg — це файл структури повідомлення, який автоматично компілюється компілятором повідомлень OMNet++. Ця структура повідомлення є спільною для всіх інших протоколів. Цей протокол підтримує необхідну інформацію мовою опису мережі

(NED), як того вимагає симулятор OMNeT++. Вся прикладна логіка цього протоколу реалізована мовою C++. Цей протокол також відстежує активні потоки, зберігаючи інформацію про цільовий пристрій, інтервал надсилання пакетів і час закінчення потоку. Робоча блок-схема цього протоколу наведена на рисунку 3.4.

Під час запуску симуляції вузли IoT виконують необхідні налаштування конфігурації, а потім встановлюють таймер для надсилання запиту на реєстрацію. Коли таймер реєстрації закінчується, OMNeT++ надсилає повідомлення таймера до відповідного модуля як власне повідомлення. Пристрій IoT генерує повідомлення із запитом на реєстрацію (Тип = 1), кодуючи інформацію свого профілю в полі даних, а потім надсилає це повідомлення базовому протоколу транспортного рівня для подальшої передачі на сервер віртуалізації. Таймер знову встановлюється для повторної передачі запиту на реєстрацію, якщо протягом певного часу не буде отримано підтвердження. Отримавши зовнішнє повідомлення, ми спочатку перевіряємо, чи це власне повідомлення (таймери) чи зовнішнє повідомлення, а потім воно відповідно обробляється. Коли отримано повідомлення про підтвердження реєстрації (Тип = 2), пристрій оновлює свої внутрішні налаштування як зареєстровані та скасовує попередньо встановлений таймер, щоб уникнути повторної передачі запиту на реєстрацію.

Після цього пристрій IoT може отримати зовнішнє командне повідомлення (Тип = 3) від сервера віртуалізації, щоб ініціювати передачу даних із заданою швидкістю надсилання. Повідомлення аналізується, генерується активний потік і встановлюється його таймер закінчення (таймер потоку). Потім надсилаються перші пакети даних (Тип = 4), а таймер датчика встановлюється на постійне надсилання пакетів. Якщо пристрій IoT є виконавчим пристроєм, він може отримати керуюче повідомлення (Тип = 5) від сервера віртуалізації для виконання визначеної операції.

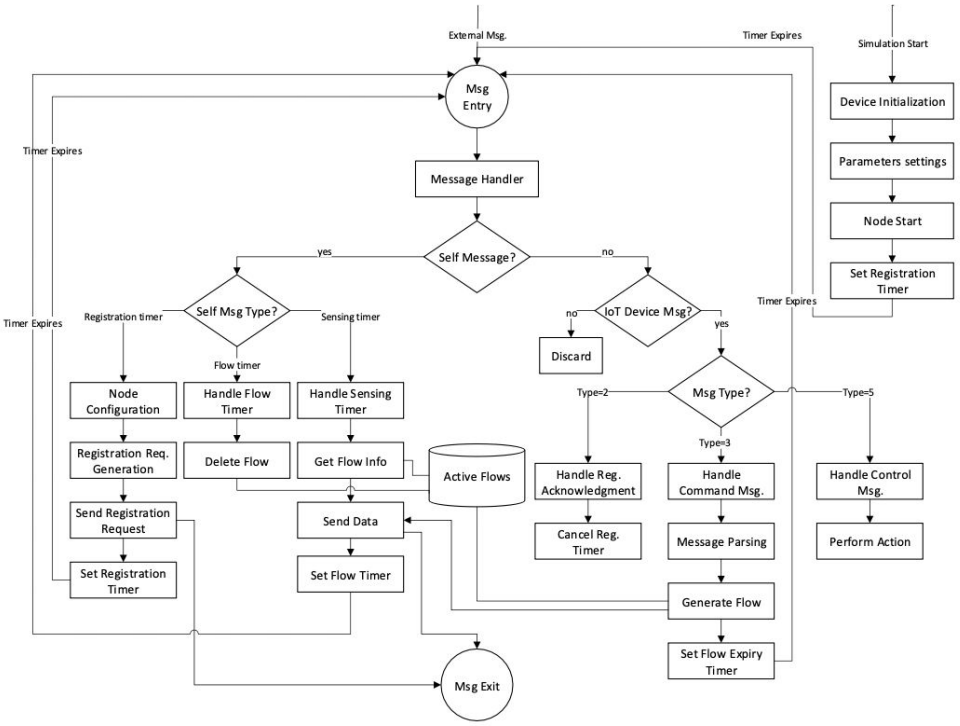


Рисунок 3.4 - Блок-схема протоколу «*IoTDeviceApp*»

Протокол VirtualizationServerApp

VirtualizationServerApp — це протокол прикладного рівня, розроблений для вузла сервера віртуалізації IoT. Логіка додатків цього протоколу реалізована мовою C++, а інформація про активні пристрої IoT зберігається в структурі даних. Активні пристрої IoT – це ті пристрої, які наразі використовуються в деякій віртуалізованій мережі IoT і мають активні потоки зв’язку. Також зберігається інформація профілю зареєстрованого пристрою IoT. Робоча схема цього протоколу наведена на малюнку 3.5. Під час запуску симуляції сервер віртуалізації виконує необхідні налаштування конфігурації, а потім очікує зовнішнього повідомлення від пристроїв IoT або клієнтів. На сервері віртуалізації очікуються три типи повідомлень (a) Повідомлення із запитом на реєстрацію (Тип = 1), яке призводить до створення віртуальних об’єктів для відповідних пристроїв Інтернету речей після необхідної перевірки з подальшим надсиланням повідомлення підтвердження; (b) повідомлення запиту від клієнтської програми (Тип = 10) для встановлення віртуалізованої мережі IoT шляхом вибору бажаних віртуальних об’єктів. Команда активації надсилається на вибраний пристрій IoT

для початку передачі даних. Клієнтській програмі надсилається підтвердження; (c) повідомлення зондування (Тип = 4) від активованих пристроїв IoT для подальшого пересилання на відповідний цільовий пристрій IoT.

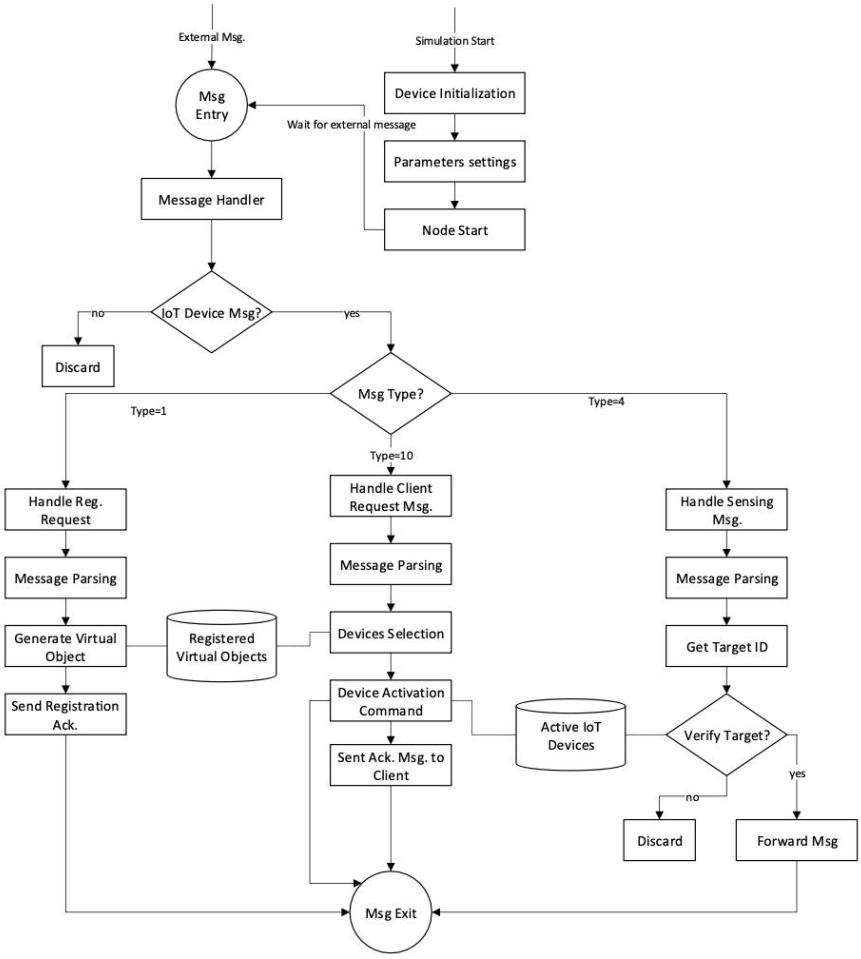


Рисунок 3.5 – Блок-схема протоколу «VirtualizationServerApp»

Для ілюстрації графічне зображення внутрішньої багаторівневої архітектури вузлів виділено на рисунку 3.6 (червоний прямокутник для сервера віртуалізації, синій прямокутник для клієнтського пристрою та жовтий прямокутник для вузла IoT). На сервері віртуалізації екземпляр протоколу VirtualizationServerApp використовується на прикладному рівні. Подібним чином екземпляри протоколу IoTClientApp і IoTDeviceApp використовуються на прикладному рівні клієнтського пристрою та вузлах IoT відповідно.

На початку пристрої IoT пов’язані з призначеними вузлами шлюзу. Потім усі пристрої IoT надсилають запити на реєстрацію на сервери віртуалізації через вузол шлюзу. Колір вузла IoT вказує на його статус реєстрації, тобто

• Червоний: запит ще не надіслано,

• Жовтий: запит надіслано,

• Зелений: зареєстровано.

Запит на реєстрацію містить відповідну інформацію профілю пристрою IoT. Після реєстрації сервер віртуалізації створює віртуальний об’єкт (VO) для кожного пристрою IoT, щоб зберігати інформацію про його профіль, яка також використовується для подальшого зв’язку та керування відповідними пристроями IoT. Знімок екрана, наведений на рисунку 3.6, зроблений після завершення процесу реєстрації всіх пристроїв IoT і повідомлення команди активації надсилається сервером віртуалізації на пристрій IoT. Крім того, тег на прикладному рівні також вказує, що на сервері віртуалізації зареєстровано 20 пристроїв IoT (зареєстровані VO = 20). Подібним чином тег на прикладному рівні пристрою IoT змінюється на Registered = yes, що вказує на успішне завершення його реєстрації на сервері. Після реєстрації сервери віртуалізації можуть отримувати запити від клієнтських пристроїв і надсилати різні команди зареєстрованим пристроям IoT за допомогою свого VO, наприклад, щоб отримати його робочий статус, ініціювати передачу даних тощо. Передача даних ініціюється через запит, надісланий клієнтом до VirtualizationServerApp за допомогою вказану частину у файлі omnetpp.ini, як показано на рисунку 3.7.

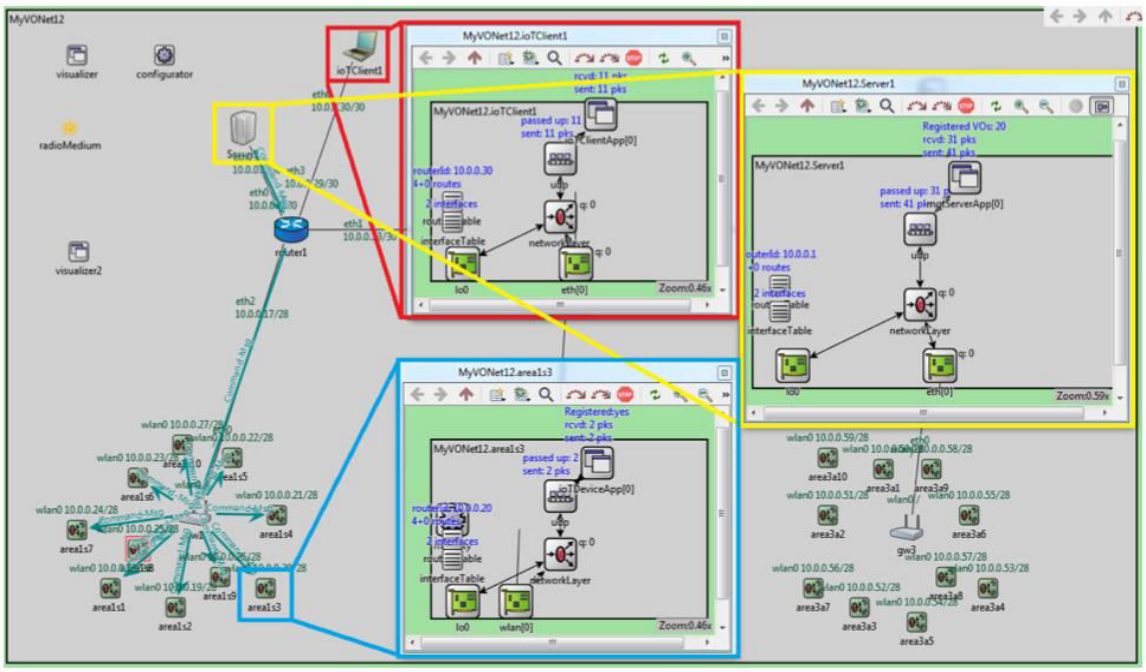


Рисунок 3.6 - Зображення внутрішньої багаторівневої архітектури вузлів

```
omnetpp.ini
##########################################################################################################
[Config VirtualIoTNet60-part5]
description = Testing Virtual IoT Network with 60 IoT devices, 01 Client and 01 Virtualization Server using One 2 One Scenarios
network = My_Virtual_IoT_Net

#Virtualization Server Configuration
*.Server1.numVirtualServerApps = 1
*.Server1.virtualServerApp[0].typename = "VirtualizationServerApp"
*.Server1.virtualServerApp[0].localPort=1000

*.Server1.virtualServerApp[0].destPort=0
*.Server1.virtualServerApp[0].messageLength=0B
*.Server1.virtualServerApp[0].sendInterval=0
*.Server1.virtualServerApp[0].dataRate=0
*.Server1.virtualServerApp[0].numActiveDevices=0

#IoT Client Device Configuration
*.ioTClient1.numClientApps = 1
*.ioTClient1.ioTClientApp[0].typename = "IoTClientApp"
*.ioTClient1.ioTClientApp[0].VirtualizationServerAddress="Server1"
*.ioTClient1.ioTClientApp[0].localPort=1000

*.ioTClient1.ioTClientApp[0].reqStartTime=2s
*.ioTClient1.ioTClientApp[0].reqStopTime=20s
*.ioTClient1.ioTClientApp[0].dataSendTo="One2One"
*.ioTClient1.ioTClientApp[0].reqDataRate=100
*.ioTClient1.ioTClientApp[0].reqNumActiveDevices=5
*.ioTClient1.ioTClientApp[0].reqMapping="VS01>VA01,VS02>VA02,VS03>VA03,VS04>VA04,VS05>VA05"

*.ioTClient1.ioTClientApp[0].destPort=0
*.ioTClient1.ioTClientApp[0].messageLength=0B
*.ioTClient1.ioTClientApp[0].sendInterval=0

#IoT Sensors Configuration
*.area1s*.numIoTApps =  1
*.area1s*.ioTDeviceApp[0].typename = "IoTDeviceApp"
*.area1s*.ioTDeviceApp[0].VirtualizationServerAddress="Server1"
*.area1s*.ioTDeviceApp[0].LocationName="Area1"
*.area1s*.ioTDeviceApp[0].localPort=1000
*.area1s*.ioTDeviceApp[0].DeviceClass="Sensor"
*.area1s*.ioTDeviceApp[0].DeviceType="Temperature"
*.area1s*.ioTDeviceApp[0].destPort=0
*.area1s*.ioTDeviceApp[0].messageLength=0B
*.area1s*.ioTDeviceApp[0].sendInterval=0

#IoT Actuators Configuration
*.area2a*.numIoTApps  =  1
*.area2a*.ioTDeviceApp[0].typename = "IoTDeviceApp"
*.area2a*.ioTDeviceApp[0].VirtualizationServerAddress = "Server1"
*.area2a*.ioTDeviceApp[0].LocationName="Area2"
*.area2a*.ioTDeviceApp[0].localPort=1000
*.area2a*.ioTDeviceApp[0].DeviceClass="Actuator"
*.area2a*.ioTDeviceApp[0].DeviceType="Fan"
*.area2a*.ioTDeviceApp[0].destPort=0
*.area2a*.ioTDeviceApp[0].messageLength=0B
*.area2a*.ioTDeviceApp[0].sendInterval=0
```

Рисунок 3.7 – Симуляція налаштувань віртуалізованого серверу

3.4 План моделювання та результати

У наступних підрозділах ми представляємо наші параметри моделювання та результати оцінки запропонованої схеми віртуалізації мережі IoT. Серед різних сценаріїв використання ми вибрали спрощений сценарій «один-на-один», де створюється віртуалізована мережа IoT для встановлення динамічного зв'язку «один-на-один» між вибраним датчиком і приводами. Детальне обговорення конфігурації вибраних сценаріїв і параметрів мережі представлено в наступному підрозділі.

Сценарій і параметри моделювання

Ми провели низку експериментів із фіксованою кількістю пристроїв IoT у мережі з різною швидкістю надсилання пакетів. На рисунку 3.8 показано сценарій моделювання, який використовується в цих експериментах. Ми розглянули два домени (A і B) з 30 пристроями IoT в кожному домені. Усі пристрої IoT у певному домені підключаються до сервера віртуалізації через локальний вузол шлюзу за допомогою бездротового зв'язку зі швидкістю передачі даних 54 Мбіт/с. Вузли шлюзу підключаються до сервера віртуалізації через дротове з'єднання зі швидкістю 100 Мбіт/с через проміжні маршрутизатори через Інтернет.

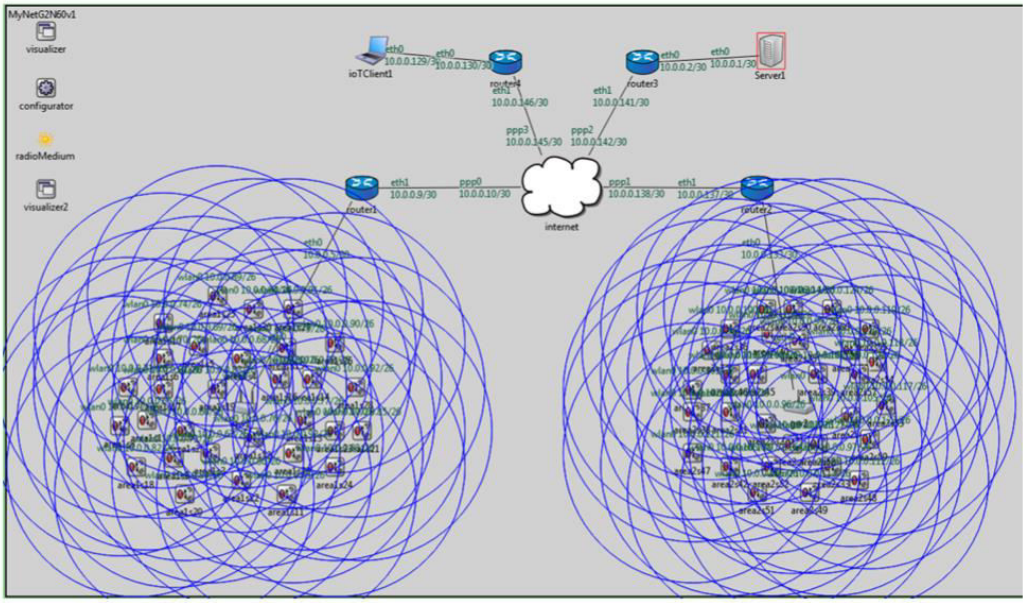


Рисунок 3.8 - Сценарій моделювання IoT мережі в OMNET++

Під час запуску симуляції всі пристрої IoT надсилають запит на реєстрацію, що містить інформацію про профіль, на попередньо налаштований сервер віртуалізації. Для кожного підключеного пристрою IoT створюються віртуальні об’єкти. Після цього клієнтський вузол «ioTClient1» надсилає запит на віртуальний сервер для створення віртуалізованої мережі IoT. Бажана топологія клієнта створюється шляхом створення віртуальної мережі IoT серед вибраних віртуальних об’єктів. У цих експериментах простий сценарій використання моделюється шляхом створення п’яти динамічних з’єднань (один-до-одного) між п’ятьма парами датчиків і приводів. Датчики з домену A підключаються до виконавчих механізмів у домені B. Після цього датчики активуються, щоб розпочати надсилання даних із заданою швидкістю передачі даних, які отримують відповідні віртуальні об’єкти, а потім пересилаються до пов’язаних виконавчих механізмів через відображення пристроїв. Створюється п’ять потоків для встановлення індивідуального зв’язку між датчиками від домену A до приводів у домені B. Результати збираються з різною швидкістю надсилання даних 80, 100 і 120 пакетів/с на потік. Розмір пакета, який використовується в цих експериментах, становить 1000 байт. Таблиця 3.2 представляє конфігурацію для різних параметрів, які використовуються в моделюванні.

Ми вибрали три параметри для аналізу продуктивності запропонованої системи шляхом моделювання, тобто наскрізну затримку, пропускну спроможність і коефіцієнт доставки пакетів, щоб вивчити вплив змінної швидкості надсилання даних на ці показники. Наскрізна затримка обчислюється шляхом вимірювання загального часу, необхідного для отримання пакетів даних від датчика та надсилання команди активації пов’язаному виконавчому пристрою. Іншими словами, наскрізна затримка - це різниця між часом, коли було згенероване повідомлення зондування, і часом, коли команда приведення в дію була отримана цільовим виконавчим пристроєм. Пропускна здатність – це загальна кількість даних, отриманих цільовими пристроями IoT за одиницю часу з усіх джерел, тобто активних пристроїв IoT. Коефіцієнт доставки пакетів — це відсоткове співвідношення між загальною кількістю отриманих пакетів даних і загальною кількістю пакетів даних, згенерованих у мережі.

Таблиця 3.2

Параметри симуляції

Parameter	Value/Range		
	IoT Device(s)	**Virtualization Server**	**Client Device**
Nodes count	60	1	1
Application-Layer	*IoTDeviceApp*	*VirtualizationServerApp*	*IoTClientApp*
Size of Packet	1000 Bytes	N/A	N/A
Sending rate of Packets (per node)	80, 100, 120 pkts/s	N/A	NA
Transport-Layer		UDP Protocol	
Routing-Layer		Static IP Protocol	
MAC-Layer	IEEE 802.11	Ethernet	Ethernet
Bit-Rate	54 Mbps	N/A	N/A
Wireless Communication Range	100 m	N/A	N/A
Size of Area		1000 m^2	
Nodes Mobility		Static	
Simulation Duration		20 s	

Результати

Вплив збільшення розміру мережі на пропускну здатність можна оцінити двома різними способами: (a) збільшити швидкість передачі даних, тобто пакетів/с, зберігаючи кількість джерел незмінною; (b) збільшення кількості джерел, тобто активних пристроїв IoT, щоб генерувати більше даних у мережі. Ми використали перший підхід у цьому дослідженні для оцінки продуктивності мережі з точки зору пропускної здатності. Для цих експериментів ми розглянули фіксовану мережу з 60 вузлів із п’ятьма джерелами, тобто активними пристроями IoT. Ті самі експерименти повторювали (п’ять разів) із різними швидкостями передачі даних 80, 100 і 120 пакетів/с, а середні результати щодо пропускної здатності та наскрізної затримки наведені на рисунку 3.9 і 3.10. Ці результати показують спостережувану варіацію в пропускна здатність і затримка під час моделювання. З п’ятьма джерелами, швидкістю передачі даних 80 пакетів/с і розміром пакета 1000 байт загальна кількість даних, згенерованих у мережі, становить 5×80×1000×8÷106 =3,2 Мбіт/с.

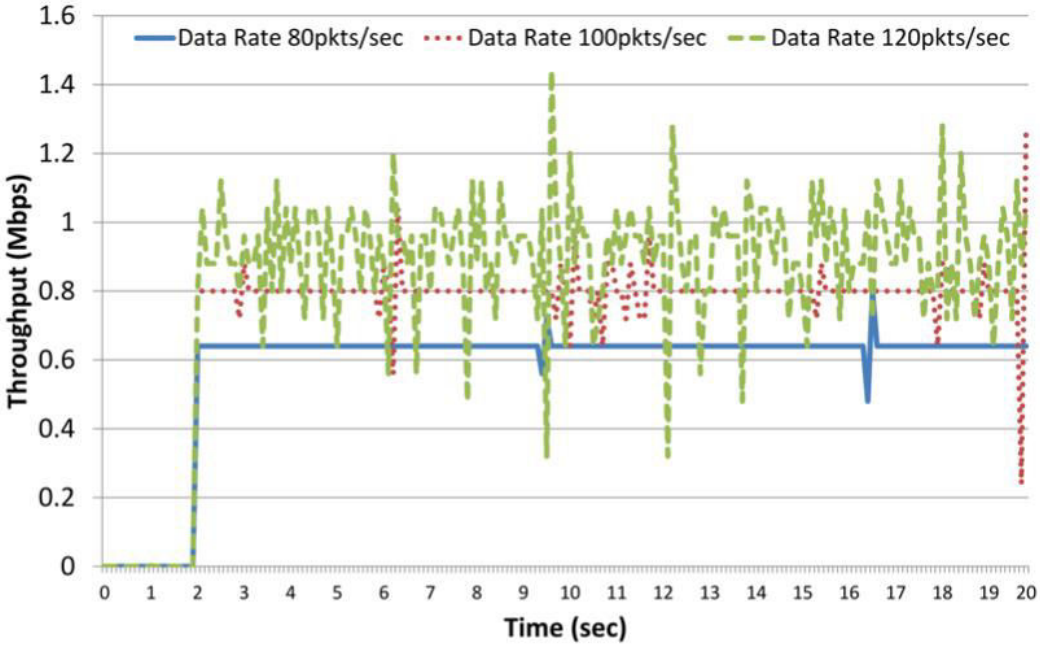


Рисунок 3.9 – Проміжна здатність

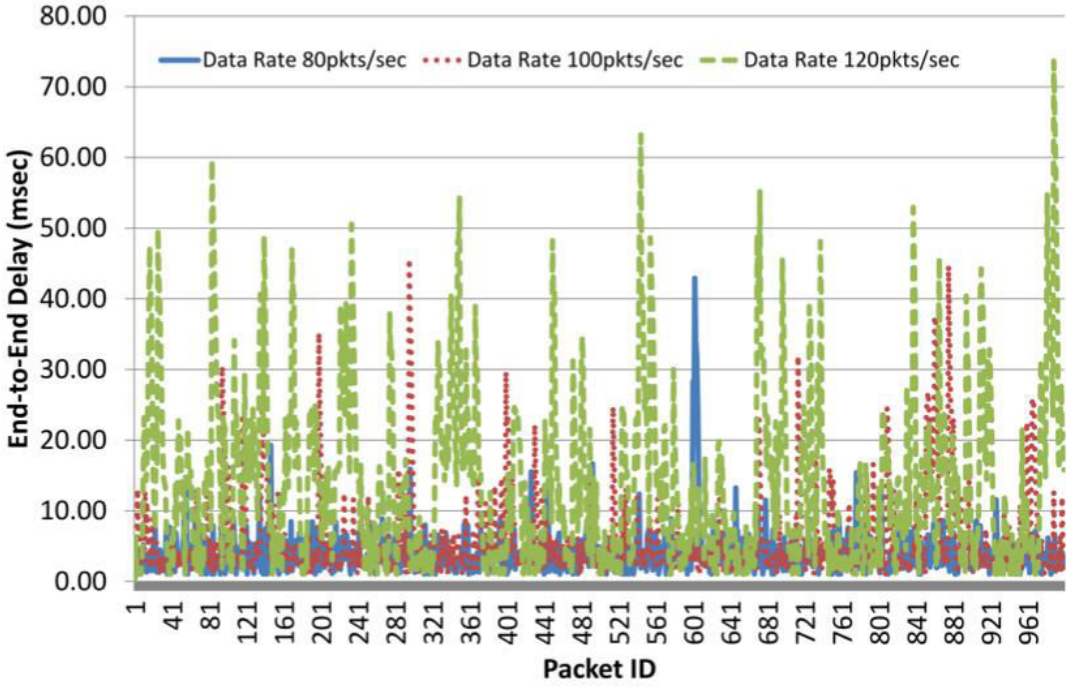


Рисунок 3.10 – Затримка у мсек

Середнє значення пропускної здатності, обчислене на всіх цільових пристроях IoT за допомогою моделювання, також становило близько 3,2 Мбіт/с зі стандартним відхиленням 0,02 Мбіт/с, як показано в таблиці 5. Це означає, що при швидкості передачі даних 80 пакетів/с коефіцієнт доставки пакетів у мережі

становила 100%, що підтверджується графіком коефіцієнта доставки пакетів, наведеним на рисунку 3.11.

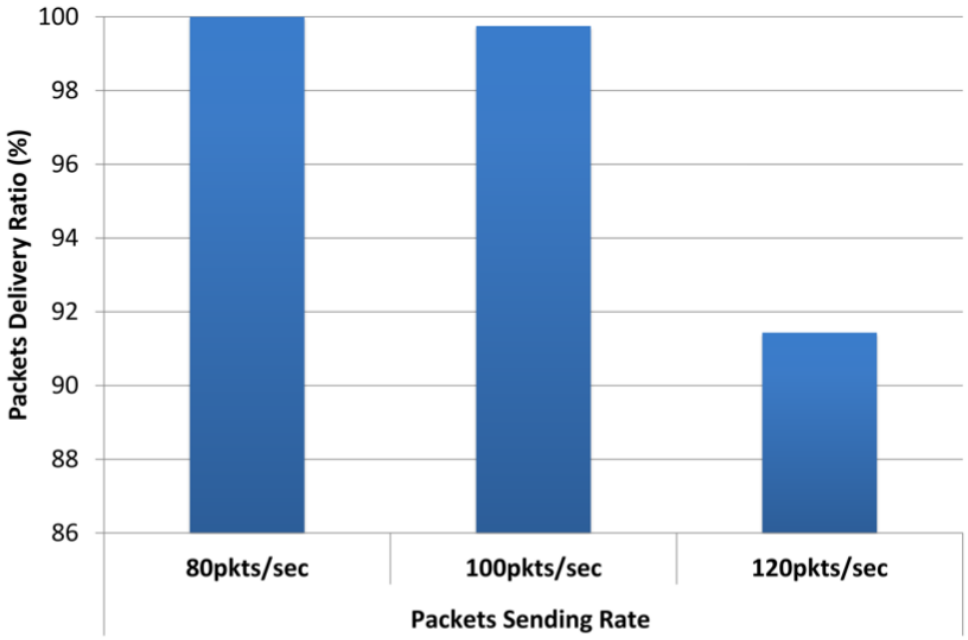


Рисунок 3.11 – Відсоток пакетів що дійшли

Крім того, результати пропускної спроможності з 80 пакетами/с є досить стабільними, і немає жодних змін, що свідчить про безперебійну доставку всіх пакетів даних. Незначні варіації результатів наскрізної затримки для швидкості надсилання даних 80 пакетів/с, як показано на рисунку 3.12, також вказують на те, що в мережі немає перенавантаження. Середнє значення наскрізної затримки для швидкості надсилання даних 80 пакетів/с було зареєстровано як 3,94 мс зі стандартним відхиленням 3,25 мс. Коли швидкість надсилання пакетів була збільшена, тобто 100 пакетів/с, ми отримали пропускну здатність близько 3,99 Мбіт/с, тоді як дані, згенеровані в мережі, становили 4 Мбіт/с. Зміни в результатах пропускної здатності для швидкості надсилання даних 100 пакетів/с, також показують, що мережа стає перевантаженою, і коефіцієнт доставки пакетів дещо знижується. Крім того, результати наскрізної затримки для 100 пакети/с, також мають більші варіації порівняно з результатами затримки для 80 пакетів/с. Середнє значення наскрізної затримки для швидкості надсилання даних 100 пакетів/с було зареєстровано як 6,38 мс зі стандартним відхиленням 4,51 мс, що майже на 60% більше порівняно із затримкою зі швидкістю надсилання даних 80

пакетів/с. с. Крім того, збільшення швидкості передачі даних до 120 пакетів/с призводить до середньої пропускної здатності 4,39 Мбіт/с, тоді як дані, що генеруються в мережі, становлять 4,8 Мбіт/с. Спостерігаються більш високі коливання в результатах пропускної здатності. Це вказує на перевантаження мережі, показано значне погіршення коефіцієнта доставки пакетів із збільшенням швидкості надсилання даних. Таким чином, більш високі коливання були зареєстровані в результатах наскрізної затримки для 120 пакетів/с. Середнє значення наскрізної затримки для швидкості надсилання даних 120 пакетів/с було зареєстровано як 13,46 мс з стандартне відхилення становить 8,38 мс, що є збільшенням майже на 240% і 110% порівняно із затримкою зі швидкістю надсилання даних 80 і 100 пакетів/с. Статистичний підсумок результатів пропускної здатності та наскрізної затримки представлено в таблицях 3.3 і 3.4 відповідно.

Таблиця 3.3

Статистика проміжної здатності

Table 5. Statistical results of throughput (Mbps).

	Packet Sending Rate		
Statistics	**80 pkts/s**	**100 pkts/s**	**120 pkts/s**
Mean	3.2	4	4.39
Stdev	0.02	0.07	0.21
Min.	0.48	0.24	0.32
Max.	0.8	1.28	1.44

Таблиця 3.4

Статистика затримки

	Packets Sending Rate		
Statistics	**80 pkts/s**	**100 pkts/s**	**120 pkts/s**
Mean	3.94	6.38	13.46
Stdev	3.25	4.51	8.38
Max.	42.9	45.67	73.65
Min.	0.13	1.05	0.13

Під час дослідження було виявлено з файлів трасування моделювання, що зі збільшенням швидкості надсилання даних більшість пакетів відкидається на MAC-рівні вузлів датчиків IoT. Пакети відкидаються, оскільки вузли не можуть отримати доступ до спільного бездротового каналу через конкуренцію. Це пов’язано з тим, що всі пристрої IoT використовують один вузол шлюзу, який

перевантажений. Результати моделювання показують, що зі збільшенням розміру мережі та навантаження на дані шлюзовий вузол стає вузьким місцем продуктивності. Для подальшої перевірки цього твердження ми провели інший набір моделювання з різними вузлами шлюзу для подібних умов мережі. Зі збільшенням швидкості надсилання пакетів результати коефіцієнта доставки пакетів погіршуються для одного вузла шлюзу. Значне покращення коефіцієнта доставки пакетів можна спостерігати, коли використовуються кілька вузлів шлюзу для швидкості передачі даних 120 пакетів/с. Подібним чином середнє значення наскрізної затримки зменшується з 13,46 мс (з одним вузлом шлюзу) до 6,82 мс і 4,12 мс (з двома і трьома вузлами шлюзу), відповідно, як показано на рисунку 3.13. Іншими словами, значне зменшення наскрізної затримки спостерігається з декількома вузлами шлюзу, тобто зменшення на 49,33% і 69,39% з двома і трьома вузлами шлюзу, відповідно, порівняно з результатами для одного вузла шлюзу. В екстремальних умовах мережі нам, можливо, доведеться пожертвувати меншою кількістю бітів, що насправді нічого не варте в порівнянні з гнучкістю та контролем, які пропонує запропонована система над пристроями IoT.

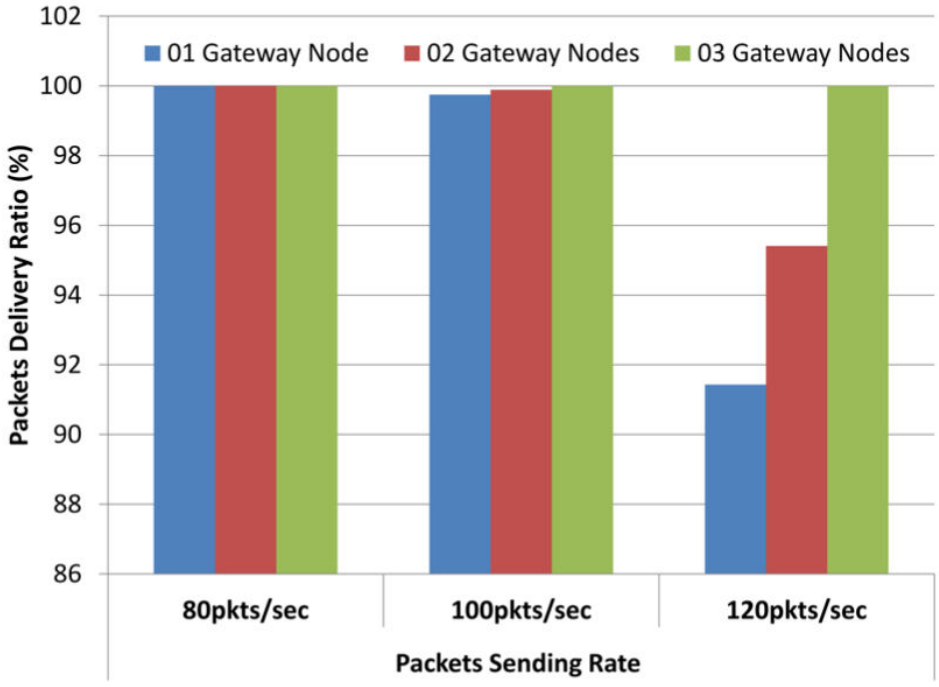


Рисунок 3.12 – Відсоток доставлених пакетів з різними шлюзами

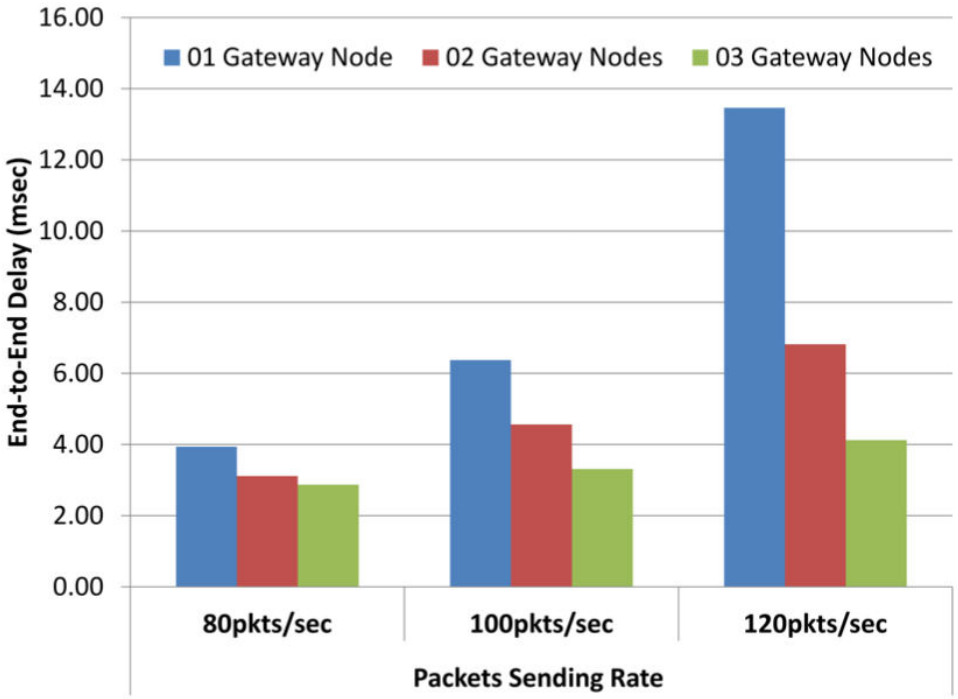


Рисунок 3.13 – Затримка з різними шлюзами

Аналіз вимог до ресурсів віртуальних об'єктів

Структура профілю віртуального об'єкта залишиться тією самою; однак вимоги до пам'яті можуть відрізнятися залежно від записаних властивостей зареєстрованих пристроїв IoT. Щоб вивчити вимоги до пам'яті на серверах віртуалізації для зареєстрованих віртуальних об'єктів, ми провели ще один набір експериментів зі збільшенням розміру мережі. У цих експериментах ми розглядаємо сім різних типів пристроїв IoT, тобто двигун, камеру, датчик температури, датчик осі, зелений світлодіод, звуковий сигнал, мотор і датчик світла, де потреба в пам'яті для профілю віртуального об'єкта кожного типу пристрою становить 154, 170, 122. , 122, 122, 122, 106 байт відповідно. Результати збираються в результаті кількох симуляцій із зростаючою кількістю пристроїв IoT від 200 до 1000 (розмір кроку 200). У кожному експерименті пристрої IoT вищезазначених типів створювалися шляхом рівномірного розподілу. Після завершення процесу реєстрації всіх пристроїв IoT на сервері віртуалізації ми зафіксували вимоги до пам'яті для зберігання інформації профілю віртуальних об'єктів, і середні результати представлені на рисунку 3.14.

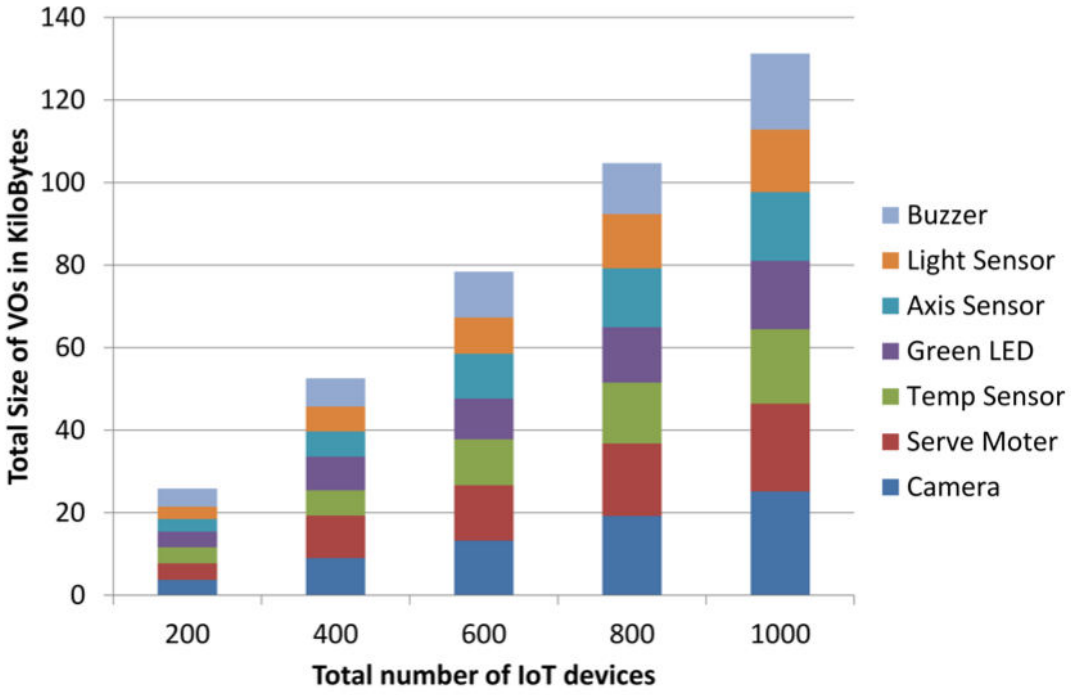


Рисунок 3.14 – Аналіз вимог до пам’яті з збільшенням кількості пристроїв IoT

Можна помітити, що зі зростанням кількості пристроїв IoT у мережі вимоги до пам’яті на сервері віртуалізації також демонструють лінійне зростання. Однак, залежно від типу операції, яку ми хочемо виконати, вимоги до обчислень будуть абсолютно різними. У цих експериментах ми розглядали прості операції зі складністю обчислень від O(n) до O(n2). У майбутньому ми впроваджуватимемо складні операції, які потребуватимуть більше обчислювальних ресурсів, а звичайні підходи не будуть масштабованими, якщо виконувати їх у великому масштабі з мільйонами зареєстрованих пристроїв IoT. Для цього, безумовно, знадобляться інтелектуальні та розподілені рішення.

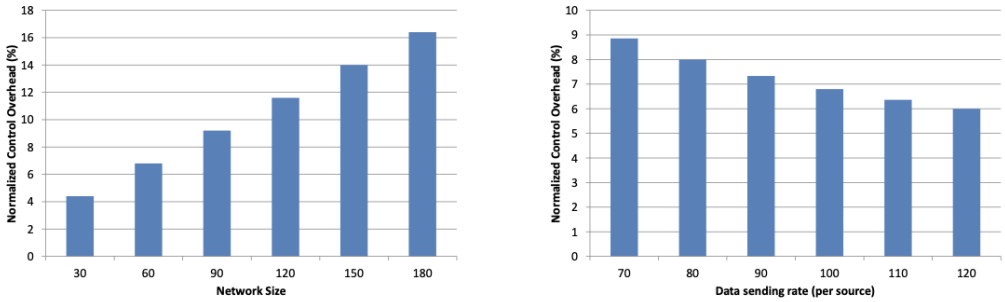
Аналіз контрольних витрат

Запропонована система генерує різні пакети керування в мережі, які включають пакети початкової реєстрації, командне повідомлення для ініціювання та завершення різних активних потоків у мережі та періодичні повідомлення запиту статусу для зареєстрованих пристроїв IoT. Щоб проаналізувати накладний трафік через централізований сервер віртуалізації, ми провели кілька наборів

експериментів. Експерименти проводяться з різними швидкостями надсилання даних і розміром мережі. Ми використали нормовану метрику контрольних витрат (NCO) для кількісної оцінки мережевих витрат запропонованої системи. NCO — це співвідношення між загальною кількістю контрольних пакетів і загальною кількістю пакетів даних, згенерованих у мережі. Математично NCO можна обчислити так:

$$NormalizedControlOverhead(\%) = \frac{Controlpackets}{Datapackets} \times 100. \quad (3.1)$$

Як зазначено в рівнянні (1), цей показник дає нам витрати в контрольних пакетів, необхідних для створення 100 пакетів даних. Пакети керування, створені в мережі, в основному залежать від кількості активних потоків і розміру мережі, тоді як NCO також залежить від двох факторів, тобто пакетів даних і пакетів керування. NCO збільшиться, якщо розмір мережі зростатиме без збільшення швидкості надсилання даних або активних потоків. Таку саму поведінку можна спостерігати в експериментальних результатах, представлених на рисунку 3.15a. Швидкість надсилання даних і кількість активних вихідних вузлів залишаються фіксованими; тому пакети даних, згенеровані у всій мережі, залишаються незмінними. Однак кількість мережевих контрольних пакетів зростатиме зі збільшенням розміру мережі. Таким чином, зі збільшенням розміру мережі спостерігалося лінійне зростання NCO. У мережі розміром 180 пристроїв IoT зареєстроване значення NCO, тобто 16 (приблизно), вказує на те, що запропонована система генерує близько 16 контрольних пакетів на кожні 100 пакетів даних. Однак із зростанням розміру мережі також очікується збільшення швидкості передачі даних, що потім компенсує збільшення витрат на контроль. Це твердження підтверджується ще одним набором експериментів із мережею з 60 пристроїв IoT і зростаючою швидкістю надсилання даних. Більше пакетів даних вводяться в мережу зі збільшенням швидкості надсилання даних і незмінною кількістю контрольних пакетів. Це призводить до зменшення NCO, як показано на рисунку 3.15 b.



(**a**) Normalized control overhead with growing network size. (**b**) Normalized control overhead with increasing data rate.

Рисунок 3.15 – Результати симуляції з п’ятьма активними вузлами що надсилають дані (100 пакетів у секунду)

ВИСНОВКИ

В магістрській роботі проаналізовано можливість віртуалізації мереж IoT. Описано основи понять віртуалізації, розглянуто перспективи застосування даної технології в мережах IoT.

Ця робота представляє ідею віртуалізації мережі IoT у хмарному середовищі. Проведено аналіз продуктивності запропонованої системи, реалізувавши два протоколи прикладного рівня в симуляторі OMNeT++. Проводяться різні експерименти з різними розмірами мережі та швидкістю надсилання пакетів. Результати моделювання показують, що коли мережевий трафік перевищує певний поріг, вузькі вузли стають вузьким місцем продуктивності. Використання кількох вузлів шлюзу може значно підвищити продуктивність мережі в таких умовах. Інші рішення включають удосконалення протоколів бездротового підключення для пристроїв IoT. Заторів можна уникнути, динамічно регулюючи швидкість надсилання даних датчиків за допомогою оптимізації на сервері віртуалізації. У запропонованій системі ми використали прості віртуальні об’єкти для створення віртуальної мережі IoT шляхом виконання операції на основі правил.

ПЕРЕЛІК ПОСИЛАНЬ

1. Miorandi, D.; Sicari, S.; De Pellegrini, F.; Chlamtac, I. Internet of things: Vision, applications and research challenges. Ad Hoc Netw. 2012, 10, 1497–1516.

2. Madria, S.; Kumar, V.; Dalvi, R. Sensor cloud: A cloud of virtual sensors. IEEE Softw. 2014, 31, 70–77.

3. Giezeman, W.; Stokking, J. The Things Network, 2016. Посилання: https://www.thethingsnetwork.org/

4. Bhattacharya, A.; Fernando, M.S.; Dasgupta, P. Community Sensor Grids: Virtualization for sharing across domains. In Proceedings of the First Workshop on Virtualization in Mobile Computing, Breckenridge, CO, USA, 17 June 2008; 49–54.

5. Vaidya M (2016) Handling critical issues of big data on cloud, pp 100–131. https://doi.org/10.4018/978-1-4666-9834-5.ch005

6. Zhang Q, Cheng L, Boutaba R (2010) Cloud computing: state-of-the- art and research challenges. J Internet Serv Appl 1:7–18. https://doi.org/10.1007/s13174-010-0007-6

7. Vaquero L, Rodero-Merino L, Caceres J, Lindner M (2009) A break in the clouds: Towards a cloud definition. Comput Commun Rev 39:50–55. https://doi.org/10.1145/1496091.1496100\

8. Dash SK, Sahoo JP, Mohapatra S, Pati SP (2012) Sensor-cloud: assimi- lation of wireless sensor network and the cloud. In: Meghanathan N, Chaki N, Nagamalai D (eds) Advances in computer science and information technology, networks and communications. Springer, Berlin, pp 455–464

9. Mosharaf Kabir Chowdhury NM, Boutaba R (2009) Network virtual- ization: state of the art and research challenges. IEEE Commun Mag 47(7):20–26. https://doi.org/10.1109/MCOM.2009.5183468

10. Merentitis A, Zeiger F, Huber M, Frangiadakis N, Mathioudakis K, Sasloglou K, Mazarakis G, Gazis E, Boufidis Z (2013) Wsn trends: sensor infrastructure virtualization as a driver towards the evolution of the internet of things

11. Cao QH, Khan I, Farahbakhsh R, Madhusudan G, Lee GM, Crespi N (2016) A trust model for data sharing in smart cities. In: 2016 IEEE international conference on communications (ICC), pp 1–7. https://doi.org/10.1109/ICC.2016.7510834

12. Rashid A, Chaturvedi A (2019) Virtualization and its role in cloud computing environment. Int J Comput Sci Eng 7:1131–1136. https://doi.org/10.26438/ijcse/v7i4.11311136

13. Othman MF, Shazali K (2012) Wireless sensor network applications: a study in environment monitoring system. Proced Eng 41:1204 – 1210. https://doi.org/10.1016/j.proeng.2012.07.302

14. Carlos-Mancilla M, López-Mellado E, Siller M (2016) Wireless sen- sor networks formation: approaches and techniques. J Sens 2:25

15. Abdul-Qawy ASH, Srinivasulu T (2018) SEES: a scalable and energy- efficient scheme for green IoT-based heterogeneous wireless nodes. J Ambient Intell Human Comput 20:1–26

16. Wang T, Lu Y, Cao Z, Shu L, Zheng X, Liu A, Xie M (2019) When sensor-cloud meets mobile edge computing. Sensors (Basel, Switzerland) 19:20

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
Кафедра Інженерії програмного забезпечення автоматизованих систем

Магістрська робота
На тему:
«Застосування хмарної мережевої віртуалізація для динамічного підключення пристроїв в мережі IoT**»**

Студента гр.ІСДМ-61
Пешкова І.О.

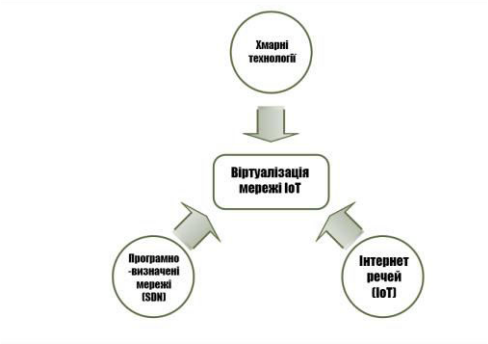
ЗАВДАННЯ РОБОТИ

Мета роботи – аналіз ефективності застосування хмарної мережевої віртуалізації для динамічного підключення пристроїв в мережі IoT
Для досягнення поставленої мети в роботі виконано наступні завдання:
1. Проведення аналіз існуючих рішень по віртуалізації IoT
2. Дослідження можливостей віртуалізації мереж IoT
3. Проведення моделювання мереж віртуальних об'єктів для оцінки ефективності запропоновоного рішення віртуалізації.

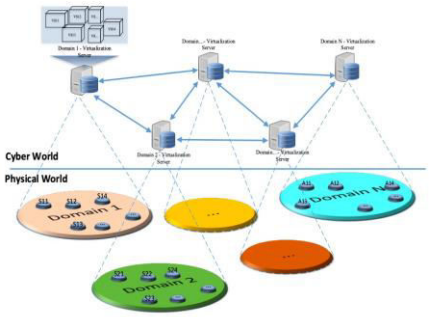
Об'єкт дослідж ення – процес підключення пристроїв в мережі IoT
Предмет дослідж ення – хмарна мережева віртуалізація для підключення пристроїв в мережі IoT

Наукова новизна одерж аних результатів. У магістрській роботі представляємо детальний дизайн системи для побудови віртуальної мережі IoT.
Практична значущість одержаих результатів. Отримані результати можуть бути використані при побудові ефективних мереж IoT з елементами віртуалізації.

Аналіз передумов створення віртуальних мереж IoT



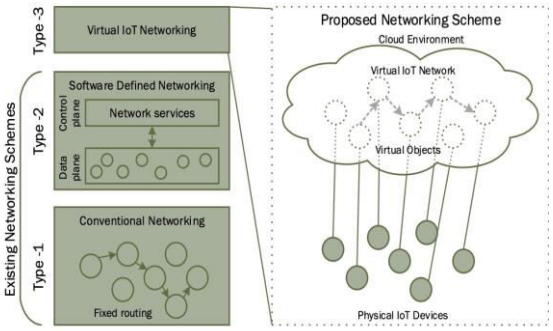
Три найбільш релевантні сфери, які призвели до розвитку віртуалізованих мереж IoT



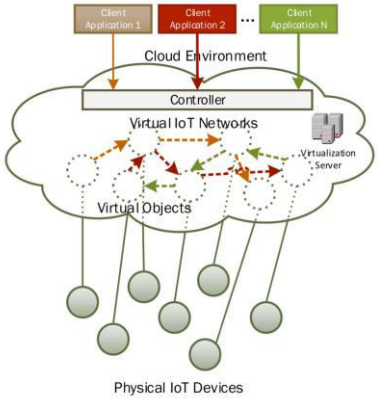
Логічна мережа серед серверів віртуалізації для різних доменів для підтримки спільного використання ресурсів для надання скоординованих послуг клієнтським програмам для майбутнього розумного світу

3

Структура віртуальної IoT мережі



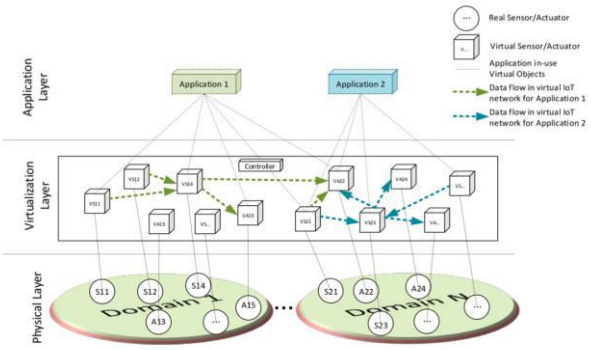
Концептуальний вигляд формування мережі віртуального IoT



Детальне уявлення про створення віртуалізованої мережі IoT для різноманітних програм у хмарному середовищі

4

Формування віртуальної мережі IoT на рівні віртуалізації для двох різних клієнтів



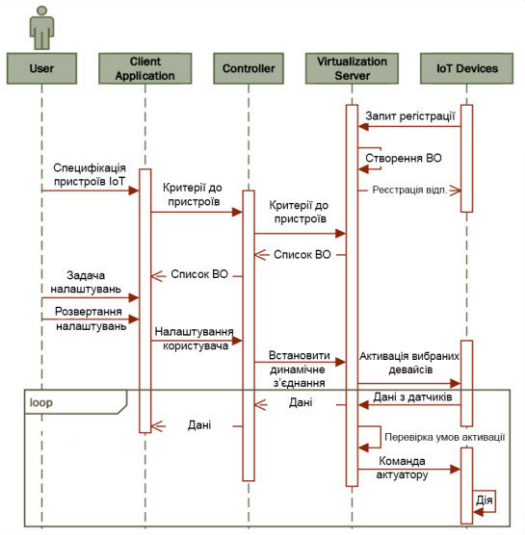
1. Фізичний рівень складається з фактичних пристроїв IoT, які можуть бути сенсорними та керуючими пристроями, підключеними до системи.

2. Рівень віртуалізації діє як проміжне програмне забезпечення між клієнтськими програмами та фізичними пристроями IoT

3. Різні додатки розміщені на прикладному рівні для споживання послуг базового рівня віртуалізації.

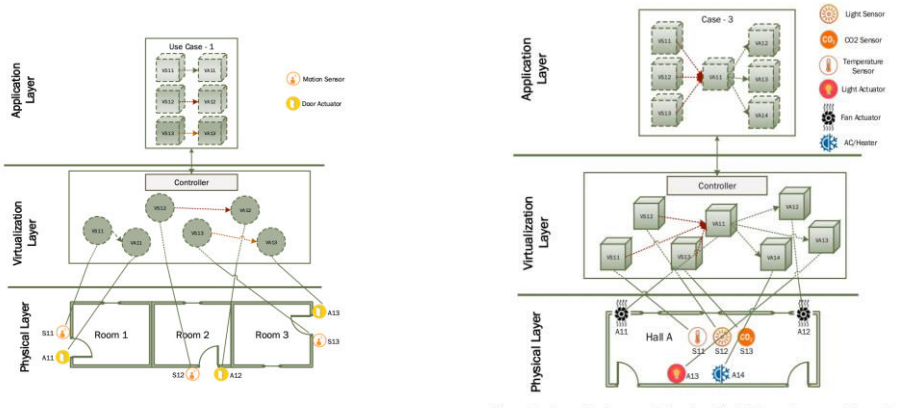
5

Представлення основних компонентів запропонованої системи для віртуалізації мережі IoT та її функціонування у вигляді діаграми послідовності



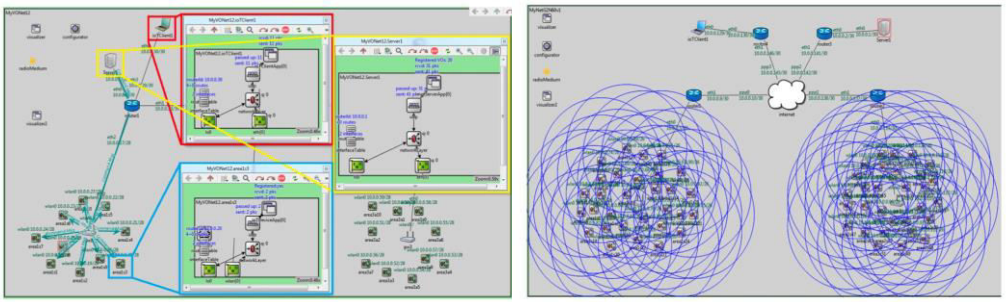
6

Схема досліджуємої системи автоматизації житлового приміщення



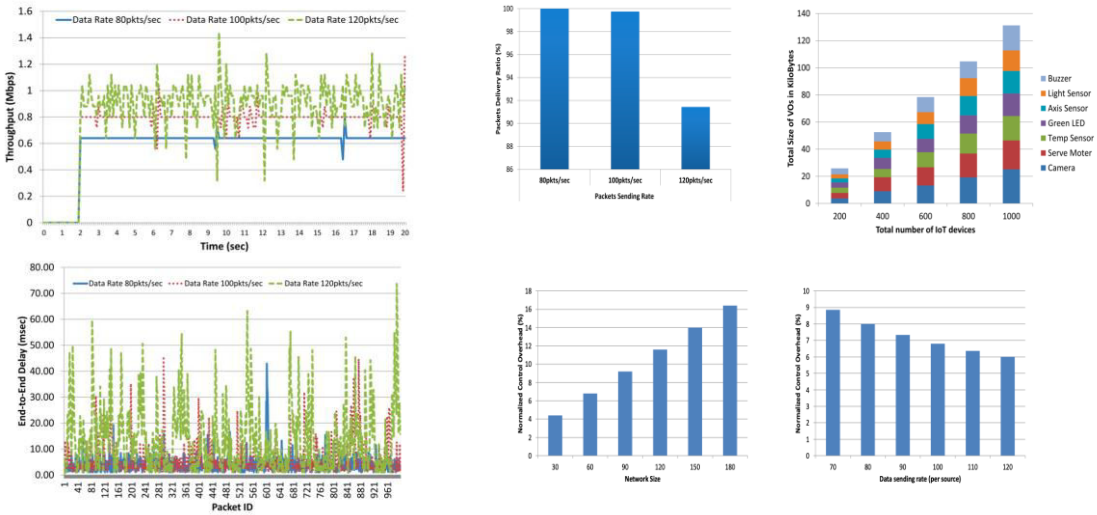
7

Мережа в OMNeT++



8

Результати моделювання



9

Висновки

В магістрській роботі

1. Досліджено можливість віртуалізації мережі IoT у хмарному середовищі.
2. Проведено аналіз продуктивності запропонованої системи, реалізувавши протоколи прикладного рівня в симуляторі OMNeT++.
3. Проведено різні експерименти з різними розмірами мережі та швидкістю надсилання пакетів. Результати моделювання показують, що коли мережевий трафік перевищує певний поріг, вузли стають вузьким місцем продуктивності. Використання більшої кількості вузлів може значно підвищити продуктивність мережі в таких умовах.