Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
(повна назва інституту/факультету)

Кафедра телекомунікацій
(повна назва кафедри)

«На правах рукопису»
УДК _____

До захисту допущено
В.о. завідувача кафедри

_____ Явіся В.С.___
(підпис) (ініціали, прізвище)

“___”_____2019_р.

Магістерська дисертація

на здобуття освітнього ступеня «магістр»

Спеціальність 172 Телекомунікації та радіотехніка,
(код і назва)

За освітньо-професійною програмою Інженерія та програмування інфокомунікацій.

на тему: Розробка відмовостійкої архітектури для IoT платформи.

Виконав: студент II курсу, групи ТЗ-81мп
(шифр групи)

Дуля Олександр Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник професор каф. ТК, д.т.н., академік НАНУ Ільченко М.Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант 1, 2, 3 доцент каф. ТК, к.т.н., с.н.с. Міночкін Д.А. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент професор каф. ІТМ, д.т.н., с.н.с. Скулиш М.А._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2019 рік

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

Інститут телекомунікаційних систем
(повна назва)

Кафедра телекомунікацій
(повна назва)

Спеціальність 172 Телекомунікації та радіотехніка
(код і назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою Інженерія та програмування інфокомунікацій.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____Явіся В.С.
(підпис) (ініціали, прізвище)
«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Дулі Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Розробка відмовостійкої архітектури для IoT платформи.

науковий керівник дисертації професор каф. ТК, д.т.н., академік НАНУ Ільченко Михайло Юхимович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_07_» «_11_» 2019р. № _3840-с_

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження архітектура IoT платформи

4. Предмет дослідження відмовостійке архітектурне рішення для IoT платформи

5. Перелік завдань, які потрібно розробити
– _____ Ан
аліз сучасної IoT платформи;
– _____ ана
ліз існуючих архітектурних рішень для IoT платформи;

– _____ проектування та оцінка ефективності власного архітектурного рішення;

– _____ ана
ліз отриманих результатів для формулювання рекомендацій по використанню запропонованої архітектури.

6. Орієнтовний перелік ілюстративного матеріалу

1) _____ Те
ма, мета та завдання магістерської дисертації
2) _____ Заг
альні відомості про традиційну IoT архітектуру;
3) _____ Ан
аліз хмарної платформи;
4) _____ Роз
роблене відмовостійке архітектурне рішення;
5) _____ Пр
инцип роботи архітектурного рішення;
6) _____ Ан
аліх отриманих результатів;
7) _____ Ви
сновки

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Доцент Міночкін Д.А.	26.12.2019	27.03.2019
2	Доцент Міночкін Д.А.	27.03.2019	20.09.2019
3	Доцент Міночкін Д.А.	20.09.2019	20.11.2019

9. Дата видачі завдання 20.09.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Розробка, оформлення, узгодження та затвердження технічного завдання на дипломну роботу	20.09.2018	Виконано
2	Опрацювання літературних джерел з теми досліджень	26.12.2018	Виконано
3	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	18.01.2019	Виконано
4	Вивчення та аналіз сучасної IoT платформи.	27.03.2019	Виконано
5	Аналіз хмарних платформ для IoT платформи	18.06.2019	Виконано
6	Розробка відмовостійкого архітектурного рішення	20.09.2019	Виконано

7	Аналіз отриманих результатів	20.11.2019	Виконано
9	Оформлення пояснювальної записки	07.12.2019	Виконано

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Робота містить 70 сторінок, 9 рисунків, 1 таблицю. Було використано 18 джерел інформації.

Актуальність. Протягом останніх десятиліть інформаційні та комунікаційні технології намагалися постійно збільшувати кількість інтернет-пристроїв. Окрім традиційних комп'ютерів та мобільних пристроїв, - це пристрої що варіюються від домашньої або побутової техніки (телевізор, холодильник, кавоварка, пилосмок), промислового обладнання та автоматики, охорони здоров'я, транспорту, енергії, будівель. Додавання інших девайсів, які традиційно були поза мережею інтернет стало можливим завдяки технологічному прогресу за допомогою обладнання та нового програмного забезпечення. Розроблюване відмовостійке та екномічне архітектурне рішення для IoT платформи полегшить розробку нових систем та мереж інтернету речей.

Метою роботи є підвищення відмовостійкості традиційної архітектри для IoT платформи шляхом її удосконалення.

Для досягнення поставленої мети в роботі вирішуються наступні задачі:

- аналіз сучасної IoT платформи;
- дослідження існуючих архітектурних рішень для IoT платформи;
- аналіз хмарної платформи для побудови IoT інфраструктури;
- проектування та валідація власного архітектурного рішення;
- аналіз отриманих результатів для формулування рекомендацій по використанню запропонованої архітектури.

Об'єкт дослідження – архітектура IoT платформи.

Предмет дослідження – відмовостійке архітектурне рішення для IoT платформи.

Ключові слова: Інтернет речей, IoT платформа, хмарна платформа, відмовостійке архітектурне рішення, Google Cloud Platform

ABSTRACT

The work contains 70 pages, 9 illustrations, 1 table, 18 sources of information were used.

Topicality. Over the last decades, information and communication technologies have sought to steadily increase the number of Internet devices. Except for traditional computers and mobile devices, these are devices that range from home appliances (TV, refrigerator, coffee maker, vacuum cleaner), industrial equipment and automation, healthcare, transportation, energy, buildings. Adding other devices that have traditionally been offline has been made possible by technological advances with the help of equipment and new software. Developed fault-tolerant and cost-effective architectural solution for the IoT platform will facilitate the development of new systems and Internet of Things network.

The purpose of the work is to increase the resiliency of traditional architecture for the IoT platform by improving it.

To achieve this goal, the following tasks are solved in the work:

- analysis of modern IoT platform;
- research on existing architectural solutions for the IoT platform;
- cloud platform analysis for building IoT infrastructure;
- development and validation of my own architectural solution;
- analysis of the results obtained to formulate recommendations for the use of the architecture solution.

The object of study is a architecture of the IoT platform.

The subject of study is the fault-tolerant architectural solution for the IoT platform

Keywords: Internet of Things, IoT platform, cloud platform, fault-tolerant architectural solution, Google Cloud Platform

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API (application programming interfaces) – прикладний програмний інтерфейс,

AWS (Amazon web services) – веб сервіси амазону,

BaaS (Backend as a Service) – бекенд як сервіс,

CORBA (Common Object Request Broker Architecture) – загальна архітектура брокера об'єктних запитів,

DoS (denial of service) – відмова в обслуговуванні,

ESB (enterprise service bus) – сервісна шина підприємства,

FaaS (Function as a Service) – функція як сервіс,

HTML (hyper text markup language) – мова розмітки гіпертекстових документів,

HTTP (hyper text transfer protocol) – протокол передачі гіпертексту,

JSON (JavaScript object notation) – запис об'єктів JavaScript,

REST (representational state transfer) – передача репрезентативного стану,

RPC (Remote procedure call) – виклик віддаленої процедури

SAM (serverless application model) – без серверна модель застосування,

SOA (service oriented architecture) – сервісно-орієнтована архітектура,

SOAP (simple object access protocol) – простий протокол доступу до об'єкта,

UI (user interface) – інтерфейс користувача,

UNIX (uniplexed information and computing system) – уніфікована інформаційно-обчислювальна система,

UX (user experience) – досвід користувача,

WSDL (web service description language) – мова визначення інтерфейсу веб-сервісу,

ПЗ – програмне забезпечення,

IoT (Internet of Things) – Інтернет речей,

PAAS (Platform as a Service) – Платформа як Сервіс,

FAAS (Function as a Service) – Функція як Сервіс,

SLA (Service Level Agreement) – згода про рівень обслуговування,

CLI (Command Line Interface) – інтерфейс командного рядку,

gRPC (Remote Procedure Calls) – фреймворк, розроблений компанією Google для виклику віддалених процедур,

DAS (Data acquisition system) – система збору даних,

WAN (Wide Area Network) – глобальна комп'ютерна мережа,

TCP (Transmission Control Protocol) – протокол керування передачею,

UDP (User Datagram Protocol) – протокол датаграм користувача.

ВСТУП

Актуальність

Протягом останніх десятиліть інформаційні та комунікаційні технології намагалися постійно збільшувати кількість інтернет-пристроїв. Окрім традиційних комп’ютерів та мобільних пристроїв, - це пристрої що варіюються від домашньої або побутової техніки (телевізор, холодильник, кавоварка, пилосмок), промислового обладнання та автоматики, охорони здоров’я, транспорту, енергії, будівель. Додавання інших девайсів, які традиційно були поза мережею інтернет стало можливим завдяки технологічному прогресу за допомогою обладнання та нового програмного забезпечення. Розроблюване відмовостійке та екномічне архітектурне рішення для IoT платформи полегшить розробку нових систем та мереж інтернету речей.

Об’єкт дослідження. Архітектура IoT платформи.

Предмет дослідження. Відмовостійке архітектурне рішення для IoT платформи.

Мета роботи

Метою роботи є розробка архітектурного рішення для IoT платформи, що буде економічною та відмовостійкою. Для досягнення поставленої мети, були сформульовані наступні задачі:

- Аналіз сучасної IoT платформи;
- дослідження існуючого архітектурного рішення для IoT платформи;
- аналіз хмарної платформи для побудови IoT інфраструктури;
- проектування та валідація власного архітектурного рішення;
- аналіз отриманих результатів для формулування рекомендацій по використанню запропонованої архітектури.

Наукова новизна одержаних результатів: підвищена відмовостійкість архітектури IoT платформи, що дозволяє використання такої для побудови проектів різного призначення та інтеграції з існуючими.

1. АРХІТЕКТУРА ІНТЕРНЕТУ РЕЧЕЙ

1.1. Загальні відомості

IoT — це мережа фізичних об'єктів або «речей», які взаємодіють один з одним. Під «Інтернетом речей» варто розуміти комплекси і системи, що складаються з сенсорів, мікропроцесорів, виконавчих пристроїв, локальних та / або розподілених обчислювальних ресурсів і програмних засобів, програм штучного інтелекту, технологій хмарних обчислювань, передача даних між якими здійснюється за допомогою мережі Інтернет, та призначені для надання послуг і проведення робіт в інтересах суб'єктів (юридичних або фізичних осіб) [1]. У своїй роботі Dr. Ovidiu Vermesan and Co. (д-р Вермезан та інші, 2011) описали цей термін, розглянувши інтернет яким він є зараз, як Інтернет Енергетики (IoE), Інтернет Медіа (IoM), Інтернет Людей (IoP) та Інтернет Сервісів (IoS) [5]. Cisco (Evans, 2012) вирішила монетизувати цей термін як Інтернет Всього (Internet of Everything), де його розглядали як систему, що складається з речей, де процес, дані та люди разом утворювали «мережу мереж». З точки зору Cisco - IoE з'єднує людей, процес, дані та «речі» разом, щоб сформувати мережу, придатну та корисну для допомоги у відстеженні «речей», а також для вирішення деяких глобальних проблем, таких як посуха, кліматичні зміни, джерела або питна вода, і голод [6].

IoT швидко розширюється, а області застосування включають розумні міста, інтелектуальну воду, інтелектуальне вимірювання, безпеку та надзвичайні ситуації, роздрібну торгівлю, логістику, промисловий контроль, інтелектуальне сільське господарство, розумне тваринництво, домашні та домашні господарства автоматизація та електронна охорона здоров'я. Однак, оскільки області застосування охоплюють різні середовища та пов'язані пристрої різноманітні, це робить IoT дуже неоднорідним, а отже, труднощами та бар'єрами, такими як зв'язність, керування живленням,

складність, швидкий розвиток, безпека та якість обслуговування, які завжди пов'язані з стандартними викликами мережі бездротових сенсорних мереж (WSN), були перелічені Chase (Chase, 2013) як перешкода для розвитку IoT. Інші проблеми це конфіденційність, чутливість до участі, аналізі даних, візуалізація на основі геоінформаційної системи (ГІС) та хмарних обчислень [6]. Крім того, проблеми з підключенням до IoT також пов'язані з архітектурними і протокольними проблемами, які (Gubbi, Buyya, Marusic, & Palaniswami, 2013) вважають у своїй роботі відкритою проблемою.

Сьогодні виробники промислового обладнання стикаються, якщо не з усіма, але з більшістю вищезгаданих проблем. Тому для того, щоб IoT працював успішно та задовольняв прогнозований обсяг пристроїв, підключених до Інтернету до 2020 року, він повинен будуватися на відкритій, гнучкій технічній, програмній та мережевій платформах, здатних розвиватися і адаптуватися.

Отже, IoT включає в себе одночасно декілька явищ. Це безпосередньо прилади, які під’єднані до мережі та взаємодіють між собою. Також, це ще й спосіб підключення – M2M – машина-до-машини, без прямої участі людини/оператора. І найважливіше – дані, які можна збирати, зберігати, аналізувати та, в подальшому, використовувати для підвищення комфорту чи прийняття бізнес-рішень.

За думкою Роба Ван Краненбурга, інтернет речей являє собою «чотиришаровий пиріг». Перший рівень пов’язаний з ідентифікацією кожного об’єкта. Другий рівень представляє разом із сервісом по обслуговуванню потреб споживача (можна розглянути як мережу власних «речей», наприклад, «розумний дім»). 3-й рівень пов’язаний з урбанізацією міського життя. Тобто, це концепція «розумного міста», де вся інформація, що стосується мешканців даного міста, зосереджується в конкретному житловому кварталі, в Вашому будинку та в сусідніх будинках. Нарешті 4-й рівень – сенсорна планета [7].

Іншими словами, інтернет речей можемо розглядати як мережу мереж, в якій невеликі мало суміжні мережі створюють більш масштабні мережі.

Звичайно, для взаємодії приладів необхідний канал зв’язку. Компанія Cisco провела детальний технічний аналіз, який показав, що IP цілком може бути адаптований до потреб мереж нового типу. В такому випадку IoT отримає наступні переваги: сумісність, масштабування і, найголовніше, єдину мову спілкування. Дані переваги у свій час перетворили складну структуру окремих та загальнодоступних мереж в об’єднану глобальну комунікаційну систему, відому як Інтернет [7].

З тих пір, як промислові підприємства почали з'єднувати практично кожен пристрій та «річ», від сміттєвих баків до термостатів, у випадку збору даних у режимі реального часу, сьогодні бізнесу стало відомо, що реальне призначення IoT — це не просто збір та обробка даних, а це аналіз даних для розуміння того ж самого бізнесу.

Деякі визначення чи назви вживаються англійською мовою. Це пов’язано зі специфікою теми.

Через видатні можливості, які обіцяє IoT, все більше організацій прагнуть включити її продукцію до своїх бізнес-процесів. Однак, коли мова заходить про реальність, ця геніальна ідея видається надто складною для реалізації - враховуючи кількість пристроїв та умов, необхідних для її роботи. Іншими словами, проблема полягає у створенні надійної архітектури Інтернету речей.

По суті, архітектура IoT - це система численних елементів: датчиків, протоколів, приводів, хмарних служб та шарів. Зважаючи на її складність, існує 4 етапи архітектури IoT. Таке число вибирається для постійного включення цих різних типів компонентів у складну і єдину мережу.

В основному, є три шари архітектури IoT [4]:

1. Клієнтська частина (шар IoT пристроїв);
2. Оператори на стороні сервера (шар IoT шлюзу);

3. Шлях для підключення клієнтів та операторів (шар IoT платформи).

Насправді задоволення потреб усіх цих шарів є вирішальним на всіх етапах архітектури IoT. Будучи основою критерію здійсненності, ця послідовність робить результат, розроблений справді спрацьовим. Крім того, основними рисами стійкої архітектури IoT є функціональність, масштабованість, доступність та ремонтопридатність.

IoT архітектура складається з 4-х частин [8]:

1. Датчики та приводи;
2. Інтернет шлюз та система збору даних;
3. Edge IT (Концепція граничних обчислень);
4. Дата центри та хмара.

Рис. 1.1. Осноні 4 етапи IoT архітектури.

1.2.1 “Речі”, датчики та приводи

В якості основи для кожної системи Інтернету речей підключені пристрої відповідають за забезпечення сутності Інтернету Речей, якою є дані. Для визначення фізичних параметрів в зовнішньому світі або всередині самого об'єкта їм потрібні датчики. Вони можуть бути вбудовані в самі пристрої або реалізовані як автономні об'єкти для вимірювання та збору даних телеметрії. Наприклад, можна подумати про сільськогосподарські датчики, завданням яким є вимірювання таких параметрів, як температура і вологість повітря, рівень pH грунту або вплив сонячного світла на культуру.

Ще одним незамінним елементом цього шару є приводи. Перебуваючи в тісному співробітництві з датчиками, вони можуть перетворити дані, що генеруються інтелектуальними об'єктами, в фізичні дії. Уявімо розумну систему поливу з усіма необхідними датчиками. Грунтуючись на вхідних даних, наданих датчиками, система аналізує ситуацію в режимі реального часу і дає команду приводам відкрити обрані водяні клапани, розташовані в місцях, де вологість ґрунту нижче заданого значення. Клапани залишаються відкритими до тих пір, поки датчики не сповістить про відновлення значень за замовчуванням. Очевидно, що все це відбувається без єдиного людського втручання.

Важливо також, щоб підключені об'єкти не тільки мали змогу робити можливим обмін даними в двох напрямках з відповідними шлюзами або системами збору даних, а й могли розпізнавати один одного і обмінюватися інформацією, а також спільно працювати в режимі реального часу для використання переваг всієї мережі. Зокрема, в разі пристроїв з обмеженими ресурсами і роботою від акумулятора, досягнення цієї мети не є простим завданням, оскільки для такого зв'язку потрібна велика кількість обчислювальної потужності, споживається дорогоцінна енергія і смуга пропускання. Таким чином, надійна архітектура може забезпечити ефективне управління пристроями тільки в тому випадку, якщо в ній використовуються спеціалізовані, безпечні і легкі протоколи зв'язку, такі як Полегшений M2M, який став провідним стандартним протоколом для управління малопотужними легкими пристроями, типовими для багатьох сценаріїв використання Інтернету речей.

Рис. 1.2. 1-й етап IoT архітектури (Датчики та приводи)

1.2.2 Шлюзи та збір даних

Незважаючи на те, що цей рівень як і раніше працює в безпосередній близькості від датчиків і виконавчих механізмів на даних пристроях, важливо описати його як окремий етап архітектури Інтернету речей, так як він має вирішальне значення для процесів збору, фільтрації та передачі даних в граничну інфраструктуру і хмарні платформи. З огляду на величезний обсяг введених і виведених даних, який може генерувати розгортання мільйонів пристроїв, в центрі уваги повинні перебувати можливості для агрегування, вибору і передачі даних. Будучи посередниками між підключеними до мережі речами, хмарою і аналітикою, шлюзи і системи збору даних забезпечують необхідну точку підключення, яка пов'язує інші рівні разом.

Перебуваючи на межі світу експлуатаційних і інформаційних технологій, шлюзи полегшують обмін даними між датчиками і іншою частиною системи, перетворюючи дані датчиків в формати, які легко переносяться і можуть використовуватися для інших компонентів системи. Більш того, вони можуть контролювати, фільтрувати і вибирати дані, щоб мінімізувати обсяг інформації, яка повинна бути передана в хмару, що позитивно впливає на витрати на передачу даних по мережі і час відгуку.

Таким чином, шлюзи забезпечують місце для локальної попередньої обробки даних датчиків, які стискаються в корисні пакети, готові для подальшої обробки.

Іншим аспектом, який підтримує шлюзи, є безпека. Оскільки шлюзи відповідають за управління інформаційним потоком в обох напрямках, за допомогою відповідних засобів шифрування і безпеки вони можуть запобігти витоку даних в хмарі Інтернету речей, а також знизити ризик шкідливих атак ззовні на пристрої Інтернету речей.

Рис. 1.3. 2-й етап IoT архітектури (Шлюзи та збір даних)

1.2.3 Граничні обчислення

Хоча периферійні пристрої не є неминучим компонентом будь-якої архітектури Інтернету речей, вони можуть принести значні переваги, особливо в великомасштабних проектах Інтернету речей. В умовах обмеженої доступності та швидкості передачі даних хмарних платформ Інтернету речей прикордонні системи можуть забезпечити більш швидкий час відгуку і велику гнучкість при обробці і аналізі даних Інтернету речей.

Оскільки швидкість аналізу даних є ключовим чинником в деяких промислових додатках Інтернету речей, останнім часом в системах промислового IoT спостерігається різке зростання популярності граничних обчислень.

Оскільки гранична інфраструктура може розташовуватися ближче до джерела даних з фізичної точки зору, IT-службі простіше і швидше працювати з матеріалами Інтернету речей в режимі реального часу і надавати результати у вигляді оперативної інформації. У цьому сценарії туди пересилаються тільки великі фрагменти даних, які дійсно потребують потужності хмари для обробки. Завдяки мінімізації ризику для мережі безпека може бути значно підвищена, а зниження енергоспоживання і навантаження на смугу пропускання сприяє більш ефективному використанню бізнес-ресурсів.

1.2.4 Центр обробки даних

Якщо датчики - нейрони, а шлюз - основа Інтернету речей, то центр обробки даних - це мозок в тілі Інтернету Речей. На відміну від рішень для периферійних пристроїв, центр обробки даних або хмарна система призначені для зберігання, обробки та аналізу великих обсягів даних з метою отримання більш глибокого розуміння за допомогою потужних механізмів аналізу даних і машинного навчання, які б ніколи не могли підтримувати периферійні системи.

За останні кілька років стали більш активно використовувати хмарні обчислення (особливо в промисловій архітектурі Інтернету речей), що сприяє підвищенню продуктивності, скорочення незапланованих простоїв і енергоспоживання, а також багатьом іншим перевагам для бізнесу.

При наявності відповідних рішень, призначених для користувача додатків, хмара може надати бізнес-аналітику і варіанти представлення, які допоможуть людям взаємодіяти з системою, контролювати і відслідковувати

її, а також приймати обґрунтовані рішення на основі звітів, інформаційних панелей і даних, що переглядаються в реальному часі.

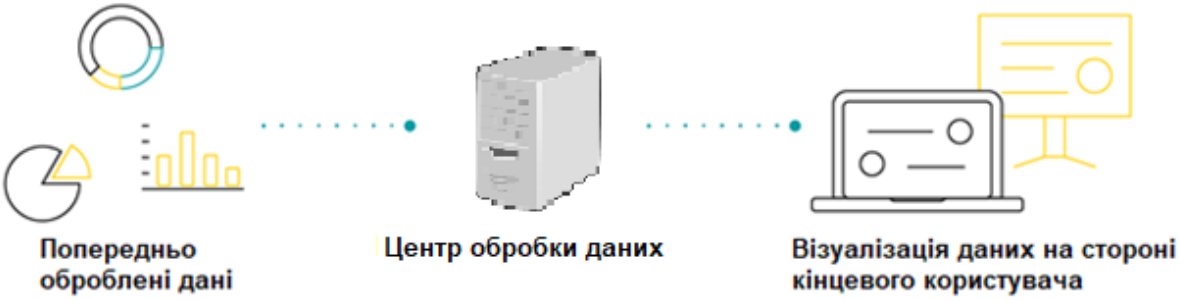

Рис. 1.4. 4-й етап IoT архітектури (Центр обробки даних)

Як вже зазначалося раніше, архітектура Інтернету речей може відрізнятися в залежності від рішення, але її ядро складається з чотирьох блоків, які є ключовими для надання фундаментальних функцій, які роблять екосистему Інтернету речей стійкою: Функціональність, масштабованість, доступність, зручність обслуговування і економічність. Варто відзначити, що все більша увага приділяється розвитку надійної архітектури Інтернету речей, яка спостерігається серед багатьох основних бізнес-гравців з різних галузей промисловості, і це привело до успіху в тому, що вони домоглися більшої цінності даних для бізнесу, що дозволило їм отримати перевагу і перевершити своїх конкурентів.

Незважаючи на те, що ще належить виконати багато роботи з подолання фрагментації технологій Інтернету речей, на сьогоднішній день абсолютно очевидно, що на сьогоднішній день робляться значні зусилля з інтеграції широкого спектру технологій і стандартів, які використовуються Інтернетом речей (наприклад, LwM2M, oneM2M), і є надія на більш уніфіковане і стандартизоване майбутнє. Однак до того, як це стане реальністю, ключем до реалізації потенціалу Інтернету речей не обов'язково є отримання єдиної технології Інтернету речей, а впровадження всіх технологій, щоб вони були ефективними в зборі, управлінні, аналізі та використанні даних за рахунок створення міцної, перспективної, масштабованої і безпечної архітектури Інтернету речей.

Традиційну архітектуру можна зобразити наступним чином:

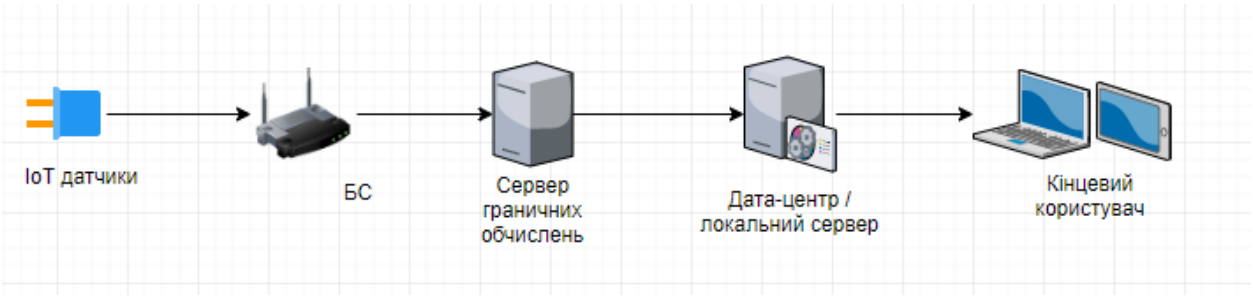

Рис. 1.5. Традиційна IoT архітектура

1.3. Стек технологій Інтернету речей

Враховуючи різноманітність і незліченну кількість технологічних рішень, які оточують IoT для простоти розіб’ємо стек технологій Інтернету речей на чотири основних технологічних рівня:

1. Апаратне забезпечення

Пристрої - це об'єкти, які фактично становлять "речі" в Інтернеті Речей. Діючи в якості інтерфейсу між реальним і цифровим світом, вони можуть мати різні розміри, форми і рівні технологічної складності в залежності від завдання, яке вони повинні виконати в рамках конкретного розгортання Інтернету речей. Як мікрофони розміром зі шпильку, так і важкі будівельні машини, практично кожен матеріальний об'єкт (навіть анімовані, наприклад, тварини або люди) може бути перетворений в підключений пристрій шляхом додавання необхідного обладнання (шляхом додавання датчиків або приводів разом з відповідним програмним забезпеченням) для вимірювання та збору необхідних даних. Очевидно, що датчики, виконавчі механізми або інші телеметричні пристрої також можуть самі по собі представляти собою автономні інтелектуальні пристрої. Єдине обмеження, яке слід прийняти в даному випадку, це фактичний сценарій використання Інтернету речей та вимоги до його обладнання (розмір, простота розгортання і управління, надійність, термін служби, економічність).

2. Програмне забезпечення пристроїв

Саме це робить підключені пристрої "інтелектуальними". Програмне забезпечення відповідає за впровадження зв'язку з хмарою, збір даних, інтеграцію пристроїв, а також проведення аналізу даних в реальному часі в мережі Інтернету речей. Крім того, це програмне забезпечення для пристроїв, яке також забезпечує можливості на рівні додатків для візуалізації даних і взаємодії з системою Інтернету речей.

3. Комунікація

Наявність апаратного і програмного забезпечення пристроїв має бути ще одним рівнем, який забезпечить інтелектуальні об'єкти способами і засобами обміну інформацією з іншим світом Інтернету речей. Незважаючи на те, що комунікаційні механізми сильно прив'язані до апаратного та програмного забезпечення пристроїв, їх вкрай важливо розглядати як окремий рівень. Рівень зв'язку включає в себе як фізичні рішення зв'язку (стільниковий зв'язок, супутниковий зв'язок, LAN), так і спеціальні протоколи, які використовуються в різних середовищах Інтернету речей (ZigBee, Thread, Z-Wave, MQTT, LwM2M) [10]. Вибір відповідного комунікаційного рішення є одним з найважливіших компонентів при створенні кожного стека технологій Інтернету речей. Обрана технологія визначає не тільки способи відправки і отримання даних із хмари, але і способи управління пристроями і їх взаємодії з пристроями сторонніх виробників.

4. Платформа

Як уже згадувалося раніше, завдяки "інтелектуальному" обладнанню і встановленому програмному забезпеченню пристрій може "визначити" те, що відбувається навколо нього, і повідомити про це користувачеві через певний канал зв'язку. Платформа для Інтернету речей - це місце, де всі ці дані збираються, управляються, обробляються, аналізуються і подаються зручним для користувача способом. Таким чином, цінність такого рішення полягає не тільки в його можливостях збору даних і управління ними, а й в його здатності аналізувати і знаходити корисні висновки з частин даних, що

надаються пристроями через рівень зв'язку. Крім того, на ринку існує цілий ряд платформ Інтернету речей, вибір яких залежить від вимог конкретного проекту Інтернету речей і таких факторів, як архітектура і стек технологій Інтернету речей, надійність, властивості установки, що використовуються, протоколи, незалежність від обладнання, безпеку і економічність. Також слід зазначити, що платформи можуть бути встановлені як локально, так і в хмарі. Платформа Coiote для управління пристроями Інтернету речей - хороший приклад такої платформи, яку можна розгорнути як на місці, так і в хмарі. Те ж саме відноситься і до іншої платформі IoT від AVSystem - Coiote IoT Data Orchestration [11].

1.4. Рішення для підключення в стеці технологій Інтернету речей

Існує дуже багато реальних додатків для технологій Інтернету речей і тут немає проблеми в її підключенні. Залежно від технічних характеристик конкретного сценарію використання Інтернету речей, кожен варіант зв'язку може запропонувати різні сценарії надання послуг з одночасним вибором співвідношення між енергоспоживанням, діапазоном і пропускною здатністю. Наприклад, якщо ви будуєте розумний будинок, вам може знадобитися вбудований в смартфон датчик температури всередині приміщення і контролер обігріву, щоб ви могли дистанційно контролювати температуру в кожній кімнаті і регулювати її в реальному часі у відповідності з поточними потребами. В цьому випадку рекомендується використовувати мережевий протокол IPv6 на основі IP-протоколу, званий Thread, спеціально розроблений для домашнього середовища автоматизації.

З огляду на це різноманіття і різноманітність стандартів і протоколів зв'язку, можна поставити запитання про реальну необхідність розробки нових рішень, в той час як існують деякі, які добре зарекомендували себе Інтернет-протоколи, які вже використовуються на протязі десятиліть. Причина цього

полягає в тому, що існуючі інтернет-протоколи, такі як протокол управління передачею / протокол Інтернету (TCP / IP), часто недостатньо ефективні і споживають занадто багато енергії, щоб ефективно працювати в нових програмах для Інтернету речей [12]. Розглянемо альтернативні інтернет-протоколи, спеціально призначені для використання системами Інтернету речей.

Розгляд стосується найпопулярніших радіотехнологій Інтернету речей, розбитих на радіочастотний діапазон, досягнутий кожним з рішень: Радіозв'язок для Інтернету речей малого радіусу дії, рішення для середнього діапазону і рішення для глобальних мереж дальнього радіусу дії.

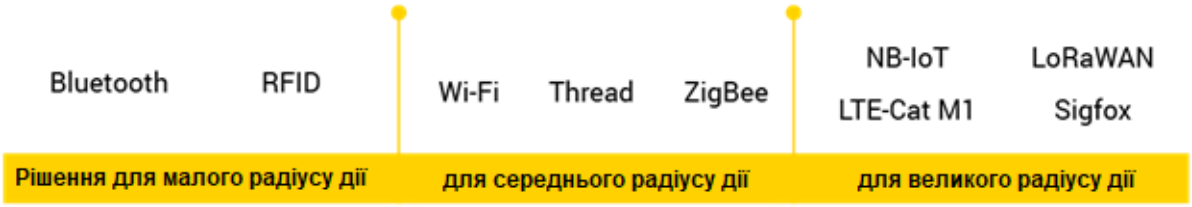

Рис. 1.6. Поділ рішень підключення, що використовуються в Інтернеті речей за радіусом дії

Мережеві рішення для Інтернету речей малого радіусу дії

- Bluetooth

  Будучи широко поширеною технологією підключення на короткі відстані, Bluetooth вважається ключовим рішенням, особливо для майбутнього ринку переносних електронних пристроїв, таких як бездротові навушники або датчики геолокації, особливо з огляду на їх широку інтеграцію зі смартфонами. Протокол Bluetooth Low-Energy (BLE), розроблений з урахуванням економічної ефективності та зниження енергоспоживання, вимагає дуже малого енергоспоживання від пристрою. Проте, це приходить з

компромісом: При передачі часто великих обсягів даних BLE може бути не найефективнішим рішенням.

- RFID

  Будучи одним з перших в історії застосування Інтернету речей, радіочастотна ідентифікація (RFID) пропонує рішення для позиціонування для додатків Інтернету речей, особливо в сфері управління ланцюжком поставок і логістики, які вимагають можливості визначення положення об'єкта всередині будівель. Майбутнє технології RFID виходить далеко за рамки простих послуг з локалізації, а також від відстеження пацієнтів лікарень для підвищення ефективності в охороні здоров'я до надання даних про місцезнаходження товарів в режимі реального часу, що дозволяє звести до мінімуму ситуації, коли магазини не перебувають в наявності.

Рішення для середньої дистанції

- Wi-Fi

  Розроблений на основі стандарту IEEE 802.11, він залишається найбільш поширеним і широко відомим протоколом бездротового зв'язку. Широке використання Інтернету речей в світі в основному обмежується більш високим енергоспоживанням, ніж в середньому, що пов'язано з необхідністю збереження високої потужності сигналу і швидкої передачі даних для поліпшення можливостей підключення і надійності. Будучи ключовою технологією в розробці Інтернету речей, WiFi надає широкий спектр рішень для Інтернету речей, але при цьому необхідно керувати ними і використовувати їх в маркетингових цілях, щоб отримати прибуток як постачальникам послуг, так і користувачам. Прекрасним прикладом платформи управління WiFi, яка пропонує додаткові послуги, що

розширюють можливості загальнодоступних точок доступу WiFi, є Linkify. Будучи одним з найсучасніших рішень AVSystem, Linkify забезпечує практично безмежні можливості настройки і маркетингу гостьовій мережі WiFi.

- ZigBee

  Цей популярний стандарт бездротової комірчастої мережі знаходить найбільш часто використовувані додатки в системах управління трафіком, побутовій електроніці і машинобудуванні. Zigbee, створений на основі стандарту IEEE 802.15.4, підтримує низькі швидкості обміну даними, низьке енергоспоживання, безпеку і надійність.

- Thread

  Система Thread, розроблена спеціально для продуктів для розумного будинку, використовує можливості підключення по протоколу IPv6, щоб підключені пристрої могли обмінюватися даними один з одним, отримувати доступ до сервісів в хмарі або взаємодіяти з користувачем за допомогою мобільних додатків Thread [6].

Рішення для глобальних мереж дальньої дії (WAN)

- NB-IoT

  Narrowband IoT, продукт існуючих технологій 3GPP, - це абсолютно новий стандарт радіотехнології, що забезпечує надзвичайно низьке енергоспоживання (10 років роботи від акумулятора) і забезпечує підключення з рівнем сигналу приблизно на 23 дБ нижче, ніж в разі 2G. Більш того, він використовує існуючу мережеву інфраструктуру, яка забезпечує не тільки глобальне покриття в мережах LTE, але і гарантовану якість сигналу. У багатьох випадках цей факт дозволяє

реалізувати NB-IoT замість рішень, які вимагають побудови локальних мереж, таких як LoRa або Sigfox [8].

- LTE-Cat M1

  LTE Cat M1 - це стандарт зв'язку з широкою областю дії (LPWA) з низьким енергоспоживанням, який дозволяє підключати пристрої Інтернету речей і M2M із середньою швидкістю передачі даних. Він підтримує більш тривалий термін служби акумулятора і пропонує розширений діапазон в будівлі в порівнянні з технологіями стільникового зв'язку, такими як 2G, 3G або LTE Cat 1.

  Будучи сумісним з існуючою мережею LTE, CATM1 не вимагає від операторів створення нової інфраструктури для її впровадження. У порівнянні з NB-IoT, LTE Cat M1 ідеально підходить для використання в мобільних пристроях, оскільки передача даних між стільниковим обладнанням значно краще і дуже схожа на високошвидкісну LTE [9].

- LoRaWAN

  LoRaWAN - це протокол глобальної мережі з низьким енергоспоживанням, оптимізований для низького енергоспоживання і підтримує великі мережі з мільйонами пристроїв. Система LoRaWAN призначена для роботи з додатками глобальної мережі (WAN) та надає енергоекономічні WAN-мережі з функціями, необхідними для підтримки недорогих, мобільних і безпечних двонапрямлених комунікацій в рамках Інтернету речей, M2M, інтелектуальних міст і промислових додатків [3].

- Sigfox

  Концепція Sigfox полягає в тому, щоб забезпечити ефективне рішення для зв'язку малоенергоємних M2M-додатків, що

вимагають низького рівня передачі даних, для яких зона покриття WiFi занадто мала, а зона покриття стільникового зв'язку занадто дорога і дуже енергозатратна. У Sigfox використовується технологія UNB, яка дозволяє йому працювати з низькою швидкістю передачі даних від 10 до 1000 біт в секунду. Енергоспоживання в 100 разів менше в порівнянні з рішеннями стільникового зв'язку, вона забезпечує стандартний час роботи в режимі очікування 20 років для акумулятора 2,5 Ач. Завдяки надійній, енергоефективній і масштабованій мережі, здатної підтримувати зв'язок між тисячами пристроїв, що працюють від батарей, на площі в кілька квадратних кілометрів, Sigfox довів свою придатність для різних застосувань M2M, включаючи інтелектуальне вуличне освітлення, інтелектуальні лічильники, монітори пацієнта, пристрої безпеки і датчики довкілля. В даний час Sigfox використовується в зростаючій кількості технологічних рішень для Інтернету речей, таких як Coiote IoT Data Orchestration компанії AVSystem, і це тільки один з них.

1.5. Аналіз протоколів Інтернету речей

Говорячи про Інтернет Речей, ми завжди думаємо про комунікації. Взаємодія між датчиками, пристроями, шлюзами, серверами і одними додатками є найважливішою характеристикою Інтернету Речей. Але всі ці інтелектуальні функції дозволяють спілкуватися і взаємодіяти з Інтернетом речей, які можна розглядати як мови, що використовуються для Інтернету речей.

Розумний пристрій від його звичайного аналога відрізняється тим, що хоча останній залишається вимкненим в разі поломки, перший може спілкуватися з іншими пристроями (а не тільки з тими, які мають той же

тип), якщо він стикається з будь-якими проблемами і, при необхідності, повідомляти користувачеві про збій або автоматично звертатися за допомогою. Але кожен такий випадок взаємодії можливий тільки в тому випадку, якщо є середовище комунікації, загальна «мова», якою будуть користуватися всі пристрої в даній екосистемі Інтернету речей. В Інтернеті речей, середовище надається IoT протоколами: або тими інтернет-протоколами, що вже давно використовуються, або IoT протоколами, спеціально розробленими для зв'язку з підключеними пристроями.

Протоколи Інтернету речей є важливою частиною стека технологій Інтернету речей - без них обладнання буде марним, оскільки протоколи Інтернету речей дозволяють IT-підрозділу обмінюватися даними структурованим і осмисленим способом. З цих переданих фрагментів даних можна витягти корисну інформацію для кінцевого користувача, і завдяки цьому все розгортання стає економічно вигідним.

Це одна з причин, через яку Інтернет Речей потребує стандартизованих IoT протоколів. Вони допомагають уникнути подальшої фрагментації, тим самим мінімізуючи ризик загроз безпеки.

Незважаючи на те, що всі згодні з цим твердженням, до сих пір не було зроблено жодних зусиль з розробки світового стандарту, який об'єднав би всі комунікації Інтернету речей. Проте, в останні кілька років Інтернет Речей став свідком появи протоколів, спрямованих на вирішення цього завдання, вони пропонують універсальність без компромісів щодо безпеки, швидкості і простоти розгортання. Один з таких протоколів Інтернету речей, призначених для задоволення конкретних потреб різних сценаріїв використання управління пристроями для надання спеціалізованих рішень і пропозиції універсального стандарту - OMA Lightweight M2M.

З іншого боку, фрагментація Інтернету речей є результатом самої природи Інтернету Речей: Неоднорідність Інтернету речей, яку представляє різноманітність технологій і стандартів Інтернету речей, відповідає різноманітності Речей в світі, до яких прагне Інтернет речей. Крім того, існує

безліч аспектів зв'язку з Інтернетом речей, кожен з яких має свій тип протоколів, відповідних його цілям. Протоколи Інтернету речей можна розділити на ролі, які вони відіграють в мережі. Серед багатьох інших протоколів, використовуваних в інфраструктурі підключення (наприклад 6LoWPAN) є також протоколи зв'язку (Wi-Fi, Bluetooth), передачі даних (MQTT, CoAP, XMPP), безпеки (DTLS), управління пристроями, а також телеметричної системи (LwM2M).

1.5.1 Протокол Обмеженого Застосування (CoAP)

Незважаючи на те, що існуюча інтернет-інфраструктура доступна і може використовуватися для будь-якого пристрою Інтернету речей, вона часто виявляється занадто важкою і енерговитратною для більшості сценаріїв використання Інтернету речей. Створений робочою групою Ietf Constrained RESTful Environments і запущений в 2013 році, Протокол Coap (Constrained Application Protocol) був розроблений для перетворення моделі HTTP, щоб її можна було використовувати в обмежених пристроях і мережевих середовищах.

Розроблена для задоволення потреб систем Інтернету речей на основі HTTP, CoAP використовує протокол призначених для користувача датаграм (UDP) для встановлення безпечного зв'язку між кінцевими точками. Забезпечуючи трансляцію і багатоадресну передачу, UDP може передавати дані на кілька вузлів, зберігаючи при цьому швидкість зв'язку і низький рівень використання смуги пропускання, що робить його відмінною підсистемою для бездротових мереж, зазвичай використовуваних в середовищах M2M з обмеженими ресурсами. Ще одна річ, яка спільно використовується CoAP з HTTP, це архітектура RESTful, яка підтримує модель взаємодії запитів / відповідей між кінцевими точками додатків. Більш того, CoAP використовує базові методи HTTP GET, POST, PUT і DELETE,

завдяки яким можна уникнути двозначності під час взаємодії між клієнтами [12].

Як і MQTT, CoAP має функцію якості обслуговування, яка використовується для управління відправленим повідомленням і їх позначення як "підтверджені" або "непідтверджені" відповідно, що вказує, чи повинен одержувач повернути "підтвердження" чи ні. Іншими цікавими особливостями CoAP є підтримка механізму узгодження змісту і виявлення ресурсів. Крім передачі даних Інтернету речей, CoAP використовує систему безпеки транспортного рівня Datagram Transport Layer Security (DTLS) для безпечного обміну повідомленнями на транспортному рівні. CoAP повністю задовольняє потребам протоколу, щоб відповідати вимогам пристроїв, що працюють від батарей або з низьким рівнем енергоспоживання. В цілому, CoAP є хорошою підсистемою для існуючих систем Інтернету речей на основі веб-сервісів.

1.5.2 Message Queuing Telemetry Transport (MQTT)

Ймовірно, найбільш широко використовуваний на сьогоднішній день стандарт промислового Інтернету Речей, Message Queuing Telemetry Transport є полегшеним протоколом обміну повідомленнями типу публікації / підписки (PUB / SUB). Архітектура MQTT, розроблена для пристроїв з живленням від акумулятора, проста і легка і забезпечує низьке енергоспоживання для пристроїв. Працюючи над протоколом TCP / IP, він був спеціально розроблений для ненадійних мереж зв'язку, щоб відповісти на проблему зростаючого числа малорозмірних дешевих об'єктів з низьким енергоспоживанням, які в останні роки стали з’являтися в мережі [14].

MQTT заснована на моделі підписника, видавця і брокера. У моделі, завдання видавця - зібрати дані і відправити інформацію підписникам через посередницький рівень, який є брокером. Роль брокера, з іншого боку, полягає в забезпеченні безпеки шляхом перехресної перевірки дозволу

видавців і підписників. MQTT пропонує три режими досягнення цього (якість обслуговування), завдяки яким видавець має можливість визначити якість свого повідомлення:

- QoS0 (не більше одного разу): Найменш надійний режим, але і найшвидший. Публікація відправлена, але підтвердження не отримано.
- QoS1 (Принаймні один раз): Забезпечує доставку повідомлення як мінімум одного разу, але можуть бути отримані дублікати.
- QoS2 (Рівно один раз): Найнадійніший режим, при цьому найбільш вимагливий до смуги пропускання. Дублікати контролюються для забезпечення доставки повідомлення тільки один раз.

Протокол MQTT знайшов широке застосування в таких пристроях для Інтернету речей, як електричні лічильники, транспортні засоби, детектори, промислове або санітарне обладнання, і він добре відповідає таким вимогам:

- Мінімальне використання смуги пропускання;
- Робота в бездротових мережах;
- Низький рівень споживання енергії;
- Висока надійність при необхідності;
- Не потребує багато ресурсів для обробки і пам'яті.

Незважаючи на свої характеристики, MQTT може бути проблематичним для деяких дуже обмежувальних пристроїв, через факт передачі повідомлень по TCP і управління довгими іменами заголовків. Ця проблема вирішується за допомогою варіанту MQTT-SN, який використовує UDP і підтримує індексацію назв заголовків. Однак, незважаючи на широке впровадження, MQTT не підтримує чітко визначену модель представлення даних і структуру управління пристроями, що робить реалізацію її функцій

управління даними і пристроями повністю залежною від платформи або постачальника.

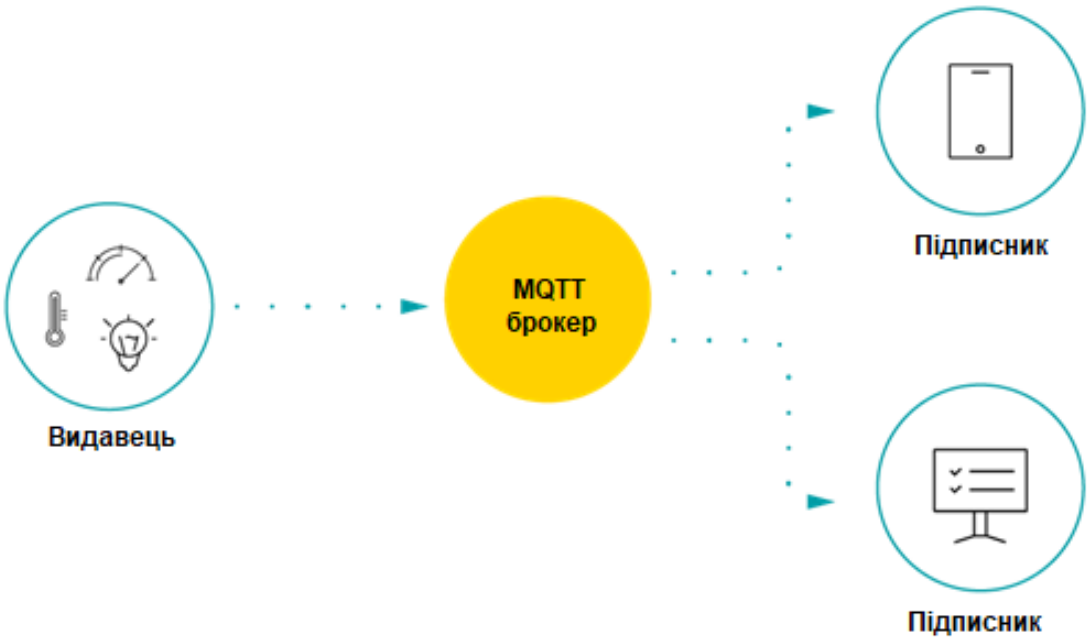

Рис. 1.6. Принцип роботи протоколу MQTT

1.5.3 Extensible Messaging and Presence Protocol (XMPP)

Розроблений в 1999 році спільнотою відкритого коду Jabber і спочатку призначений для обміну повідомленнями в реальному часі, цей комунікаційний протокол Інтернету речей для міжплатформного ПО, орієнтованого на повідомлення, заснований на мові XML. Це дозволяє здійснювати обмін структурованими, але розширюваними даними в реальному часі між двома або кількома мережевими клієнтами.

З моменту свого створення XMPP широко застосовується в якості протоколу зв'язку. Згодом і з появою полегшеної специфікації XMPP: XMPP-IoT, вона стала використовуватися в контексті Інтернету Речей. Будучи стандартом, підтримуваним відкритим співтовариством, переваги XMPP IoT забезпечують адресацію і масштабованість, що робить його ідеальним для розгортання Інтернету речей, орієнтованих на споживача.

Серед недоліків використання XMPP в комунікаціях Інтернету речей слід відзначити, що він не пропонує ні якості обслуговування, ні наскрізного шифрування. Через ці обмеження, серед іншого, передбачається, що його додаток в Інтернеті речей залишатиметься слабо пов'язаним з галуззю, оскільки протокол безумовно не стане стандартним повсякденним використанням для обміну даними та управління пристроями з обмеженими ресурсами, так само, як MQTT або LwM2M .

1.5.4 Data-Distribution Service (DDS)

Як і XMPP, протокол DDS був розроблений на основі методології публікації-підписки. Протокол DDS, розроблений групою Object Management Group (OMG), забезпечує масштабований, надійний, високопродуктивний і сумісний обмін даними між підключеними пристроями незалежно від устаткування і програмної платформи. На відміну від протоколів MQTT і CoAP IoT, DDS підтримує безброкерну архітектуру і багатоадресну передачу для забезпечення високої якості обслуговування і сумісності.

Архітектура протоколу DDS заснована на шарі Data Centric Publish-Subscribe (DCP) і додатковому шарі Data-Local Reconstruction Layer (DLRL). Рівень DCP відповідає за розподіл даних з урахуванням ресурсів, масштабованості та ефективності, а DLRL пропонує інтерфейс для функцій DCP, що дозволяє передавати дані між об'єктами, підключеними до Інтернету речей.

Хоча DDS не є типовим рішенням для Інтернету речей, він як і раніше знаходить своє застосування в деяких промислових розгортання Інтернету Речей, таких як управління повітряним рухом, інтелектуальне управління електромережею, автономні транспортні засоби, транспортні системи, робототехніка, виробництво електроенергії і медичні послуги. В цілому, DDS може використовуватися для управління обміном даними між легкими

пристроями і для об'єднання великих високопродуктивних мереж датчиків. Він також може відправляти і отримувати дані з хмари.

1.5.5 Advanced Message Queuing Protocol (AMQP)

AMQP - це протокол відкритого стандарту типу публікації / підписки, який був створений в 2003 році і має своє коріння в секторі фінансових послуг. Незважаючи на те, що IT-галузь отримала певний досвід в області інформаційних технологій, його використання як і раніше досить обмежене в галузі Інтернету речей. Специфікація AMQP описує такі функції, як орієнтація повідомлень, організація черг, маршрутизація (включаючи "точка-точка" і "публікація і підписка"), надійність і безпеку. Ймовірно, найбільшою перевагою AMQP є його надійна модель зв'язку. На відміну від MQTT, AMQP може гарантувати виконання транзакцій, що, хоча і корисно, не завжди є тим, що потрібно додаткам Інтернету речей.

Через свою тяжкості AMQP не підходить для сенсорних пристроїв з обмеженою пам'яттю, потужністю або пропускною спроможністю мережі, але для окремих сценаріїв використання Інтернету речей він може бути єдиним протоколом, придатним для наскрізного застосування, включаючи такі приклади, як промислове важке обладнання або системи SCADA , де пристрої та мережа значно більш потужніші, як правило.

1.5.6 Lightweight M2M (LwM2M)

Відмінною рисою LwM2M від інших протоколів, що застосовуються в IoT, є те, що він спеціально розроблений для задоволення вимог до комплексної обробки пристроїв з обмеженими ресурсами. Цей стандарт, запущений в 2014 році Open Mobile Alliance (тепер OMA SpectWorks), забезпечує чітко визначений стандарт для передачі даних і управління пристроями Інтернету речей.

У полегшеній специфікації M2M також описані багато типових функцій управління пристроями, такі як операції з віддаленими пристроями, оновлення мікропрограм і програмного забезпечення (FOTA і SOTA), моніторинг і управління підключенням, включаючи управління стільниковим зв'язком і налаштування.

На відміну від будь-якого іншого протоколу Інтернету Речей на ринку, архітектура LwM2M підтримує чотири логічних інтерфейсу, які допомагають стандартизувати спосіб фактичного управління пристроями і телеметрії:

- Інтерфейс початкового завантаження - цей інтерфейс забезпечує управління пристроями без використання головного пристрою. Це в основному означає, що пристрій можна налаштувати для надання правильного обслуговування без попередньої настройки на заводі, що значно знижує витрати і оптимізує час виведення продукту на ринок або послуги.
- Інтерфейс реєстрації клієнта - інформує сервер про "існування" клієнта і підтримуваних функціях. Крім того, він дозволяє оновлювати вбудоване програмне забезпечення та програмне забезпечення по повітрю.
- Інтерфейс управління пристроями і надання послуг - LwM2M дозволяє постачальнику отримувати доступ до екземплярів об'єктів і ресурсів, що дозволяє змінювати налаштування та параметри пристрою.
- Інтерфейс створення інформаційних звітів. Завдяки взаємодії з системою публікації і підписки користувач може отримувати звіти про помилки від пристроїв, коли служба більше не працює належним чином, а також відправляти запити про стан пристрою.

LwM2M використовує легкий і компактний протокол, який дозволяє йому працювати дуже добре в потенційно нестабільних і низькошвидкісних мережах, таких як стільникові мережі або мережі датчиків. Однак його легка

конструкція не змінює того факту, що протокол і раніше дуже ефективно працює на великих ресурсах Інтернету речей, таких як маршрутизатори або шлюзи для бізнесу. Завдяки величезному спектру можливих областей застосування, Легкий M2M знаходить своє застосування в таких областях, як виробництво, логістика, телемедицина, комунальні служби, дистанційне керування, робототехніка, автомобільна промисловість і безпеку, і це лише деякі області застосування.

Переваги протоколу LwM2M над іншими протоколами

Крім простого і ефективного протоколу для керування пристроями з обмеженими ресурсами, LwM2M має низку функцій, які допоможуть йому випередити конкурентів.

На відміну від традиційних M2M-рішень, в яких пристрою зазвичай потрібна підтримка декількох стеків технологій, протоколів і служб безпеки, Полегшена модель M2M дозволяє користувачам мати один стек технологій для управління пристроями не тільки на рівні самого пристрою, але і на рівні додатків. Крім того, LwM2M пропонує міжплатформову сумісність, що робить його ідеальним рішенням для постачальників послуг, які хочуть уникнути прив'язку до постачальника. Об'єднуючи DTLS, CoAP, Block, Teking, SenML LwM2M і Resource Directory, протокол використовує їх для формування інтерфейсу пристрою-сервера з певною структурою об'єктів. Завдяки всім перерахованим вище перевагам Легкий M2M здатний забезпечити ідеальний час виходу на ринок, так як він доступний для миттєвого розгортання.

Ще однією сильною стороною LwM2M, що відрізняє його від інших M2M-протоколів, доступних на ринку, є спосіб вирішення проблем безпеки, особливо на пристроях з обмеженими ресурсами. Він заснований на розширеному протоколі DTLS, який підтримує облікові дані на основі попередньо загальних ключів, необроблених відкритих ключів або сертифікатів і реалізує аутентифікацію, конфіденційність і цілісність даних між сервером і клієнтом.

Нарешті, Полегшений протокол M2M пропонує чітко визначену структуру моделі управління пристроєм і даними, яка дозволяє використовувати ряд функцій, які залежать від постачальника, таких як безпечне завантаження пристроїв, доступ до об'єктів або ресурсів і звітність про пристрої.

В цілому, Полегшений протокол M2M забезпечує гнучке, що масштабується і не залежне від виробників управління пристроями з більш високим часом виведення на ринок, що робить його особливо підходящим для пристроїв з низьким енергоспоживанням з обмеженими можливостями обробки і зберігання даних. З огляду на все це, LwM2M є кращим рішенням для великих, складних і тривалих розгортань, що включають крос-платформні і крос-стандартні сервіси Інтернету речей.

За останні два десятиліття Інтернет Речей швидко розширювався по всьому світу. Пропрацювавши в різних галузях промисловості, таких як виробництво, охорона здоров'я, автомобілебудування, безпека, транспорт і багато іншого, він значно розширив можливості підприємств і приніс їм економічну цінність.

Сьогодні Інтернет Речей підтримує десятки різних протоколів Інтернету речей. З огляду на це багато експертів по Інтернету речей почали закликати до глобальної стандартизації протоколів. Проте, будучи фрагментованим за своєю суттю, ринок Інтернету речей, ймовірно, ніколи не буде мати потребу в всеохоплюючій моделі. Так само як в галузі Інтернету речей з'являються нові і нові додатки і сценарії використання, як і раніше будуть з'являтися спеціалізовані протоколи Інтернету речей для їх розгортання. Слід також підкреслити, що безпечне та ефективне управління пристроями є наріжним каменем стійкого розвитку мереж Інтернету речей у всьому світі. Це одна з причин, по якій опис і розуміння різних протоколів Інтернету речей дійсно мають значення. Тому, що дійсно необхідно, так це знання власних бізнес-потреб і вимог, розуміння переваг і недоліків

пропонованих ринком протоколів, а також здатність вибрати той, який найкраще підходить для даного прикладу використання.

Висновок до розділу 1

В першому розділі була розглянута традиційна архітектура IoT платформи, проаналізовані протоколи Інтернету речей, розглянуті їх переваги та недоліки, Розглянуті варінти підключення датчиків до мережі Інтернету Речей.

Отже, традиційна архітектура, що застосовується для IoT платформи, має свої недоліки, а саме:

1. Недостатня відмовостійкість, через використання звичайних серверів, які не підходять для мережі Інтернету речей.
2. Складнощі масштабування, сервери не зможуть обробляти всплески інформаційних запитів, через що, частина даних може бути втрачена.

Ці проблеми можна виправити, завдяки використанню хмарної платформи та Serverless (Функція як сервіс). В наступному розділі буде проводитись аналіз хмарних сервісів для побудови інфраструктури IoT, будуть розглянуті переваги від використання хмарних сервісів в IoT архітектурі.

2. АНАЛІЗ ХМАРНИХ СЕРВІСІВ ДЛЯ ПОБУДОВИ ІНФРАСТРУКТУРИ IoT

2.1. Загальні відомості

Існує ряд визначень терміну «хмарні обчислення» наданих в літературі. Фактично, сам термін відноситься до концепції нової технології, яка все ще змінюється і розвивається. Незважаючи на те, що запропоновано кілька визначень, які відображають суттєві характеристики цієї нової обчислювальної парадигми, кожне визначення зосереджується на певних особливостях та характеристиках. Вакеро, Родеро-Меріно та співавтори зазначають, що набір мінімальних функцій, які включено в більшість визначень, включає віртуалізацію, модель оплати за використання утиліти та масштабованість, причому віртуалізація розглядається як основна технологія хмарних обчислень. Тим не менш, існує безліч інших функцій та характеристик, таких як забезпечення, простоти використання, а також передача даних та процесів аутсорсингу, що, на мою думку, має дати більш вичерпне визначення. Хоча 16-а версія визначення хмарних обчислень, опублікована Національним інститутом стандартів і технологій США (NIST) широко прийнята та добре продумана, визначення хмарних обчислень, запропонована в , є більш комплексним, оскільки воно охоплює більше аспектів, таких як простоти використання, і це визначення полягає в наступному [14]:

Хмара - це наступний етап розвитку Інтернету. Це засіб, за допомогою якого вст - від обчислювальної потужності до бізнес-процесів і персональної спільної роботи - надається вам як послуга в будь-який час і в будь-якому місці.

Хмару можна уявити як спосіб доступу до нових видів сервісів з підтримкою технологій. Сьогодні організації можуть вибирати способи

надання та впровадження бізнес-послуг. Вони можуть отримати доступ до них власними силами, розмістити їх на хостингу, повністю передати на аутсорсинг або придбати їх через хмару. Але в кінцевому результаті більшість організацій матимуть гібридну середу, що включає послуги з декількох джерел.

Це означає, що не всі технологічні бізнес-процеси будуть переміщені в хмару. Компанії хочуть уважно вивчити свої найбільш стратегічні бізнес-процеси, інтелектуальну власність і бізнес-інформацію, а також визначити, які обчислювальні ресурси повинні як і раніше надаватися за допомогою традиційних моделей надання технологій і які з них готові використовувати можливості, пропоновані хмарою.

Сьогодні ми вже бачимо, що хмара дозволяє виділяти технологічні ресурси або послуги з підтримкою технологій на основі самообслуговування. Одним з ключових індикаторів хмарної служби є те, що технологія абстрагується від користувача.

Архітектура хмарних сервісів базується на динамічному підході, який є масштабованим, керованим запитами, і в разі інфраструктури може одночасно підтримувати безліч різних типів робочих навантажень. Хмарні сервіси повинні бути спроектовані таким чином, щоб забезпечити розраховану на багато користувачів функціональність - різні компанії спільно використовують одні й ті ж базові ресурси. IT-організації повинні мати можливість керувати даними таким чином, щоб вони були точними і захищеними. Вона повинна бути здатна адаптуватися до ситуації, наприклад, до простою електроживлення, і тому повинна бути стійкою. Хмарні сервіси повинні забезпечувати певний ступінь надійності, безпеки і керованості в умовах мінливого середовища. Найголовніше, вони повинні мати можливість масштабуватися, щоб не обмежувати зростання кількості запитів [7].

Визначення хмарних обчислень повинне визначати фундаментальні характеристики, які роблять обчислювальні послуги цінними та помітними.

NIST зазначив більшість характеристик, які широко використовуються серед спільноти і такі характеристики включають:

- Замовлення самообслуговування. Хмарні обчислювальні ресурси (наприклад, процесор, накопичувач, програмне забезпечення) надаються за потребою та заплановано, не вимагаючи людської взаємодії з провайдером послуг. Ключовими моментами цієї функції є економія часу, економічність, зручність використання та обсяг наданих послуг.
- Широкий доступ по мережі. Хмарні послуги доступні через широко доступну мережу, головним чином Інтернет, який використовує стандартні протоколи та механізми для підтримки різних типів пристроїв і платформ, наприклад, смартфонів, тонких клієнтів та КПК.
- Поєднання ресурсів. Фізичні хмарні ресурси, засновані на технології віртуалізації, розподіляються серед хмарних клієнтів, залежно від потреб споживання. Клієнти не усвідомлюють фізичних обмежень ресурсів, оскільки вони віртуально забезпечуються та автоматично вилучаються відповідно до попиту.
- Розрахункові послуги. Оскільки послуга надається згідно з бізнес-моделлю «оплати за використання», використання послуг та ресурсів можна виміряти та автоматично виставляти рахунок за кожний конкретний сеанс клієнта.
- Незалежність від місцевості. Хмарні ресурси можуть бути фізично розташовані в будь-якій географічній локації, якщо доступні можливості зв’язку. Хмарні клієнти також можуть володіти доступом до послуги з будь-якої точки за тими самими умовами.

Технологія “хмарних” обчислень дозволяє задовольняти в тій чи іншій мірі інформаційні потреби підприємства чи окремого користувача зовнішніми провайдерами. В основі Cloud Computing лежать декілька підходів [2]:

- Доступність через Інтернет. При цьому “назовні” хмара видає себе за звичайний сервер;
- Віртуалізація. Завдяки віртуалізації користувачі отримують необхідну кількість ресурсів. Що для цього потрібно з боку сервера і яким чином він може виділити такі ресурси - все приховано за стінами віртуальних машин;
- Cloud Computing - це послуга. Хмара для користувача - це деякий набір послуг, які споживаються без найменшого представлення, що відбувається всередині хмари;
- Простота і стандартність. При роботі з “хмарою” немає необхідності створювати складні конфігураційні файли. Все, що пропонується всередині хмари, доступно через прості виклики API та протоколи, наприклад, так званий REST, за допомогою якого всі операції над даними можна робити через http-запити.

2.2. Які технології можна назвати хмарними

Національним інститутом стандартів і технологій США було виділено п’ять основних принципів побудови хмарної моделі [15]:

- самообслуговування за вимогою. Споживач самостійно визначає і змінює у односторонньому порядку потребу у обчислювальних ресурсах та їх параметри, наприклад: серверний час, швидкість доступу та обробки даних, обсяг збережених даних без безпосередньої взаємодії з представником постачальника послуг хмарних технологій;

- універсальність доступу. Послуги хмарних обчислень мають бути доступні споживачам незалежно від термінального пристрою, що використовуються [16];
- об’єднання ресурсів. Постачальник послуг поєднує ресурси для обслуговування великої множини споживачів в єдиний пул для динамічного перерозподілу обчислювальних потужностей між ними в умовах швидкоплинних змін попиту на обчислювальні потужності. При цьому споживачі контролюють лише основні параметри наданих послуг (обсяг даних, швидкість доступу), але фактичний розподіл ресурсів, що надаються споживачеві, здійснює постачальник (в окремих випадках споживач може безпосередньо керувати деякими фізичними параметрами перерозподілу, наприклад обирати бажаний центр обробки даних з міркувань географічного розташування);
- еластичність масштабування. Послуги можуть бути надані, розширені і звільнені у автоматичному режимі в будь-який момент, без витрат часу та ресурсів на взаємодію з представником постачальника послуг хмарних технологій;
- облік споживання. Постачальник послуг автоматично обчислює спожиті обчислювальні ресурси на певному рівні абстракції: обсяг збережених даних, пропускна здатність, кількість користувачів, кількість транзакцій тощо і на основі цих даних оцінює обсяг наданих споживачам послуг.

2.2.1 Універсальність доступу

Хмарні обчислення відокремлюють обчислювальні ресурси від споживачів, тому їм не потрібно самостійно їх підтримувати та обслуговувати. Наслідком цього є необхідність доступу за допомогою мережі інтернет до територіально розподілених центрів обробки інформації.

Більшість додатків, що призначені для обміну даними по мережі використовують протокол передачі гіпертекстових документів (HTTP), але деякі додатки додатково використовують інші протоколи поверх протоколу HTTP для формування веб-служб. Консорціум Всесвітньої павутини (W3C) визначає веб-службу як програмну систему, що застосовується для підтримки взаємодії типу машина-машина. Можна вирізнити два основні класи вебслужб: ті, що засновані на SOAP – протоколі обміну структурованими повідомленнями в розподілених обчислювальних системах, та ті, що засновані на REST – особливому підході до архітектури мережевих протоколів, який забезпечує доступ до інформаційних ресурсів.

Простий протокол доступу до об’єктів (SOAP) заснований на форматі XML у якості формату повідомлень та зазвичай використовує протокол HTTP для передачі повідомлень. SOAP утворює основу цілого набору специфікацій, що зазвичай називають стеком WS-*. Мова опису веб-сервісів (WSDL) та інструментарій для опису веб-сервісів (UDDI) дозволяють відкривати вебслужби, що пропонують певну послугу використовуючи реєстр UDDI, а потім зв’язатися з ним за допомогою WSDL, без попереднього ознайомлення з прикладним програмним інтерфейсом (API) замовника. Через складність стеку WS-*, SOAP додає велику кількість накладних витрат, тому спостерігається тенденція переходу до більш простих реалізацій [17].

Використання архітектурного стилю REST дозволяє враховувати масштабованість веб-служби та використовує тільки HTTP-методи, що значно спрощує інтерфейс та дозволяє використовувати одночасно XML та JavaScript Object Notation (JSON) у якості формату повідомлень. Стосовно хмарних технологій, спостерігається поступове зменшення популярності SOAP на користь REST.

2.2.2 Самообслуговування за вимогою

Хмарні обчислення надають ресурси на вимогу, тобто, коли вони потрібні споживачу. Це стає можливим завдяки самообслуговуванню і автоматизації, споживач сам виконує всі дії, необхідні для модернізації наданих йому послуг, замість того, щоб звертатися до постачальника послуг або ІТ-відділу своєї компанії. Запит споживача автоматично обробляється хмарною інфраструктурою, без втручання посередників.

Для реалізації самообслуговування, постачальник повинен мати інфраструктуру для автоматичних запитів споживачів. Зазвичай, ця інфраструктура має віртуальний характер, що дозволяє різним споживачам використовувати одне й теж апаратне забезпечення.

Автоматизація самообслуговування вимагає від постачальника хмарних послуг високого рівня планування. Так, споживач може запросити нову віртуальну машину у будь-який час, та очікує, що вона запрацює за кілька хвилин, але постачальнику хмарних послуг може знадобитися значно більший проміжок часу для отримання фізичного обладнання для центру обробки даних. Тому необхідно постійно стежити за тенденціями використання обчислювальних ресурсів та своєчасно планувати оновлення та модернізацію обладнання.

Постачальник хмарних технологій не може вимагати спеціалізованих знань від споживача, бо у традиційній ІТ-структурі компанії, відповідні спеціалісти заздалегідь прогнозують очікуваний рівень навантаження на обчислювальні ресурси підприємства. Але для споживача хмарних послуг таке прогнозування є неприйнятним. Відповідно, кінцевий користувач хмарного сервісу не має піклуватися про розподіл навантаження та інші технічні параметри [18]. Замість цього, постачальник хмарних обчислень має надати зрозумілий інтерфейс користувача за замовчанням, а при необхідності забезпечити доступ для ІТ-фахівця споживача для перегляду та зміни відповідних технічних параметрів наданої хмари.

Високий рівень автоматизації, необхідний для роботи з хмарою, означає, що для користувача немає ніякої можливості для перевірки кожної окремої ситуації та прийняття зваженого рішення для запиту на основі його контексту. Замість цього рішення мають бути своєчасно формалізовано у вигляді обмеженого набору політик, що автоматично виконуються хмарною інфраструктурою.

2.2.3 Спільне використання обчислювальних потужностей

Об’єднання ресурсів, спільне використання обчислювальних потужностей призводить до збільшення коефіцієнта використання ресурсів, відповідно використовуючи віртуальну інфраструктуру та динамічний перерозподіл обчислювальних потужностей дозволяє заощадити певну частину коштів за рахунок збільшення кінцевих користувачів.

Об’єднання ресурсів на програмному рівні, накладає відповідні обмеження на розробників програмного забезпечення, що мають передбачити можливість одночасного використання програмного продукту одночасно кількома користувачами.

Коли кілька споживачів використовують одні й ті самі ресурси, виникає питання, як саме має розраховуватися вартість отриманий послуг. Білінгова та дозуюча інфраструктури автоматично збирають інформацію про кожного споживача. Для цього кожному запиту має бути встановлено відповідний ідентифікатор транзакції, що також пов’язаний з відповідним споживачем хмарних технологій. Ідентифікатор транзакції передається всім використаним компонентам для розрахунку загальною вартості наданих послуг. Розмір плати може залежати від технічних характеристик наданого обладнання: центрального процесору, обсягу пам’яті, пропускної здатності мережі, кількість одночасно оброблюваних запитів, кількість одночасно працюючих користувачів.

2.2.4 Масштабування за потребою

Оскільки споживачі хмарних послуг очікують отримувати обчислювальні ресурси у будь-якій кількості і у будь-якій час, хмара повинна повсякчас мати змогу для швидкого масштабування вгору й вниз, як того вимагає поточне навантаження. Слід мати на увазі, що масштабування вниз так само важливо, як і масштабування вгору, для відповідного розподілу навантаження та зберігання вільних ресурсів.

Різне програмне забезпечення, що працює у хмарі, мають різні моделі робочого навантаження. Через ці відмінності, високі робочі навантаження у деяких додатках, буде збігатися з низьким навантаженням у інших. Саме тому об’єднання ресурсів призводить до підвищення коефіцієнта використання ресурсів та економії.

Для досягнення цієї економії хмарна інфраструктура повинна мати можливість для швидкого масштабування. У загальному випадку, масштабування – це здатність системи збільшувати продуктивність пропорційно до збільшення апаратного забезпечення. У масштабованій хмарі, можна просто додати нове обладнання при збільшенні попиту на нього, забезпечуючи забезпечувати продуктивність додатків на необхідному рівні.

Спираючись на те, що ресурси у системі, як правило, мають певний рівень службового навантаження, важливо зрозуміти, який саме відсоток від загального обчислювального ресурсу доступно для користувача. Вимірювання додаткового збільшення продуктивності шляхом додавання одиниці апаратного забезпечення, в порівнянні з раніше доданим забезпеченням ресурсу називається коефіцієнтом масштабування.

Але існує це один спосіб погляду на масштабованість: вертикальна та горизонтальна масштабованість. Вертикальна масштабованість передбачає збільшення або зменшення продуктивності одного вузла в системі, шляхом зміни його апаратного забезпечення. Але існує межа того, як сильно можна

змінити продуктивність вузла, через обмеження операційної системи на об’єм оперативної пам’яті, сталу кількість фізичних портів, відповідність роз’єму процесору й материнської плати тощо. Горизонтальна масштабованість спрямована на зміну кількості вузлів у системі, що дозволяє обійти обмеження вертикальної масштабованості. Горизонтальна масштабованість дозволяє використовувати обладнання загального призначення, не використовуючи дороге спеціалізоване апаратне забезпечення.

2.2.5 Плачу за те що споживаю

Для оперативного масштабування вгору чи вниз, потрібно постійно аналізувати поточний попит на обчислювальні ресурси хмари, тобто навантаження на центральний процесор, оперативну пам'ять та пропускну здатність мережі, щоб переконатися, що споживачі не відчують вичерпності цих ресурсів.

Щоб зменшити власні ризики, споживачі підписують з постачальником хмарних технологій угоду про рівень обслуговування (SLA), що має гарантувати відповідність послуг, що надаються і мають надаватися поставником хмарних обчислень. Одна з найбільш важливих послуг у SLA є доступність. Хоча апаратне забезпечення, як правило, дуже надійне, воно теж може бути виведене з ладу, тому угода про рівень обслуговування має передбачати дії поставника хмари на цей випадок. Наприклад, це може бути реалізована реплікація даних до територіально відокремлених центрів обробки даних, що може гарантувати, що принаймні у одному з цих центрів інформація буде доступна для споживача.

2.3. Переваги хмарної платформи для IoT

Масштабованість

Одне з головних переваг розміщення системи Інтернету речей в хмарі - простота масштабування. У разі складних локальних мережевих інфраструктур масштабування вимагає придбання більшої кількості обладнання, збільшення часу і додаткових зусиль по настройці, щоб забезпечити його правильну роботу. З іншого боку, в хмарній системі Інтернету Речей додавання нових ресурсів зазвичай зводиться до оренди іншого віртуального сервера або більш хмарного простору, яке зазвичай має додаткову перевагу швидкого впровадження. Крім того, хмарні сервіси для Інтернету речей забезпечують більшу гнучкість, якщо ви хочете обмежити вимоги до систем зберігання даних або зменшити кількість пристроїв з підтримкою Інтернету речей.

Мобільність даних

Завдяки тому, що ваші дані зберігаються і обробляються на хмарному сервері, до них можна отримати доступ практично з будь-якої точки світу, що також означає, що вони не будуть пов'язані будь-якими інфраструктурними або мережевими обмеженнями. Мобільність особливо важлива при реалізації проектів Інтернету речей, що включають моніторинг і управління підключеними пристроями в реальному часі. Незважаючи на те, що дані, що зберігаються на локальних серверах, можуть оброблятися тільки всередині компанії, вдосконалена хмарна платформа Інтернету Речей надасть вам інструменти для виділення ресурсів, управління і поновлення пристроїв і датчиків, а також обробки отриманих даних віддалено і в реальному часі.

Час виходу на ринок

З хмарними рішеннями для Інтернету речей зазвичай потрібно менше часу і зусиль для їх впровадження і значно знижуються загальні витрати, але це досягається за рахунок індивідуальної настройки платформи. Незважаючи на те, що системи Інтернету Речей, встановлені на території підприємства, можуть бути легко встановлені для цілей проекту, вони також включають в себе трудомістке розгортання функцій управління даними і аналізу, а також модернізацію існуючої мережевої структури компанії в зв'язку зі

збільшенням трафіку даних. В цілому, хмарна інфраструктура Інтернету речей виявляється більш прибутковою, коли час виходу на ринок є ключовим фактором бізнесу.

Безпека

Проблеми безпеки, які з самого початку були головною проблемою для Інтернету речей, можуть виявитися тут складними. На відміну від локальної інфраструктури Інтернету речей, в хмарній платформі вся справа у відповідальності. У разі використання локальних серверів вони знаходяться в руках компанії і залежать тільки від методів забезпечення безпеки в організації, якщо дані зберігаються в безпеці. Тому цілком зрозуміло, що деякі організації можуть відчувати дискомфорт у зв'язку з тим, що вони не можуть контролювати свої конфіденційні дані і передавати їх зовнішнім сторонам. Проте, як постачальники послуг, так і клієнти згодні з тим, що зберігання і обробка даних Інтернету Речей в хмарі більш безпечна, ніж зберігання даних в локальних середовищах. Завдяки можливості регулярного оновлення програмного забезпечення і мікропрограм, а також цілодобового моніторингу, що забезпечується деякими постачальниками платформ, можна уникнути серйозних порушень безпеки.

Рентабельність

Великі початкові інвестиції і підвищений ризик впровадження в разі внутрішньої системи Інтернету Речей можуть обескуражувати. Крім того, виникає проблема поточних витрат на обслуговування обладнання і IT-персоналу. З точки зору хмари, все виглядає краще. Значне скорочення витрат на обслуговування і гнучка схема ціноутворення на основі фактичного використання дозволяють підприємствам, заснованим на Інтернеті речей, перейти на хмарні технології. В рамках цієї бізнес-моделі простіше прогнозувати витрати, і вам не доведеться турбуватися про збій обладнання, який в разі внутрішніх систем Інтернету Речей може привести до величезних додаткових витрат, не кажучи вже про бізнес-втрати, викликаних зникненням обслуговування.

Використання технологій Інтернету речей шляхом переміщення їх з централізованих локальних серверних систем в розподілену хмарну інфраструктуру може бути корисним у багатьох випадках. Це особливо вірно в разі керування пристроями, коли масштабоване, адаптоване для конкретних цілей і економічне хмарне рішення для Інтернету Речей може мати всі переваги. Хмарна платформа для Інтернету речей AVSystem відповідає зростаючому ринковому попиту на послуги Інтернету речей, розміщених в хмарі. Завдяки кращій в своєму класі системі безпеки (на основі протоколів TLS і DTLS), функціям автоматичного масштабування і автоматичного визначення розмірів, а також гнучкій моделі оплати в міру необхідності, вона забезпечує кращий сервіс хмарної платформи для Інтернету речей не тільки для розгортання з пристроями, обмеженими ресурсами, але і для багатьох інших галузевих вертикалей.

Хмарні обчислення, а також Інтернет речей, допомагають підвищити ефективність повсякденних завдань і мають взаємодоповнюючі відносини. З одного боку, IoT генерує багато даних, а з іншого боку, хмарні обчислення прокладають шлях для переміщення цих даних. Існує безліч постачальників хмарних послуг, які використовують це для надання моделі оплати в міру використання, в якій клієнти платять за певні використовувані ресурси. Крім того, хмарний хостинг як послуга підвищує цінність для стартапів Інтернету речей, забезпечуючи економію за рахунок масштабу для зниження загальної структури витрат.

Крім того, хмарні обчислення також забезпечують більш ефективну спільну роботу для розробників. Полегшуючи розробникам зберігання і віддалений доступ до даних, хмара дозволяє розробникам впроваджувати проекти без затримок. Крім того, завдяки зберіганню даних у хмарі компанії Інтернету речей можуть отримати доступ до величезного обсягу великих даних.

Використання інтерфейсу веб-служб

Хмарні сервіси мають стандартизовані інтерфейси веб-служб, які дозволяють замовникам більш легко пов'язувати хмарні функції з внутрішніми програмами. Без стандартизованих інтерфейсів кожен постачальник хмарних послуг повинен придумати інтерфейси, що тільки ускладнює роботу. Хороша аналогія - думати про залізничної мережі. Що буде з залізничним транспортом, якби кожен постачальник залізничних послуг проекту залізничний транспорт з різними розмірами рейок? Це могло б спрацювати, коли кожна залізнична компанія вважала себе бореться за контроль. Однак з часом цей підхід дозволив би ще більше знизити вартість і складність транспортної мережі.

Тепер зробимо таблицю порівняння традиційної архітектури Інтрнету речей та архітектури з використанням хмарної платформи:

Параметр	Традиційна архітектура Інтернету Речей	Архітектура Інтернету Речей з використанням хмарних платформ
Доступність	Обмежена доступність за рахунок використання локальних серверів	Дуже широка доступність, з будь якої точки земного шару
Відмовостійкість	Невисока відмовостійкість	Дуже висока відмовостійкість, як заявляє постачальник хмарних послуг, система працює 99.999% часу
Зберігання даних	Як правило, має лімітоване місце для зберігання даних, розширення відбувається за допомогою покупки та встановлення нових жорстких дисків	Має безлімітне місце для зберігання даних
Масштабованість	Масштабування здійснюється за допомогою покупки та встановлення на сервери нового обладнання, проблеми з обробкою всплесків запитів	Має автоматичне масштабування

Таблиця 2.1. Переваги від використання хмарних платформ для IoT

Висновок до розділу 2

В другому розділі був проведений аналіз хмарної платформи, за результатами аналізу визначені переваги використання хмарної платформи для побудови інфраструктури IoT. Як ми бачимо, переваги дійсно дуже вагомі, найбільшою перевагою від використання хмарних платформ є відмовостійкість. Також дуже важливими перевагами для архітектури є:

- Масштабованість;
- Доступність системи;
- Місце для зберігання даних;
- Економічність.

Отже, для того щоб підвищити відмовостійкість традиційної IoT архітектури необхідно впровадити використання хмарної платформи та хмарних функцій (Serverless).

3. РОЗРОБКА ВІДМОВОСТІЙКОГО АРХІТЕКТУРНОГО РІШЕННЯ

3.1. Постановка задачі

Необхідно удосконалити існуючу традиційну архітектуру для IoT платформи, шляхом запровадження використання хмарної платформи та Serverless (Хмарні функції).

Розглянемо випадок, коли є дуже багато IoT датчиків, це необхідно для того, щоб створити велике навантаження на систему. Наприклад нехай це буде метеорологічна станція, з цих датчикыв ми отримуємо наступну інформацію:

- Погодні умови;
- Рух небесних тіл;
- Геологічні дані;
- Фізичні явища.

Вся ця інформація аналізується за допомогою локальних ресурсів і в хмарі автоматично з використанням попередньо заданих алгоритмів.

В результаті цього аналізу, ми отримуємо наступну інформацію:

- Прогнозування погоди;
- Прогнозування катаклізмів;
- Пошук мінералів, води, життя на інших небесних тілах.

При таких початкових умовах, необхідно досягти наступних цілей:

1. «Вчений» повинен працювати як оператор платформи, здатний працювати без проблем з усіма алгоритмами і функціями на всіх об'єктах;
2. Потрібна швидка установка і настройка середовища;
3. Збір наукових даних і обробка даних в цілому;
4. Скорочення часу обробки наукових даних;

5. Розгортання обробки даних поруч з джерелом даних;
6. Отримати конкурентну перевагу, використовуючи моделі PaaS і FaaS на основі хмари;
7. Легка інтеграція зі сторонніми сервісами;
8. Уніфікувати обробку даних та отримувати дані по всіх каналах сторонніх сервісів.

Вимоги до архітектурного рішення:

- Повинен бути інтерфейс командного рядка для управління, пошуку та усунення несправностей, і т.д;
- Ресурси введення даних повинні мати безпечне підключення до функції;
- Інтернет-сховище даних має бути захищеним;
- Система повинна виконувати більш 25000 функцій в день;
- Масштабованість - При високому навантаженні, система повинна успішно масштабироваться вгору і вниз, щоб справлятися з періодичними сплесками трафіку;
- Аварійне відновлення (при будь-яких обставинах, в разі регіональної аварії в одному з центрів обробки даних, в яких розміщено додаток, має бути автоматичне аварійне перемикання на вторинний регіон з мінімальним часом простою служби);
- Зручність Технічного Обслуговування.

Ризики

Розглянемо потенційні проблеми з високим пріоритетом, які, як очікується, нададуть значний вплив на архітектурне рішення, якщо не будуть належним чином усунуті.

Зручність обслуговування: Завдяки підходу до мікросервісу існує безліч окремих функціональних додатків, які мають велику ступінь дублювання налаштувань конфігурації. Одне значення конфігурації

необхідно змінити у всіх місцях, де воно відображається для правильної роботи програми та реагування на зміну конфігурації.

Безпека: публічні та приватні ключі API, розміщені в конфігурації.

Масштабованість: рішення базується на базі баз даних, черг і об'єктних сховищ.

Проектний план

Доцільно буде, якщо проект матиме дві ключові фази життєвого циклу: початкова фаза та допоміжна фаза.

Стан, поки архітектурне рішення залишається в активному режимі прототипування, називається початковою фазою. Метою початкової фази є реалізація всіх прототипів та технічних рішень для того, щоб рішення було готове до переходу на допоміжну фазу.

Впродовж допоміжної фази виконуються заходи по просуванню та вдосконаленню архітектурного рішення.

Важливою метою допоміжної фази є підтримка бажаної якості обслуговування та впровадження більш сучасних технічних рішень.

3.2. Пропоноване архітектурне рішення

Спроектуємо вимоги до системи на доступні технології, отримаємо наступні **обмеження**:

- Централізоване зберігання і управління транзакціями в хмарі;
- Безшовна інтеграція з учасниками з інших організацій (наукові організації) і вченими;
- Безпечний доступ до даних, їх передача, і зберігання даних, захищених від несанкціонованого доступу;
- Робочий процес перетворення даних на основі конфігурації;
- Оптимальний час безвідмовної роботи до 99,9%;
- Регіональні простої не повинні впливати на доступність системи;

- Конвеєр прийому даних на гібридній хмарній платформі, з використанням Serverless,
- Проведення аналіу даних (Наукові Дані, Телеметрія)
- Повний контрольний журнал всіх операцій, які виконуються в системі (реєстрація, відстеження)
- Усі місця інтеграції повинні бути охоплені повідомленнями логування;
- Безпечне з'єднання між локальними серверами та хмарою;
- Характер основних баз даних NoSQL [4];
- Event brokers використовуються як транспорт подій.

Показники якості:

- Затримка відповіді (пінг) - при нормальній роботі система повинна обробляти запити з затримкою меншьше 200 мілісекунд;
- Масштабованість - При високому навантаженні система повинна успішно масштабироваться вгору і вниз, щоб справлятися з періодичними сплесками трафіку користувачів;
- Пропускна здатність - при нормальній роботі система повинна мати можливість обробляти 250 запитів в секунду;
- Доступність - 99.99 SLA;
- Аварійне відновлення - за будь-яких обставин, у разі регіональної катастрофи в одному з центрів обробки даних, що розміщує додаток, повинно бути автоматичне переключення до другого регіону з мінімальним часом простою.
- Аудит - при нормальній роботі користувач, який пройшов перевірку автентичності, намагається управляти системою. Необхідно вести журнал аудиту дій користувача з інформацією, достатньою для відстеження змін, внесених ним.

- Конфігурація - за нормальної роботи одна зміна конфігурації системи, яка впливає на декілька підсистем, не потребує більше 1 хвилини, і її потрібно застосувати в одному місці.

3.3. Розроблене відмовостійке архітектурне рішення для IoT платформи

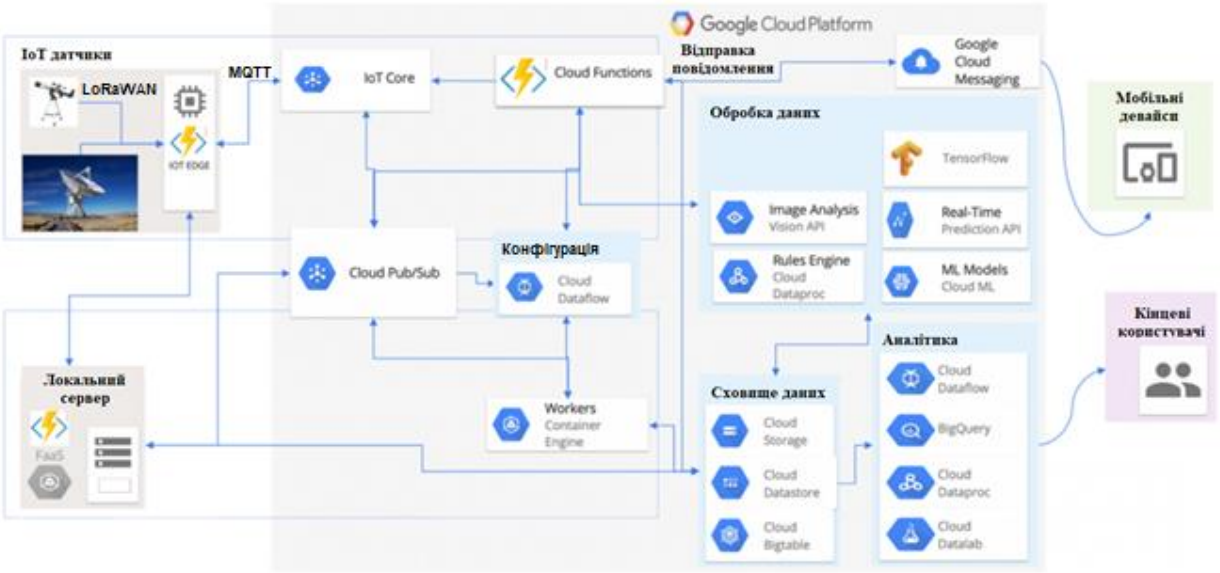

Рис.3.1. Відмовостійка архітектура для IoT платформи

Таке архітектурне рішення (архітектурний дизайн) відповідає вимогам та обмеженням зазначеним в архітектурних драйверах.

1. Таке рішення надає можливість обробки даних через платформи FaaS та послуги хмарної платформи.
2. Ці дані будуть автоматично взяті науковим обладнанням, розташованим по всьому світу.
3. Дані, отримані обладнанням, автоматично збираються та розміщуються у розподіленому сховищі.
4. Звідси дані запускають керований подіями конвеєр обробки даних, який, в свою чергу, виконує деякі етапи, включаючи аналіз, обробку даних як частину процесу прийому даних.

5. Після цього процесу запис цього результату обробки даних створюється в розподіленій NoSQL базі даних.
6. Звідти будь-яка подальша обробка, керована спеціалістами з введення даних, здійснюється за допомогою API, CLI тощо.

   Можливості, які забезпечує інтерфейс командного рядка:

   - Зчитування інформації метаданих;
   - Перегляд міток;
   - Пошук деяких значень;
   - Побудова графіків;
   - CLI-процесор даних також пропонує можливість перегляду журналів аудиту, перегляду інформації про помилки та усунення неполадок;
   - Користувачі (вчені) зможуть увійти в систему обробки даних за допомогою облікових даних.

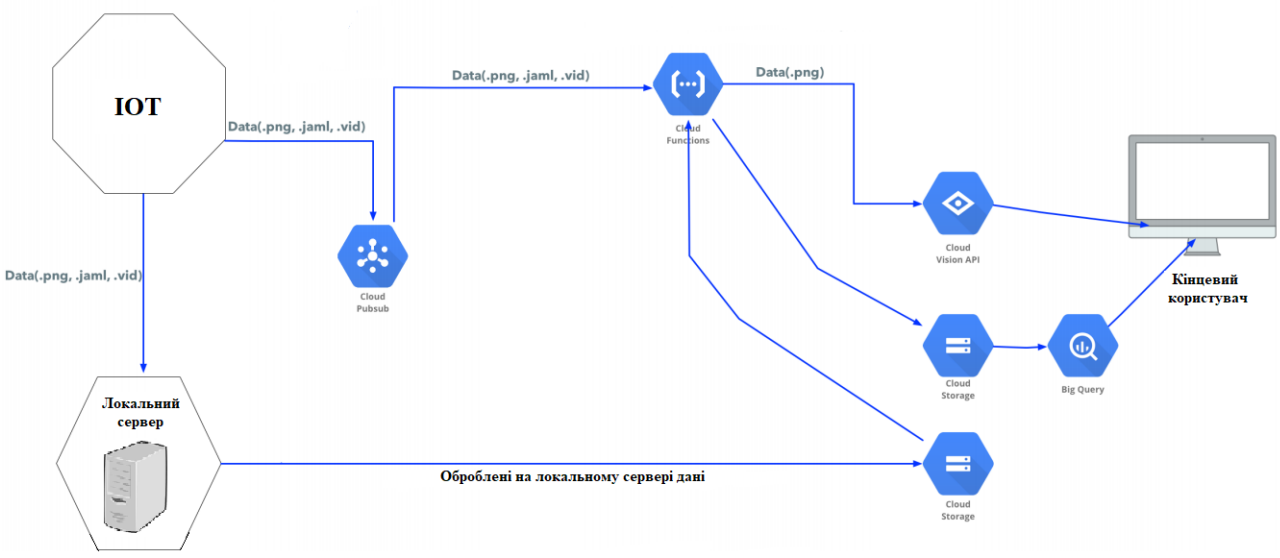

Рис. 3.2. Спрощенна модель архітектурного рішення

Таке рішення гібридне за своєю природою. Обробка даних може розміщуватися на гібридній інфраструктурі, включаючи GCP, AWS, та локальних FaaS.

Вибір використання платформи FaaS хороший тим, що її логіка забезпечує безперебійну інтеграцію з локальним сервером та будь-якими

файловими системами та файловими сховищами. За допомогою використання Cloud PAAS та FAAS зусилля по інтеграції системи введення даних будуть значно зменшені. Також експлуатаційні та управлінські витрати будуть значно нижчими за рахунок центрального управління функціями FaaS.

Система логічно розділена на окремі шари, досягаючи чудової ізоляції та розділення проблем. На даний момент визначені шари - це інтерфейс користувача (CLI та Web), служба, домен та дані:

- Шар інтерфейсу містить модулі, які відповідають за різні можливості Платформи.
- Службовий шар – це програми, включаючи тригери функцій.
- Доменний шар - містить усі модулі, які інкапсулюють логіку домену платформи.
- Шар даних - містить всі модулі, відповідальні за зберігання в системі.
- Аспекти перехресного логування журналів.

Службовий рівень надає функціональність обробки даних за допомогою кінцевих точок RESTful та gRPC.

Частина системи, відповідальна за обробку даних, реалізована відповідно до подієво-орієнтованої архітектури. Подія може бути визначена як "значна зміна стану”. У цьому випадку, це створення нового файлу даних на кінцевій точці або завантаження його в чергу.

Модуль інтерфейсу користувача, містить WEB та CLI версію.

Підсистема обробки даних включає всі необхідні функції та робочі процеси.

Підсистема ведення журналів, складається з журналу FaaS та підмодулів журналу аудиту.

Підсистема зберігання даних, містить модулі, що інкапсулюють сховище даних системи.

Підсистема візуалізації, яка включає всі модулі, відповідальні за обробку зображень та їх зберігання.

Підсистема введення даних містить модулі, відповідальні за функціонування системи прийому даних.

Система обробки даних

Система обробки даних - це конвеєрна система, яка приймає та обслуговує запити REST та виконує обробку даних. Компоненти Backend пропонується реалізувати на мові JavaScript (NodeJS). Сеанси користувачів, обробляються за допомогою токена.

Черга та шина даних

Черга - це хмарний конвеєр даних, який забезпечує швидкий рух даних (необроблені дані, телеметрія, агреговані дані та інші) від приладів IoT & Edge, що належать до різних обсерваторій, у локальне сховище та до хмарних служб Google. Технології, пропоновані для застосування: GCP, Apache, Kafka, PHP, NodeJS тощо.

Функціональні платформи

Платформи FaaS забезпечують можливість розгортання, запуску та управління робочими навантаженнями функцій. Вони також надають необхідні інструменти для інтеграції функцій з компонентами екосистеми (база даних, сховище, тригери і т.д.).

FaaS дозволяє розробникам розробляти, запускати і керувати функціями без складності створення і обслуговування інфраструктури, зазвичай пов'язаної з розробкою і запуском програми. FaaS також надає можливості Serverless для локальних серверів, щоб додатки, які працюють на сервері (мікросервіси, робочі навантаження та інші), могли використовувати функціональну архітектуру.

Робочі процеси

Для забезпечення безперебійної роботи гібридної хмари механізм робочого процесу, об'єднує кілька функцій, розгорнутих в гібридній інфраструктурі. Ця функція називається функцією зчеплення. Розробники розгортають свої служби і функції, визначають робочий процес в файлі YAML і, нарешті, розгортають їх на платформі. Механізм робочого процесу надає кінцеву точку, яка є безсерверною службою.

3.4. Аналіз створеного архітектурного рішення

- Можливість налаштування за допомогою загального сховища конфігурацій на основі розподіленого зберігання об'єктів і інтеграції з обраним сховищем шляхом впровадження простого інтерфейсу постачальника конфігурації і підключення його до контейнера і функцій функціонального додатка.
- Розгортання управляється гібридною інфраструктурою розгортання, що надається хмарною платформою Google, а в локальній мережі - Kubernetes.
- У зв'язку з безсерверною природою рішення воно покладається на базову платформу FaaS для аварійного відновлення.
- Моніторинг доступності та ємності здійснюється за допомогою хмарних платформ і можливостей Kubernetes.
    - Ведення журналів і аудит здійснюються за допомогою хмарних платформ і внутрішніх можливостей системи.
    - Моніторинг продуктивності виконується за допомогою хмарних платформ і внутрішніх можливостей системи.
    - Моніторинг стану виконується за допомогою хмарних платформ і внутрішніх можливостей системи.
- Збір докладних функціональних і нефункціональних вимог системи.

- Основна частина архітектурного рішення працює на гібридній хмарі (хмарна платформа + Serverless).
- Використання IoT Edge:
    - Протокол – MQTT.
    - Збір даних телеметрії з пристроїв Інтернету речей.
    - Збір потокового відео з наукового обладнання.
    - Збір фотографій з наукового обладнання.
    - Впровадження видавця даних (механізм запису агрегованих даних На Локальні Пристрої).
- Використання локального Kubernetes:
    - Отримання даних від кордону Інтернету речей.
    - Встановлення аутентифікації Kafka (на основі сертифікату).
    - Обробка даних за допомогою FaaS (Serverless).
    - Агрегування даних.
    - Обробка Даних (Apache Flink, Apache Beam).

Проаналізувавши розроблене архітектурне рішення, складемо рекомендації по її використанню.

Коли використовувати – рекомендується використовувати архітектурне рішення у випадках, коли є велика кількість IoT датчиків, які створюю велике навантаження, також рекомендується використовувати дану архітектуру, якщо проблема відмовостійкості системи є критичною.

Рекомендований протокол, між датчиками Інтернету речей та базовою станцією – LoRaWAN, який має багато переваг, порівняно з іншими протоколами [3].

Рекомендований протокол між IoT Edge та хмарою – MQTT, який працює за схемою видавець / підписник і відповідає концепції хмарної платформи.

Рекомендавний хмарний провайдер – Google Cloud Platform, оскільки його ресурси використовувалися при розробці архітектурного рішення і він забезпечує необхідний рівень відмовостійкості.

Висновок до розділу 3

Була розроблена відмовостійка архітектура для IoT платформи. Розроблене архітектурне рішення було проаналізоване та надані рекомандеції по його використанню.

Розроблена рахітектура має необхідну відмовостійкість. Також до переваг даної архітектури можна віднести: економічність, масштабованість, час виходу на ринок.

В цілому, розроблене архітектурне рішення цілком відповідає поставленій задачі.

ЗАГАЛЬНІ ВИСНОВКИ

Отже, в роботі була розглянута традиційна архітектура IoT платформи, проаналізовані протоколи Інтернету речей, розглянуті їх переваги та недоліки, Розглянуті варінти підключення датчиків до мережі Інтернету Речей.

Традиційна архітектура, що застосовується для IoT платформи, має свої недоліки, а саме:

1. Недостатня відмовостійкість, через використання звичайних серверів, які не підходять для мережі Інтернету речей.
2. Складнощі масштабування, сервери не зможуть обробляти всплески інформаційних запитів, через що, частина даних може бути втрачена.

Ці проблеми можна виправити, завдяки використанню хмарної платформи та Serverless (Функція як сервіс).

Провівши аналіз хмарної платформи були визначені переваги використання хмарної платформи для побудови інфраструктури IoT. Як ми бачимо, переваги дійсно дуже вагомі, найбільшою перевагою від використання хмарних платформ є відмовостійкість. Також дуже важливими перевагами для архітектури є:

- Масштабованість;
- Доступність системи;
- Місце для зберігання даних;
- Економічність.

Отже, для того щоб підвищити відмовостійкість традиційної IoT архітектури необхідно впровадити використання хмарної платформи та хмарних функцій (Serverless).

В результаті виконання цієї роботи була розроблена відмовостійке архітектурне рішення для IoT платформи, шляхом вдосконалення існуючої

традиційної архітектури. Відмовостійкість в архітектурному рішенні досягається за допомогою:

- Впровадження використання в архітектурному рішення хмарної платформи, що має багато переваг, але найбільшою перевагою є саме відмовостійкість;
- Використання в парі з хмарною платформою – хмарних функцій (Serverless), що також підвищує відмовостійкість;
- Додаткового використання локальних серверів для обробки конфеденційних даних користувачів та зберіганню копії даних надісланих до хмари, щоб, в разі аварії в хмарі мати змогу використовувати локальні сервери, без втрат даних.

В роботі були досягнуті наступні цілі:

1. Проаналізована сучасна IoT платформа;
2. Проаналізовані існуючі архітектурні рішення для IoT платформи;
3. Спроектоване власне відмовостійке архітектурне рішення;
4. Проаналізовані отримані результати та сформовані рекомендації по використанню запропонованої архітектури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інтернет речей (IoT): огляд правових проблем / О.А. Баранов // Інтернет речей: проблеми правового регулювання та впровадження – 2017. – С. 7
2. Застосування розподілених обчислень в телекомунікаційних системах / С. О. Кравчук, Д. А. Міночкін // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. - 2015. - Вип. 50. - С. 41-44. - Режим доступу: http://nbuv.gov.ua/UJRN/Znpviknu_2015_50_9
3. Аналіз безпеки інтернету речей за технологією LoRaWAN / Д.А. Міночкін, О.О. Рибак // Тринадцята Міжнародна науково-технічна конференція ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ – 2019.
4. Забезпечення цілісності даних в мережах промислового «Інтернету речей» / А.М. Давидюк, В.В. Курдеча / Тринадцята Міжнародна науково-технічна конференція ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ – 2019.
5. Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems / Dr. Ovidiu Vermesan – 2013 – С. 15-17
6. The Internet of Everything / Dave Evans // [Електронний ресурс] URL: https://www.cisco.com/c/dam/global/en_my/assets/ciscoinnovate/pdfs/IoE.pdf
7. The Evolution of the Internet of Things. Jim Chase. 2013 [Електронний ресурс] URL: http://www.ti.com/lit/ml/swrb028/swrb028.pdf
8. Internet of Things Роба ван Краненбурга. Лекція в рамках futurodesignlab [Електронний ресурс] URL: http://internetofthings.ru/83-blog/society/5-internet-of-things-roba-van-kranenburga-lektsiya-v-ramkakh-futurodesignlab

9. Internet of Things for Architects / Perry Lea – 2018 – С. 7-9
10. A break in the clouds: towards a cloud definition / M.Lindner, J. Caceres, R. Luis, V. Luis. // ACM SIGCOMM Computer Communication Review. – 2009. – С. 50–55.
11. Mell P. The NIST Definition of Cloud Computing [Електронний ресурс] / P. Mell, T. Grance // National Institute of Standards and Technology. – 2011. – [Електронний ресурс] URL: https://csrc.nist.gov/publications /detail/sp/800-145/final
12. Nurmi, Iiro. 2017. "Application Layer Protocol Support in IoT Cloud Platforms." Iiro Nurmi Blog, April 11. Accessed 2018-06-20.
13. Ilunin, Igor. 2017. "Choosing Your IoT Platform: Should You Go Open Source?" DZone, November 04. Accessed 2018-06-15.
14. Google Cloud. 2018. "Google Cloud IoT - Fully managed IoT services from Google." Google Cloud. Accessed 2018-06-20.
15. Kuppusamy, Sumathi. 2016. "Which platform is best for Internet of things (IoT)?" ActOnCloud Blog, September 02. Accessed 2018-06-20.
16. Serverless Inc., "Serverless Framework," Serverless Inc., 12 October 2015. [Електронний ресурс] URL: https://serverless.com/framework/
17. Amazon Launches Lambda, An Event-Driven Compute Service [Електронний ресурс] URL: https://techcrunch.com/2014/11/13/amazon-launches-lambda-an-event-driven-compute-service/
18. Розподілені системи, поняття та архітектура [Електронний ресурс] URL: http://docplayer.net/69334915-International-scientific-journal-internauka.html