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

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Л.О. Уривський
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка

на тему: «Аналіз архітектури побудови мереж для реалізації концепції Internet of Things»

Виконав:
студент ІІ курсу, групи ТС-61м
Кузянін Олександр Сергійович _____

Керівник:
к.т.н., доцент
Лісковський Ігор Олегович _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ініціали _____

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

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Телекомунікаційних систем

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність (спеціалізація) – 172 «Телекомунікації та радіотехніка» (172.3620.1 «Телекомунікаційні системи та мережі»)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Л.О. Уривський
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Кузяніну Олександру Сергійовичу

1. Тема дисертації «Аналіз архітектури побудови мереж для реалізації концепції Internet of Things», науковий керівник дисертації Лісковський Ігор Олегович, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження Мережа технології Internet of Things.

4. Предмет дослідження Аналіз архітектури мережі.

5. Перелік завдань, які потрібно розробити:

- розглянути основні поняття технології “Інтернету речей”, IoT.;
- дослідити значення цифрового двійника в архітектурі IoT та провести аналіз його складових блоків;
- дослідити основні ризикияким може піддаватися мережа;
- спланувати стратегію забезпечення безпеки;
- проаналізувати інтеграцію приватного блокчейна з технологією IoT;
- розробити методологію розрахунку продуктивності мережі.

6. Орієнтовний перелік графічного (ілюстративного) матеріал

Плакат №1 «Тема, мета та завдання магістерської дисертації»

Плакат №2 «Актуальність та постановка задачі»

Плакат №3 «Архітектура IoT»

Плакат №4 «Складові блоки цифрового двійника»

Плакат №5. «Характеристика аспектів безпеки»

Плакат №6. «Інтеграція приватного блокчейна з IoT»

Плакат №7. «Методологія розрахунку продуктивності мережі»

Плакат №8. «Висновки»

7. Орієнтовний перелік публікацій

1. Olexandr Kuzianin MQTT- THE PROTOCOL FOR INTERNET OF THINGS/ Innovations in Science and Technology: the XVI All-Ukrainian Students R&D Conference Proceedings, (Kyiv, April 18, 2016)/ National Technical University of Ukraine 'Kyiv Polytechnic Institute'. Part II.- Kyiv, 2016. p. 48-49.

2. Кузянін О.С. MQTT- ПРОТОКОЛ ТЕХНОЛОГІЇ «ІНТЕРНЕТ РЕЧЕЙ»/ Міжнародна науково- практична конференція "Перспективи розвитку сучасної науки": матеріали конференції 6-7 травня 2016 року Чернігів, Україна, с. 67-70.8.

3. Кузянин А.С. Шаги повышения безопасности технологии Internet of Things, IoT/ XXII Международная научная конференция "Актуальные вопросы современной техники и технологии": 22 апреля 2016 г., г. Липецк, Российская Федерация, с. 9-11.

Дата видачі завдання 10 вересня 2016 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Розробка, оформлення, узгодження та затвердження технічного завдання на роботу. Аналітичний огляд інформаційних матеріалів. Підбір та опрацювання необхідної науково-технічної літератури.	01.09.2016- 31.12.2016	
2	Огляд областей застосування Інтернету речей.	10.01.2017 - 29.02.2017	
3	Дослідити значення цифрового двійника в архітектурі IoT та проведення аналіз його складових блоків.	01.03.2017 – 30.07.2017	
4	Дослідження основних ризиків яким може піддаватися мережа	01.08.2017 – 31.10.2017	
5	Розробка стратегії забезпечення безпеки та аліз інтеграції приватного блокчейна з технологією IoT	01.11.2017 – 30.01.2018	
6	Аналіз сервісу AWS Amazon.	01.02.2018 – 31.03.2018	
7	Розробка методології розрахунку продуктивності мережі.	01.04.2018 – 30.04.2018	
8	Узагальнення і оцінювання результатів досліджень, підготовка підсумкового звіту. Подання роботи до приймання, та її захист.	01.05.2018 - 20.05.2018	

Студент Кузянін О.С.

Науковий керівник дисертації Лісковський І.О.

РЕФЕРАТ

Обсяг магістерської дисертації складає 108 сторінок, зокрема 27 ілюстрації, 8 таблиць, 65 формул та 24 джерел інформації.

Актуальність теми. Актуальність теми даної роботи обумовлена тим, що технологія IoT дозволить з легкістю контролювати всі аспекти нашого життя. Нинішня технічна революція, та швидкий розвиток мобільних технологій можна вважати першою фазою Інтернету речей. Число фізичних об'єктів, підключених до інтернету зростає з безпрецедентною швидкістю, що реалізує ідею Інтернету речей. Вже сьогодні практичне застосування технології IoT можна знайти в багатьох галузях промисловості, в тому числі землеробстві, будівництві, енергетиці та транспорті.

Тема магістерської дисертації є актуальною, тому що архітектура інтернету речей має складну будову, та потребує вивчення. Цифрові двійники являються невід'ємною частиною IoT. Представлення будь-якого об'єкта реального світу в вигляді інформації знаходиться в цифровому світі з можливістю ідентифікації цього об'єкта з конкретним записом. Запропонована методологія розрахунку продуктивності мережі дозволяє вибрати найкращий варіант транспортного протоколу для побудови конкретної мережі.

Метою випускної кваліфікаційної роботи є аналіз архітектури побудови мережі для реалізації концепції Internet of Things. Розробка розроблено методологію розрахунку продуктивності мережі для оптимального вибору транспортного протоколу.

Методи дослідження. В ході роботи були використані: методи теоретичного дослідження, синтезу та моделювання.

Практичне значення отриманих результатів. Використання запропонованої методології дозволяє вибрати найкращий варіант транспортного протокола для побудови конкретної мережі.

Ключові слова: цифровий двійник, інтернет речей, блокчейн, інформаційна безпека, захист даних, продуктивність мережі.

ABSTRACT

The volume of the master's dissertation is 108 pages, including 27 illustrations, 8 tables, 65 formulas and 24 sources of information.

Actuality of theme. The urgency of this work is due to the fact that the technology IoT will allow you to easily control all aspects of our lives. The current technological revolution, and the rapid development of mobile technology, can be considered the first phase of the Internet of things. The number of physical objects connected to the Internet increases with unprecedented speed, which implements the idea of the Internet of things. Already, the practical application of IoT technology can be found in different spheres of our life, including agriculture, construction, energy and transport.

The topic of the master's thesis is relevant, because the architecture of the Internet of things has a complex structure and must be explored. Digital twins are an integral part of the IoT. The offered methodology of calculation of network performance allows to choose the best variant of a transport protocol for a network.

The purpose of graduation work is to analyze the architecture of network construction for the implementation of the concept of Internet of Things. A methodology for calculating network performance for optimal transport protocol selection is developed.

Research methods. In the work were used: methods of theoretical research, synthesis and modeling.

The practical value of the results. The use of the proposed methodology allows you to choose the best transport protocol for building a specific network

Keywords: digital twin, internet things, blockchain, information security, data protection, network performance.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AES	Advanced Encryption Standard- симетричний алгоритм блочного шифрування
ASIC	Application-specific integrated circuit - інтегральна схема, спеціалізована для вирішення конкретного завдання
BPM	Business Process Management - концепція процесного управління організацією, яка розглядає бізнес-процеси як особливі ресурси підприємства
COM	Component Object Model - платформа компонентно-орієнтованого програмування
CPU	Central processing unit - функціональна частина комп'ютера, що призначена для інтерпретації команд
EPC	Electronic Product Code - електронний код продукту
HTML	HyperText Markup Language – стандарт намова розмітки веб-сторінок в Інтернеті
ISP	Internet Service Provider – Інтернет-провайдер
JSON	JavaScript Object Notation - об'єктнийзаписJavaScript
LAN	LocalAreaNetwork - об'єднання певного числа комп'ютерів на відносно невеликій території
LTE	LongTermEvolution - назва мобільного протоколу передачі даних
MSS	Maximum segment size - максимальний розмір корисного блоку даних в байтах для TCP пакету
NFC	Near Field Communication - технологія бездротового високочастотного зв'язку малого радіусу дії
OTA	Over-the-air programming – оновлення через повітря
PAN	Personal Area Network - мережа, побудована «навкруг» людини
REST	Representational State Transfer – підхід до архітектури мережевих

	протоколів, які забезпечують доступ до інформаційних ресурсів
RFID	Radio frequency identification - радіочастотна ідентифікація
RSA	криптографічна система з відкритим ключем
RTOS	Real-Time Operating System - операці́йна систе́ма реа́льного ча́су
TCP/IP	Transmission Control Protocol / Internet Protocol - набір протоколів мережі Інтернет
UDP	User Datagram Protocol – один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання
QoS	Quality of service - якість обслуговування
WSN	Wireless sensor network - бездротова сенсорна мережа
ОС	Операційна система
ПЗ	Програмне забезпечення
САПР	Систе́ма автоматизо́ваного проектува́ння — автоматизована система, призначена для автоматизації технологічного процесу проектування виробу

ВСТУП

Майже вся наявна сьогодні інформація в інтернеті була отримана та введена людьми, шляхом натискання клавіш, записом та копіюванням даних, чи, наприклад, скануванням штрих-кодів. Однак проблема полягає в тому, що люди мають обмежений час, увагу і точність. Все це означає, що людина не найкращий інструмент по збору даних. Якщо ми будемо мати комп'ютери, що містять в собі всю необхідну інформацію, яка була зібрана автоматично та без додаткової участі, то людство зможе значно скоротити кількість відходів, втрат та витрат.

Актуальність теми даної роботи обумовлена тим, що технологія IoT дозволить з легкістю контролювати всі аспекти нашого життя. Нинішня технічна революція, та швидкий розвиток мобільних технологій можна вважати першою фазою Інтернету речей. Число фізичних об'єктів, підключених до інтернету зростає з безпрецедентною швидкістю, що реалізує ідею Інтернету речей. Вже сьогодні практичне застосування технології IoT можна знайти в багатьох галузях промисловості, в тому числі землеробстві, будівництві, енергетиці та транспорті. Схематичне зображення областей застосування технології зображено на рис. 1.

Дана робота складається з трьох основних частин: в першій частині проведено аналіз архітектури інтернету речей, та його складових елементів. Досліджено значення цифрового двійника в архітектурі IoT та аналіз його складових блоків.

В другій частині описуються загрози, що можуть виникнути в IoT, та методи їх подолання. Визначено вимоги для забезпечення безпеки пристроїв IoT та захищеної взаємодії між ними. Проаналізовано способи та принципи реалізації технології блокчейн для забезпечення безпеки.

Рисунок А Області застосування Internet of Things, IoT

В третій частині було розроблено методологію розрахунку продуктивності мережі для оптимального вибору транспортного протоколу. Проведено аналіз реалізації сервісу IoT, розгорнутого на інфраструктурі AWS Amazon, що може бути реалізовано, на транспортному рівні, протоколом MQTT чи HTTP.

1. АНАЛІЗ АРХІТЕКТУРИ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Що таке Інтернет речей?

У зв'язку з бурхливим розвитком мереж з пакетною комутацією і перш за все Інтернету на початку 2000-х років світове телекомунікаційне співтовариство спочатку виробило, а потім і приступило до реалізації нової парадигми розвитку комунікацій - мереж наступного покоління NGN (Next Generation Networks). Технології NGN вже пройшли еволюційний шлях розвитку від гнучких комутаторів (Softswitch) до підсистем мультимедійної зв'язку IMS (IP Multimedia Subsystem) і бездротових мереж тривалої еволюції LTE (Long Term Evolution). При цьому завжди передбачалося, що основними користувачами мереж NGN будуть люди і, отже, максимальне число абонентів в таких мережах завжди буде обмежена чисельністю населення планети Земля.

Однак останнім часом значного розвитку набули методи радіочастотної ідентифікації RFID (Radio Frequency IDentification), бездротові сенсорні мережі WSN (Wireless Sensor Network), комунікації малого радіусу дії NFC (Near Field Communication) і міжмашинні комунікації M2M (Machine-to-Machine), які, інтегруючись з інтернетом, дозволяють забезпечити простий зв'язок різних технічних пристроїв («речей»), число яких може бути величезним. За розрахунками консалтингового підрозділу Cisco IBSG в проміжку між 2008 і 2009 роками кількість підключених до інтернету предметів перевищило кількість людей, до 2015 року кількість підключених пристроїв досягне 25 мільярдів, а до 2020 року - 50 мільярдів. Таким чином, в даний час відбувається еволюційний перехід від «Інтернету людей» до «Інтернету речей», IoT (Internet of Things).

У загальному випадку під Інтернетом речей розуміється сукупність різноманітних приладів, датчиків, пристроїв, об’єднаних в мережу за допомогою будь-яких доступних каналів зв'язку, що використовують різні протоколи взаємодії між собою і єдиний протокол доступу до глобальної мережі. У ролі глобальної

мережі для Інтернет-речей зараз використовується мережа Інтернет. Спільним протоколом є IP.

Слід особливо відзначити, що Інтернет речей не виключає участь людини. IoT в повному обсязі автоматизує речі, так як він орієнтований на людину і надає йому можливість доступу до речей. Але багато речей зможуть вести себе інакше, ніж ми уявляємо собі сьогодні. У IoT кожна річ має свій унікальний ідентифікатор, які спільно утворюють континуум речей, здатних взаємодіяти один з одним, створюючи тимчасові або постійні мережі. Так речі можуть брати участь в процесі їх переміщення, ділячись інформацією про поточну геопозіції, що дозволяє повністю автоматизувати процес логістики, а маючи вбудований інтелект, речі можуть змінювати свої властивості і адаптуватися до навколишнього середовища, в тому числі для зменшення енергоспоживання. Вони можуть виявляти інші, так чи інакше пов'язані з ними речі, і налагоджувати з ними взаємодію. IoT дозволяє створювати комбінацію з інтелектуальних пристроїв, об'єднаних мережами зв'язку, і людей. Спільно вони можуть створювати найрізноманітніші системи, наприклад, для роботи в середовищах, незручних або недоступних для людини (в космосі, на великій глибині, на ядерних установках, в трубопроводах і т.п.).

Вважається, що першу в світі інтернет-річ створив один з батьків протоколу TCP/IP Джон Ромкі в 1990 році, коли він підключив до мережі свій тостер. Але тільки в 21 столітті в зв'язку з бурхливим розвитком інформаційно-комунікаційних технологій сформувалася концепція IoT і отримала своє практичне втілення. Все почалося з необхідності оптимізації системи логістики та управління системою постачання підприємств. Друга хвиля інновацій була обумовлена необхідністю скорочення витрат в системах спостереження, безпеки, транспорту та ін. Третя була викликана потребою в геолокаційних сервісах. Четверта хвиля буде обумовлена необхідністю дистанційної присутності людини на місці події і вимагатиме його уваги, яка стане можливим завдяки мініатюрним вбудованим процесорам. З розвитком Інтернету речей все більше предметів будуть підключатися до глобальної мережі, тим самим створюючи нові можливості в сфері безпеки, аналітики та управління, відкриваючи все нові і більш широкі перспективи і сприяючи

підвищенню якості життя населення. Передбачається, що в майбутньому «речі» стануть активними учасниками бізнесу, інформаційних і соціальних процесів, де вони зможуть взаємодіяти і спілкуватися між собою, обмінюючись інформацією про навколишнє середовище, реагуючи і впливаючи на процеси, що відбуваються в навколишньому світі, без втручання людини[1].

1.2 Архітектура IoT

Архітектура IoT включає чотири функціональних рівня (рисунку 1.1), описаних нижче.

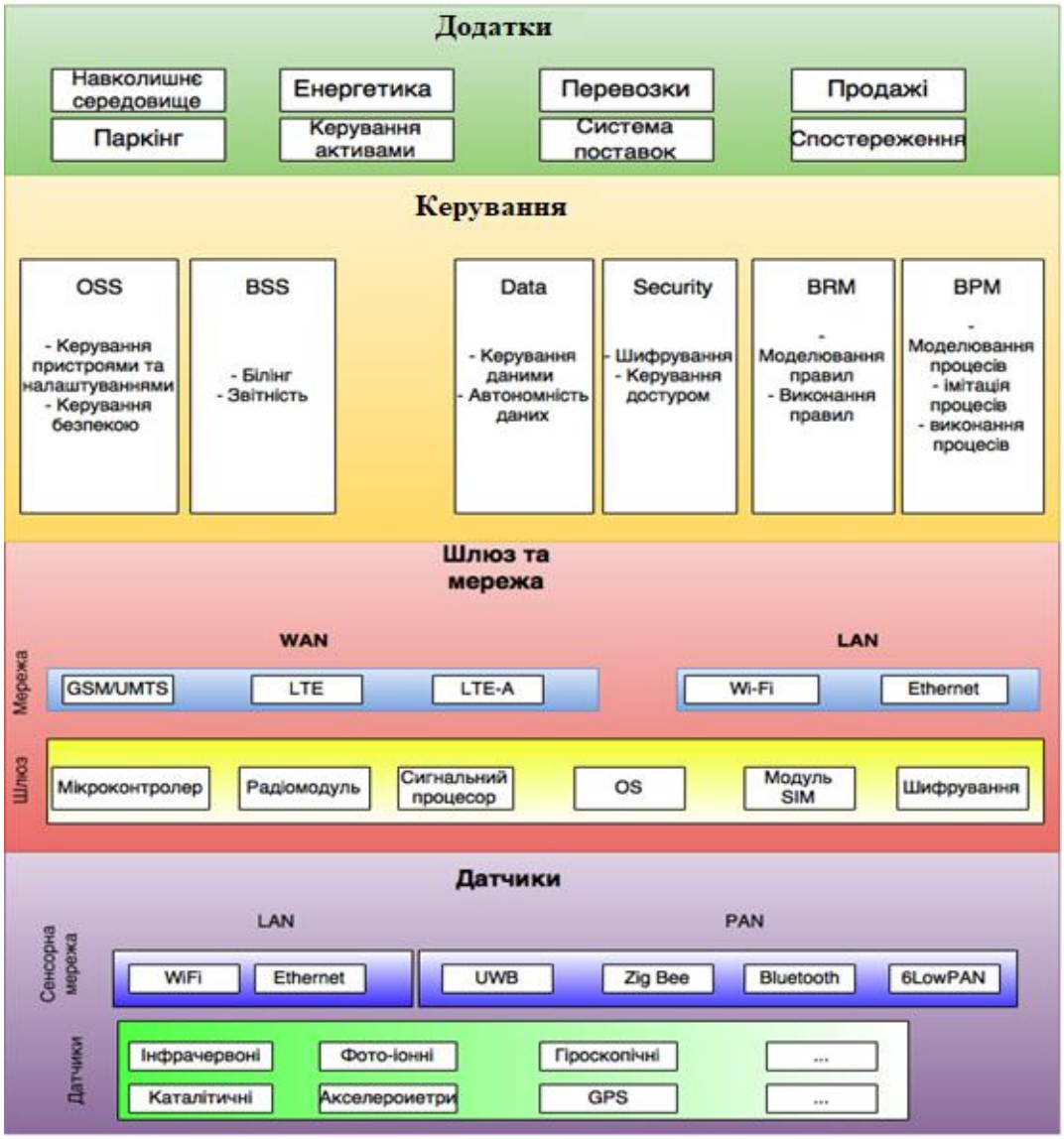

Рисунок 1.1 Архітектура Інтернету речей

1.2.1 Рівень сенсорів і сенсорних мереж.

Найнижчий рівень архітектури складається з «розумних» (smart) об'єктів, інтегрованих з сенсорами (датчиками). Сенсори реалізують з'єднання фізичного і віртуального (цифрового) світів, забезпечуючи збір і обробку інформації в реальному часі. Мініатюризація, яка призвела до скорочення фізичних розмірів апаратних сенсорів, дозволила інтегрувати їх безпосередньо в об'єкти фізичного світу. Існують різні типи сенсорів для відповідних цілей, наприклад, для вимірювання температури, тиску, швидкості руху, місця розташування та ін. Сенсори можуть мати невелику пам'ять, даючи можливість записувати кілька результатів вимірювань. Сенсор може вимірювати фізичні параметри контрольованого об'єкта/явища і перетворювати їх в сигнал, який може бути прийнятий відповідним пристроєм. Сенсори класифікуються відповідно до їх призначення, наприклад, сенсори навколишнього середовища, сенсори для тіла, сенсори для побутової техніки, сенсори для транспортних засобів і т. п.

Більшість сенсорів вимагає з'єднання з агрегатором сенсорів (шлюзом), які можуть бути реалізовані з використанням локальної обчислювальної мережі (LAN, Local Area Network), таких як Ethernet і Wi-Fi або персональної мережі (PAN, Personal Area Network), таких як ZigBee , Bluetooth і ультраширокополосного бездротового зв'язку на малих відстанях (UWB, Ultra-Wide Band). Для сенсорів, які не вимагають підключення до агрегатора, зв'язок з серверами/додатками може надаватися з використанням глобальних бездротових мереж WAN, таких як GSM, GPRS і LTE[2].

Сенсори, які характеризуються низьким енергоспоживанням і низькою швидкістю передачі даних, утворюють широко відомі бездротові сенсорні мережі (WSN, Wireless Sensor Network).

1.2.2 Рівень шлюзів і мереж

Великий обсяг даних, що створюється на першому рівні цифрового двійника численними мініатюрними сенсорами, вимагає надійної та високопродуктивної проводової або бездротової мережевої інфраструктури в якості транспортного середовища. Існуючі мережі зв'язку, що використовують різні протоколи, можуть бути використані для підтримки міжмашинних комунікацій M2M і їх додатків. Для реалізації широкого спектру послуг і додатків в IoT необхідно забезпечити спільну роботу безлічі різних технологій і протоколів доступу в гетерогенній конфігурації. Ці мережі повинні забезпечувати необхідні значення якості передачі інформації, і перш за все по затримці, пропускній спроможності і безпеці. Даний рівень складається з конвергентної мережевої інфраструктури, яка створюється шляхом інтеграції різнорідних мереж в єдину мережеву платформу. Конвергентний абстрактний мережевий рівень дозволяє через відповідні шлюзи декільком користувачам використовувати ресурси в одній мережі незалежно і спільно без шкоди для конфіденційності, безпеки і продуктивності[3].

1.2.3 Сервісний рівень.

Сервісний рівень містить набір інформаційних послуг, покликаних автоматизувати технологічні і бізнес операції двійників: підтримки операційної і бізнес діяльності (OSS / BSS, Operation Support System / Business Support System), різної аналітичної обробки інформації (статистичної, інтелектуального аналізу даних і текстів, прогностичної аналітики та ін.), зберігання даних, забезпечення інформаційної безпеки, управління бізнес-правилами (BRM, Business Rule Management), управління бізнес-процесами (BPM, Business Process Management) та ін.

1.2.4 Рівень додатків

На четвертому рівні архітектури існують різні типи додатків для відповідних промислових секторів і сфер діяльності (енергетика, транспорт, торгівля, медицина, освіта та ін.). Додатки можуть бути «вертикальними», коли вони є специфічними для конкретної галузі промисловості, а також «горизонтальними», (наприклад, управління автопарком, відстеження активів та ін.), Які можуть використовуватися в різних секторах економіки.

1.3 Загальні положення цифрових двійників

Не дивлячись на те, що вперше ця ідея була запропонована вже в 2001 році як спосіб здешевлення розробки продуктів за допомогою цифрових моделей, тільки зараз цей підхід набув широкого застосування. Термін "цифровий близнюк" був визначений доктором технічних наук Майклом Грайєсом в Мічиганському університеті в 2001 році. Він спочатку визначив це в контексті управління життєвим циклом продукту. У своїй роботі він представив концепцію "Digital Twin" як віртуальне представлення того, що було зроблено. Він висунув ідею порівняння Digital Twin з його інженерним дизайном, щоб краще зрозуміти, що було зроблено порівняно з тим, що було розроблено, посилення циклу між дизайном та виконанням. Грівс - професор, дослідник і помічник директора Центру управління життєвим циклом і інноваціями в технологічному інституті Флориди, у своїй статті 2003 року «Цифрові близнюки: перевага у виробництві на основі віртуального прототипу заводу» він писав: «Застосування цифрового двійника відбувається протягом усього життєвого циклу виробу - щоб забезпечити високий рівень якості для споживача, і надати інформацію про те, як він на насправді користується продуктом - для виробника». Незалежно від того, чи є даний об'єкт штучним колінним суглобом або аеродинамічним двигуном, Грівс вважає, що застосовуючи єдиний підхід можна значно скоротити витрати на проектування, виготовлення, експлуатацію та подальше обслуговування виробу.

Цілий ряд компаній, що працюють в області автоматизованого проектування схвалили цю ідею. Компанія PTC, що займається розробкою промислового програмного забезпечення, підтримала ідею цифрового двійника фізичного продукту як для організації, так і для обслуговування, і підтримки продукту. Спільно з цим PTC застосовує технології віртуальної реальності і доповненої реальності. Так, наприклад, наводячи планшет з камерою, на машину, яка потребує ремонту, програмне забезпечення, робить запит до цифрового двійника, і визначає, які дії необхідно виконати працівнику. «Дані з САПР [створені в ході проектування] можуть бути використані для формування « інструкцій доповненої реальності », які паралельно використовують реальні дані, що надходять від реально існуючого виробу. Таким чином, можна знайти, де стався збій і навіть зрозуміти причину збою», - пояснює віце-президент підрозділу САПР компанії PTC Брайан Томпсон. «Це схоже найбільш перспективний промисловий варіант використання технологій Інтернету Речей і VR / AR, заснованих на даних з бази САПР. Спільне використання цих технологій може дати позитивний ефект. Це дуже зручно, коли ви, як сервіс-інженер, можете отримувати якесь цифрове уявлення про те, як вам необхідно працювати з виробом далі ». Задовго до того, як сервіс-інженер добереться до готового виробу, був створений віртуальний близнюк у вигляді САПР моделі, яка потім, протягом декількох років, була значно опрацьована, щоб її можна було вважати цифровим двійником.

Спочатку Грівс виклав три основних вимоги до моделі, яку можна було назвати справжньою цифровою копією. «Я запропонував три вимоги до віртуального двійника, перша вимога - перевірка на відповідність зовнішнього вигляду, але потім я розширив це вимоги до відповідності оцінки сприйняття всіма органами чуттів». Важливо відзначити, що основним в першій вимозі є не тільки зовнішній вигляд. Віртуальний візуальний аналіз включає в себе аналіз продукту на частини і детальний огляд всіх його складових частин[4].

Друга вимога Грівса - віртуальний продукт повинен вести себе реалістично при проведенні різних випробувань, наприклад, таких як продування в цифровий аеродинамічній трубі або аналіз напружень.

Його третя вимога полягає в отриманні інформації від віртуального продукту за допомогою фізичної експертизи, як якщо б користувач міг перевірити реальний продукт, наприклад, визначення величини витрати палива двигуном.

Грівс відзначає зростання промислових можливостей для виконання його вимог: «Де ж знаходиться реалізація цифрового близнюка сьогодні? Коли я тільки починав свою роботу, ми не могли виконати навіть візуальну вимогу - сьогодні це рідко є проблемою. Сьогодні ви вже не зможете відрізнити фотографію справжнього автомобіля від фотографічного рендеринга.

З розростанням систем Internet of Things (IoT) важливість концепції цифрового двійника фізичних речей визвав значний інтерес в останні роки.

Наглядний приклад цієї тенденції можна знайти у доповіді Gartner під назвою " 10 найважливіших стратегічних тенденцій для 2018 року ", опублікованому в жовтні 2017 року. Digital Twins став стратегічною тенденцією № 5 на 2017 рік у цьому звіті. Очікується, що ці цифрові проксі-сервери будуть побудовані з використанням знань фахівців предметів, а також даних в реальному часі, зібраних з пристроїв. Не дивно, що більшість постачальників IoT платформ реалізували деяку форму цифрового близнюка. Вони, як правило, називаються близнюками, тінями, віртуалізацією пристроїв тощо.

Цифровий Двійник (Digital Twin) - це програмний аналог фізичного пристрою, що моделює внутрішні процеси, технічні характеристики і поведінку реального об'єкта в умовах впливів завад і навколишнього середовища. Важливою особливістю цифрового двійника є те, щоб завдати на нього вплив завад використовується інформація з датчиків реального пристрою, що працюють паралельно. Робота можлива як в онлайн, так і в офлайн режимах. Далі можливо проведення порівняння інформації віртуальних датчиків цифрового двійника з датчиками реального пристрою, виявлення аномалій і причин їх виникнення. Цифровий Двійник дозволяє істотно розширити можливості хмарних аналітичних сервісів, які використовуються в концепції Промислового Інтернету Речей (IIoT = Industrial Internet of Things) четвертої промислової революції[5].

Оскільки виробничі процеси стають все більш цифровими, цифровий двійник

тепер знаходиться в межах досяжності. Забезпечуючи компаніям повний цифровий відбиток фізичних продуктів, цифровий двійник дозволяє компаніям швидше виявляти проблеми, точніше прогнозувати результати та створювати кращі продукти. Оскільки ця тенденція тільки розгортається, багато компаній намагаються визначити план дій, які вони повинні дотримуватись, щоб забезпечувати реальну користь від технологій, як на даний момент, так і стратегічно на майбутнє.

Дійсно, цифрові рішення можуть мати величезний вплив на організацію процесів, які ніколи не могла бути реалізовані до появи поєднаних, розумних технологій. Особливий інтерес представляється поняттям “цифрового двійника”: цифровим зображенням у реальному часі фізичного об'єкта або процесу, що допомагає оптимізувати виробничу ефективність.

До недавнього часу цифровий двійник і величезна кількість даних, які він обробляє, часто залишалися ілюзорними для підприємств через обмеженість обчислювальної чи пропускної здатності, вартості обладнання. Однак впливи цих перешкод останнім часом суттєво зменшились. Значно нижчі витрати та покращення потужності та можливостей призвели до експоненційних змін, які дозволяють лідерам поєднувати інформаційні технології та технологію операцій, що дозволяє створювати та використовувати цифрового двійника[6].

Тож чому цифровий двійник настільки важливий, і чому організації повинні це враховувати? Цифровий двійник дозволяє компаніям прослідкувати повний цикл виробничих процесів від розробки та виробництва до закінчення життєвого циклу продукту. Це, в свою чергу, може дозволити їм аналізувати не тільки кінцевий продукт, як це було задумано, але також ефективність системи, яка розробила продукт і як продукт використовується в цільовій галузі. Завдяки створенню цифрового двійника, компанії можуть значно підвищити швидкості виходу на ринок з новим продуктом, покращувати якість продукту, зменшувати кількість дефектів, дозволить швидко та ефективно усувати поломки та помилки, розробляти нові бізнес-моделі для отримання більшого доходу.

Цифровий двійник дозволяє компаніям швидше вирішувати фізичні проблеми, виявляючи їх раніше, прогнозувати результати з великою точністю, створювати

якісніші продукти та, в кінцевому рахунку, краще обслуговувати своїх клієнтів. Завдяки такому розумному дизайну архітектури, компанії можуть реалізувати ці переваги швидше, ніж будь-коли раніше.

Створення цифрового двійника може бути надзвичайно важким завданням, якщо компанія хоче впровадити його у всіх сферах відразу. Ключем до розв'язання може стати впровадження технології в одній сфері, з подальшим розширенням[7]. Але перш ніж впроваджувати технологію ми повинні мати чітке розуміння цілей та підходів до розвитку цифрового двійника, щоб уникнути перевантаження та забезпечити максимальну продуктивність.

1.4 Переваги концепції

Розглянемо типові переваги, які дає ця концепція:

1. Наглядність: цифровий близнюк дозволяє наглядно демонструвати роботу різноманітних машин, а також великих взаємозалежних систем, таких як виробничий цех або аеропорт.
2. Прогноз: використовуючи різні методи моделювання (на основі фізики та математики), модель цифрової близнюка може бути використана для прогнозування майбутнього стану машини.
3. Аналіз: через належним чином розроблені інтерфейси, легко взаємодіяти з моделлю і аналізувати ситуації критичні та недопустимі для цього виробу
4. Механізм документації та комунікації для розуміння та пояснення поведінки: модель цифрової близнюка може бути використана, як механізм зв'язку та документації, який може бути використаний для розуміння, а також для пояснення поведінки окремої машини або їх взаємодії.
5. Об’єднання бізнес-додатків різноманітних систем: модель цифрового близнюка може бути використана для роботи з бізнес-програмами для досягнення максимальних прибутків в бізнесі. На приклад в контексті операцій з ланцюгами

поставок, включаючи виробництво, закупівлю, складування, транспорт та логістику і т. д[6].

Нагальною проблемою для будь-якої компанії, яка починає оцифрування, може стати чітке розуміння користі від інвестицій у створення цифрового двійника, як цей новий підхід може призвести до змін в діяльності компанії та веденні бізнесу, що призведе до переоцінки вартості бізнесу. У минулому створення цифрових двійників було дорогим та мало продуктивним. Розвиток технологій в сферах зберігання інформації та її обчислення, значно розширило можливість для створення цифрового двійника, що у свою чергу, полегшило ведення бізнесу.

Розглядаючи комерційну цінність, яку пропонує цифровий двійник, компанії повинні зосереджуватися на питаннях, пов'язаних зі стратегічною продуктивністю та динамікою ринку, включаючи покращення якості та тривалість експлуатації продукту, швидших циклах проектування, потенціалах нових джерел надходжень та кращим управлінням витратами. Ці стратегічні питання, крім іншого, можуть перетворюватися на конкретні програми, які може реалізувати цифровий двійник. У таблиці 1.1 наведено короткий опис таких значень за категоріями.

Таблиця 1.1 – Переваги використання цифрового двійника

Категорії значення	Потенційні переваги
Якість	- Поліпшення загальної якості - Передбачення якості та попередження дефектів. - Передбачення тенденції зміни якості та готовність до вирішення проблем.
Гарантійні витрати та обслуговування	- Розуміння поточної конфігурації обладнання для проведення продуктивнішого обслуговування. - Проактивно і точніше визначення питання гарантій та претензій, щоб зменшити загальну гарантійну вартість та покращити споживання клієнтів.

Продовження таблиці 1.1

Записи затримок і серіалізації	- Створення цифрових записів серіалізації для частин і сировини, щоб краще керувати відкликаннями та гарантійними вимогами, а також задовольняти вимогам, необхідним для відстеження
Витрати на введення нового продукту та час виконання	- Скорочення часу виходу нового продукту на ринок. - Зменшення загальної вартості випуску нового продукту. - Визначення компонентів довгострокового виробництва та вплив на ланцюжок постачання.
Можливості зростання доходів	- Ідентифікація продукції, що необхідно оновити. - Покращення ефективності та вартості сервісного продукту.

Дивлячись на переваги, перераховані вище, не слід дивуватись, що більшість постачальників IoT зацікавилися цією концепцією. Майже кожна платформа IoT реалізувала певні можливості для цифрових двійників, хоча існують реальні відмінності у термінах їх зрілості та бачення. На широкому рівні ці реалізації зазвичай діляться на дві категорії:

1.5 Спрощена модель пристроїв

Ці реалізації зазвичай використовують текстовий формат обміну даними JSON, та містять два основних набори атрибутів:

а) Набір значень "Спостережуване" або: зазвичай датчики на пристроях зчитують поточні значення та оновлення цих спостережуваних атрибутів. Прикладом цього є поточна швидкість машини (наприклад, 1000 обертів на хвилину).

б) Набір бажаних значень: це значення, яке керуючий додаток бажає встановити на пристрої. Наприклад, програма може встановити швидкість двигуна до 1200 обертів на хвилину.

У доповнення до цих двох основних наборів значень, ці реалізації також зберігають пов'язану інформацію, таку як ім'я або серійний номер пристрою або його поточне місцезнаходження в документі.

По суті, ці спрощені моделі пристроїв являють собою простий формальний асинхронний механізм моніторингу стану пристрою над транспортними протоколами, такими як MQTT / HTTP. Звернемо увагу на те, що асинхронний механізм зв'язку необхідний, оскільки пристрій може бути автономним, або може бути неефективним, коли бекенд хоче з ним зв’язатись. У цьому сенсі моделі реалізують лише “Операційні дані, взяті з датчиків пристрою”.

1.6 Індустріальний цифровий двійник

Ці реалізації зазвичай приймаються постачальниками Industrial IoT, і вони являють собою інформацію від Product Lifecycle Management (PLM) технологія управління життєвим циклом виробів при проектуванні машини (подібно до концепції DTP, викладеної доктором Грейєсом), а також моделі одного пристрою (подібного до частини концепція DTI). Деякі виробники контролюють фізичний стан виробу чи виробництва, інформацію про дизайн та дані з навколишнього середовища в режимі реального часу і представляють їх у графічному вигляді моделі активів / пристроїв. Варто зазначити, що ці моделі базуються на фізичних властивостях машин.

Віртуалізація пристроїв означає створення віртуального представлення фізичного об'єкта або пристрою у хмарі. Це потрібно з кількох причин. По-перше, фізичний актив не завжди може бути пов'язаний з додатками. Наприклад, підключена машина могла б проходити через тунель і моментально втратити зв'язок. Важливо, щоб інша частина програмного забезпечення для бекенда мала змогу запитати останній відомий статус або керувати робочими параметрами, навіть якщо

пристрій не є онлайн / підключеним. По-друге, пристрої підключені за допомогою великої кількості протоколів та методів з'єднання. Бізнес-додатки, такі як система ERP, не повинні обтяжуватися такою складністю. Віртуалізація пристрою пропонує абстракцію для безпечного двостороннього спілкування між світом бізнес-додатків і пристроїв.

У доповнення до базових моделей, подібних до тих, що базуються на простих форматах JSON із спостережуваними та бажаними значеннями, хмара IoT забезпечує віртуалізацію пристроїв, використовуючи потужну семантичну модель. Ця модель семантики пропонує кілька переваг. Однією з помітних переваг цієї моделі є те, що вона дозволяє специфікувати нормальний робочий діапазон атрибутів пристрою. Це значно спрощує реалізацію Edge Computing / Fog. За типової реалізації, для того, щоб виявити порогове порушення даного параметра (наприклад, температура занадто висока), користувачі повинні написати окремий додаток шлюзу, який обробляє це, а потім керує життєвим циклом (розгортання, оновлення, безпека тощо) цієї заявки. За допомогою віртуалізації пристроїв, модель є достатньо інтелектуальною, щоб виявляти аномалії та створювати відповідні сповіщення, не вимагаючи, щоб користувач писав та розгортав Edge Computing.

Крім того, технологія віртуалізації пристроїв може значно оптимізувати об'єм мережевого трафіку, а також оптимізувати механізм передачі через семантичну обізнаність, вбудовану в модель пристрою. Хоча більшість реалізацій зосереджуються на додаткових підходах, використовуючи ефективні протоколи, такі як MQTT, щоб боротися з ціною пропускної здатності мережі, наш революційний підхід автоматизованого статистичного моделювання на краю, заснований на семантичній моделі, призводить до зменшення мережевого трафіку.

Промислові та наукові організації визначають цифрового двійника кількома різними способами. Однак, ні одна група не робить необхідного акценту на процеси, що ним виконуються. Наприклад, згідно з деякими даними, цифровий двійник є інтегрованою моделлю вбудованого продукту, який призначений для відображення всіх виробничих дефектів і постійно оновлюється, щоб включати в себе знос під час використання. Інші широко розповсюджені визначення описують цифровий двійник

як цифрову модель, що підтримує датчики фізичного об'єкта та імітує об'єкт у живій установці[8].

Цифровий двійник може бути визначений, як еволюціонуючий цифровий профіль історичної і поточної поведінки фізичного об'єкту або процесу, це допомагає оптимізувати бізнес процеси. Робота цифрового двійника заснований на вимірах даних в режимі реального часу та великими обсягами даних. Ці вимірювання можуть створити еволюціонуючий профіль об'єкта чи процесу в цифровому світі, який може забезпечити уявлення про продуктивність системи, що працює в фізичному світі.

Дійсно, реальна перевага цифрового двійника - і чому вона має таке значення - це те, що він може забезпечити взаємозв'язок у реальному часу між фізичним та цифровим світом, що дає більш реальні та цілісні вимірювання та прогнозування результатів. Завдяки більш дешевим і потужним обчислювальним можливостям ці інтерактивні вимірювання можна аналізувати за допомогою сучасних архітектур обробки даних та вдосконалених алгоритмів прогнозування в режимі реального часу та в автономному режимі. Вони можуть забезпечити фундаментальні зміни в дизайні та процесі, які майже напевно будуть недосяжними за допомогою сучасних методів.

1.7 Модель цифрового двійника

Цифрові двійники призначені для моделювання складних пристроїв або процесів, які багато в чому взаємодіють із середовищами, для яких важко передбачити результати протягом всього життєвого циклу. Дійсно, цифрові двійники можуть створюватися у різноманітних контекстах для обслуговування різних цілей. Наприклад, цифрові двійники іноді використовуються для імітації конкретних складних пристроїв, таких як реактивні двигуни та великі шахтні вантажні машини, для моніторингу та оцінки зносу та впливів навколишнього середовища, під час використання приладу. Такі цифрові двійники можуть дати важливі статистичні дані, які можуть вплинути на майбутню конструкцію пристрою,

дозволять прогнозувати, та оперативно реагувати на поломки. Цифровий двійник вітрової електростанції може розкрити дані про неефективність експлуатації, та дати поради щодо її покращення[9].

На рисунку 1.2 представлена модель процесів у фізичному світі та його двійників у цифровому світі. Цифровий двійник служить як віртуальна копія того, що насправді відбувається з об’єктом в режимі реального часу. Тисячі датчиків, вмонтованих в пристрій, колективно фіксують дані в широкому діапазоні: від поведінкових характеристик продуктивного обладнання та незавершених робіт (товщина, кольорові якості, твердість, обертаючий момент, швидкість тощо) до умов навколишнього середовища. Ці дані постійно збираються та об'єднуються за допомогою додатка цифрового двійника.

Додаток цифрового двійника постійно аналізує вхідні потоки даних. Протягом певного періоду часу аналізи можуть виявити неприйнятні тенденції фактичного виконання виробничого процесу в конкретному вимірі у порівнянні з ідеальним діапазоном допустимих показників. Таке порівняльне розуміння може призвести до дослідження та потенційної зміни деякого аспекту виробничого процесу в фізичному світі.

Зв’язок між фізичним та цифровим світами, який демонструє рис.1 підкреслює глибокий потенціал цифрового двійника: тисячі датчиків, що приймають безперервні, нетривіальні вимірювання, які транслюються на цифрову платформу, яка, у свою чергу, виконує аналіз в режимі реального часу для оптимізації бізнес-процесу. Модель зображена на (рис. 1.2.) знаходить своє вираження за допомогою п'яти можливих логічних елементів - датчиків та виконавчих механізмів, даних, системи інтеграції, аналітики та цифрового додатка. Ці складові елементи більш детально розглянуто нижче.

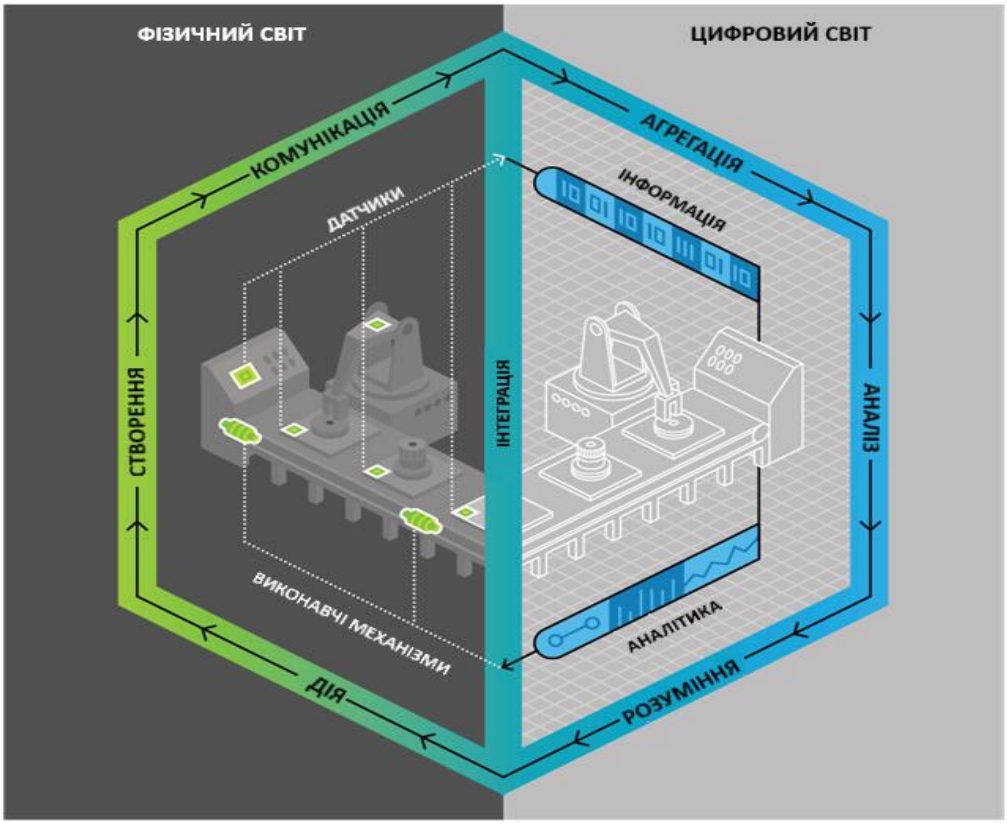

Рисунок 1.2 Модель цифрового двійника

Датчики розташовані на всіх етапах виробництва. Вони створюють сигнали, які дозволяють двійнику зафіксувати дані, що відносяться до фізичного процесу в реальному світі.

- Інформація - реальні робочі і екологічні дані від датчиків, які збираються з навколишнього середовища та об'єднуються з даними підприємства, такими як вартість матеріалів, час до наступного технічного обслуговування корпоративних систем та технічні характеристики проекту. Дані також можуть містити інші елементи, такі як технічні креслення, підключення до зовнішніх каналів даних.

- Під інтеграцією розуміється проміжне середовище через яке дані від датчиків передаються в цифровий світ, яке включає фізичне середовище передачі, інтерфейси зв'язку та безпеку.

- Методи аналітики використовуються для аналізу даних за допомогою алгоритмічного моделювання та процедур візуалізації, які використовуються цифровим двійником для отримання статистичних даних.

- Цифровий двійник. "Цифрова" сторона, зображена на малюнку 1 і є тим

самим цифровим двійником - програма, яка поєднує в собі компоненти, наведені вище, у цифрову модель, близьку до реального часу, у фізичному світі та процесі. Метою цифрового двійника є визначення недопустимих відхилень від оптимальних умов у будь-якому з вимірів. Таке відхилення сповіщає про необхідність оптимізації бізнесу; або двійник має помилку в логіці, або була виявлена можливість заощаджувати витрати, поліпшити якість або досягти більшої ефективності.

- Виконавчі механізми. Якщо в реальному світі потрібне втручання в роботу, то цифровий двійник виробляє порядок дій за допомогою виконавчих механізмів, що призводить до втручання людини, або автоматичного відновлення системи.

Зрозуміло, що світ фізичного процесу (або об'єкта) та його цифровий двійник набагато складніші, ніж модель що зображена вище. І, звичайно, модель на малюнку 1 - це лише одна з можливих конфігурацій цифрових двійників, яка зосереджується на виробничій частині життєвого циклу продукту. Але те, що наша модель прагне показати - це інтегрована, цілісна та ітеративна взаємодія фізичного та цифрового світів[8].

1.8 Вимоги до цифрового двійника

Створення цифрового двійника починається з аналізу та розробки процесу. Стандартні методи розробки технології повинні використовуватися для демонстрації взаємодії бізнес-процесів та людей, що дозволяють впливати на процеси, додатки, інформацію та фізичні ресурси. Створено діаграми, які показують процеси, потреби в даних та типом інформації, що необхідні для створення цифрового двійника. Дизайн процесу доповнюється атрибутами, за яких можна покращити вартість, час або ефективність активів. Вони зазвичай формують припущення базової лінії, з яких повинні починатися переваги використання цифрового двійника.

Ключ до цифрового двійника - фокус на типах інформації, яка буде потрібна протягом життєвого циклу розглянутого активу. Часто важливо структурувати інформацію багаторазово. З цією метою створення канонічної моделі даних може

відігравати важливу роль. Це дає змогу різним системам та додаткам підключатися та обмінюватися інформацією про процеси. Канонічна структура може дозволити різним системам, що інтегруються з цифровим двійником, спілкуватися у простому узгодженому форматі. Це у свою чергу, допоможе зменшити кількість інформації, яка повинна зберігатись та допоможе уникнути необхідності керувати великими структурами даних і може дозволити використовувати цифрового двійника різними способами з більшою гнучкістю.

Розробка референтних архітектур цифрових двійників - тривалий, трудомісткий процес, що передбачає безліч узгоджень, спрямованих на те, щоб максимально абстрагуватися від індивідуальних потреб і технологій. Вироблений еталон повинен виконувати роль загального керівництва, при цьому реалізація всіх його деталей для конкретних застосувань не обов'язкова. Проте можна перерахувати основні вимоги, які відображаються в архітектурних шаблонах цифрових двійників:

- механізми встановлення з'єднань і здійснення зв'язку повинні бути двоточковими або ж забезпечувати розповсюдження інформації між багатьма точками (шляхом багато адресної розсилки або трансляції в розрахунку на будь-який пристрій, здатний виконати прийом);

- системи управління пристроями повинні реагувати на підключення нових пристроїв і зміни в їх конфігурації, а також надавати динамічні механізми поширення змін. Динамічні зміни характерні для стану пристроїв, наприклад сплячий режим і пробудження, підключений і непідключений стан, а також контексту пристроїв, в тому числі місця розташування і швидкість.

- потрібно, щоб в цифрових двійниказ підтримувалася керованість для забезпечення нормального функціонування мережі. Як правило, додатки IoT працюють в автоматичному режимі, без участі людей, проте весь процес їх роботи повинен піддаватися управлінню відповідними сторонами.

- необхідна наявність механізмів збору, аналізу та переміщення даних, які здійснюють вилучення інформації, необхідної для надання сервісів;

- для обробки все зростаючих обсягів даних важливо визначити вимоги до масштабованості систем;
- так як пристрої є гетерогенними і базуються на різних апаратних платформах і мережах, то вони повинні взаємодіяти з іншими пристроями або платформами послуг. Потрібно, щоб забезпечувалася функціональна сумісність гетерогенних і розподілених систем з метою створення та використання різних видів інформації та послуг.
- для всіх елементів цифрових двійників потрібні механізми безпеки, встановлення довірених з'єднань і забезпечення приватності. Потрібно забезпечити, щоб з'єднання між тією чи іншою річчю встановлювалося на основі ідентифікатора цієї речі. Крім того, сюди входить вимога, щоб імовірно гетерогенні ідентифікатори різних речей оброблялися на основі єдиного підходу.
- потрібно, щоб в цифрових двійниках забезпечувався захист недоторканності приватного життя. У багатьох речей є власники і користувачі. Дані вимірювань можуть містити особисту інформацію про їхніх власників або користувачів. Потрібно, щоб забезпечувався захист недоторканності приватного життя при передачі, накопиченні, зберіганні, інтелектуальному аналізі та обробці даних[9].

Референтна архітектура повинна враховувати всі ці вимоги і передбачати додаткові функції, інформаційні структури і механізми, а референтні моделі повинні описувати елементи систем і їх взаємодії. Така модель зазвичай регламентує властивості пов'язаних елементів (користувачів, пристроїв і серверних механізмів), представляючи собою шаблон системи або її реалізації для конкретної предметної області. Архітектуру і моделі зручно використовувати для підготовки техніко-економічного обґрунтування проектів.

Цифровий двійник в складі Інтернету речей концептуально належить до мереж наступного покоління, тому його архітектура багато в чому схожа з відомою архітектурою NGN. Цифровий двійник складається з набору різних інфокомунікаційних технологій, що забезпечують функціонування Інтернету речей,

і його архітектура показує, як ці технології пов'язані один з одним.

1.9 Складові блоки цифрових двійників

Розуміння блоків з яких складається цифровий двійник допомагає отримати більш повне уявлення про його реальне значення і функціональність. У цьому розділі ми обговоримо шість основних елементів, необхідних для забезпечення функціональності цифрового двійника, як показано на (рис. 1.3.)

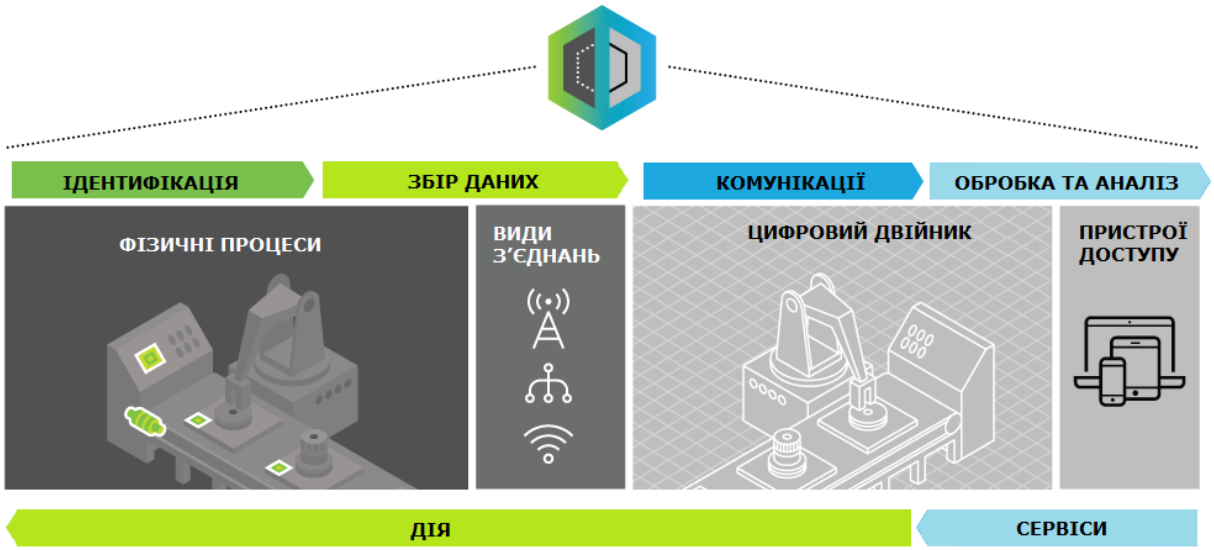

Рисунок 1.3 Основні блоки цифрового двійника

1.9.1 Ідентифікація

Ідентифікація має вирішальне значення для цифрового двійника. Доступно багато методів ідентифікації, такі як EPC (Electronic Product Code), UCODE (ubiquitous codes) і Blockchain. Вміння розрізняти ID об'єкта і його адресу має важливе значення при зверненні до об’єктів.

Ідентифікатор об'єкта позначає його назву, наприклад, "T1" для конкретного датчика температури, а адреса об'єкта вноситься до загальної мережі передачі

даних. Крім того, методи адресації об'єктів включають: IPv6 і IPv4[9].

6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks*) забезпечує механізм стиснення заголовків через IPv6, що робить адресацію IPv6 зручною для бездротових мереж з низьким енергоспоживанням[10].

У такому середовищі, як інтелектуальний будинок або фабрика, де різні пристрої оснащені сенсорами та тісно пов'язані між собою, використання приватного блокчейну може використовуватись для налаштування пристроїв для більш безпечної і надійної роботи відповідно до своїх умов. Приватний блокчейн налаштовується не тільки для виконання аутентифікації користувача, але і для взаємної аутентифікації між пристроями, генеруючи і записуючи деталі операцій і IoT-контракти на підставі сценарію.

Розрізнення між ідентифікацією та адресою об'єкта є обов'язковим, оскільки методи ідентифікації не є глобально унікальними, тому рішення допомагає однозначно ідентифікувати об'єкти. Крім того, об'єкти в межах мережі можуть використовувати публічні IP-адреси, а не приватні. Методи ідентифікації використовуються для забезпечення чіткої ідентичності для кожного об'єкта в межах мережі.

1.9.2 Збір даних

Етап збору даних включає в себе набір фізичних процесів з безліччю датчиків, які вимірюють фізичні процеси та навколишнє оточення.

Вимірювання датчиків можна розділити на дві категорії:

а) оперативні вимірювання, що стосуються критеріїв фізичної ефективності виробничого майна (у тому числі декілька незавершених робіт), таких як міцність на розрив, переміщення, крутний момент та однорідність кольорів;

б) дані навколишнього середовища, що впливають на роботу фізичного активу, наприклад, температура навколишнього середовища, барометричний тиск і рівень вологості. Вимірювання можуть бути перетворені в захищені цифрові повідомлення за допомогою кодерів, а потім передаватися в цифровий двійник.

Інформація від датчиків можуть бути доповнена інформаційною системою, яка складається з процесів, таких як системи виконання виробництва, системи управління ресурсами підприємства, моделі CAD та системи ланцюгів постачання. Це забезпечить цифровий двійник широким спектром постійно обновлюваних даних, які будуть використовуватися як матеріали для аналізу.

Збір даних зв'язаних об'єктів означає зосередження даних всередині мережі і відправку їх в сховище, базу даних, або хмару. Зібрані дані аналізуються, щоб прийняти конкретні дії, що гуртуються на необхідних послугах. Датчики IoT можуть бути інтелектуальні датчики, виконавчі механізми або пристрої зондування. Наприклад, компанії такі як Wemo, revolv і SmartThings пропонують інтелектуальні концентратори і мобільні додатки, які дозволяють користувачам контролювати та використовувати тисячі інтелектуальних пристроїв і приладів всередині будівель, за допомогою своїх смартфонів.

Комп'ютери з однією платою, інтегровані з датчиками і вбудованим TCP / IP і функціями безпеки, як правило, використовуються для реалізації пристроїв IoT (наприклад, Arduino Yun, Raspberry PI, BeagleBone Black, і т.д.). Такі пристрої зазвичай підключаються до центрального портала управління, для забезпечення необхідних даних клієнту[11].

1.9.3 Комунікації

Комунікація- етап обміну даними в реальному часі, двонаправленої взаємодії між фізичним процесом та цифровою платформою. Мережева комунікація - це один з напрямів, що дозволив створити цифрового двійника.

Інтерфейси зв'язку допомагають передавати інформацію від сенсорів до додатків, та виконавчих механізмів. В даній області існує багато варіантів, оскільки датчик, що створює інформацію, теоретично може бути розміщений практично в будь-якому місці, залежно від необхідного функціоналу цифрового двійника: всередині фабрики, будинку, в процесі виробництва або на автостоянці, у безлічі інших місць.

IoT з'єднує різнорідні об'єкти разом, щоб доставити певні інтелектуальні послуги. Як правило, вузли цифрових двійників повинні працювати з використанням низької потужності в каналах з втратами. Приклади комунікаційних протоколів є Wi-Fi, Bluetooth, IEEE 802.15.4, Z-Wave, і LTE-Advanced. Деякі конкретні комунікаційні технології також використовуються, такі як RFID, Near Field Communication (NFC) і UWB (ultra-wide bandwidth)[13].

RFID є першою технологією, яка використовується для реалізації концепції M2M (RFID-міток і зчитувачів). Мітка з чіпом - це пасивний пристрій, в основі роботи якого лежить технологія бездротового високочастотного зв'язку ближньої дії. Коли в зоні дії мітки (від 4 см до 1,5 метра) виявляється, зчитувач, наприклад смартфон, мітка активується і передає на нього інформацію. Основною перевагою міток з RFID-чіпом є висока швидкість передачі даних (до 424 кбіт/сек) при миттєвому взаємодії приладів.

Ще одна комунікаційна технологія Wi-Fi, яка використовує радіохвилі для обміну даними між пристроями в межах дальності 100 м. Wi-Fi дозволяє смарт-пристроям обмінюватися інформацією без використання маршрутизатора.

Bluetooth є технологією зв'язку, яка використовується для обміну даними між пристроями на коротких відстанях з використанням коротких хвиль, щоб звести до мінімуму споживання енергії. Останнім часом особливий інтерес представляє Bluetooth Special Interest Group (SIG), яка забезпечує Bluetooth з низьким енергоспоживанням, а також високу швидкість і можливості підключення IP.

LTE (Long-Term Evolution) спочатку був як стандарт високошвидкісного бездротового зв'язку для передачі даних між мобільними телефонами на базі GSM/ UMTS технологій. Він може фіксувати пристрої, що швидко рухаються і забезпечує багатоадресну передачу. LTE-A (LTE Advanced) являє собою поліпшену версію LTE, включаючи розширену смугу пропускання, яка підтримує до 100 МГц, низхідне і висхідне просторове мультиплексування, розширене покриття, більш високу пропускну здатність і низьку затримку.

1.9.4 Обробка та аналіз

Блоки обробки (наприклад, мікроконтролери, мікропроцесори) і прикладні програми є "мозком" в якому реалізується обчислювальна здатність цифрових двійників та IoT в цілому.

Були розроблені різні апаратні платформи такі як Arduino, UDOO, FriendlyARM, Intel Galileo, Raspberry PI, Gadgeteer, BeagleBone, Cubieboard, Z1, WiSense, Mulle, та T-Mote Sky. Крім того, багато програмних платформ використовуються для забезпечення функціональних можливостей Інтернету речей. До цих платформ відносять операційні системи, які мають життєво важливе значення, так як вони працюють протягом всього часу з моменту активації пристрою. Є декілька операційних систем реального часу RTOS (Real-Time Operating Systems), які добре підходять для інтеграції в IoT.

Якщо ви збираєте великі обсяги даних з пристрою чи іншого джерела, і вам потрібно негайно обробити ці дані, тоді перенесення даних у централізовану базу даних кожного разу означає затримку. Наприклад, скажемо, що на фабричному поверсі є машина, яка аналізує якість продукту на автоматичному конвеєрі. Якщо продукт не відповідає якості, яка визначається оптичним сканером, то він автоматично відхиляється. Передача даних та зображення в централізовану базу даних та в контролер, де визначається успіх виробничого процесу, а потім назад на машину - сповільнює виробничі процеси[14].

Отже, щоб вирішити цю проблему, багато хто пропонує "обчислення на краю". Це не нова концепція, але була нещодавно модернізована. "Обчислення на краю" переносить більшість процесів обробки даних до краю мережі, недалеко від джерела.

Концепція полягає в тому, щоб обробляти дані, які потребують швидкого повернення на пристрій. Проте ці дані також повинні зберігатись централізовано, і, в кінцевому підсумку, всі дані відправляються назад до централізованої системи або хмари, для постійного зберігання та для подальшої обробки.

Хмарні платформи утворюють ще одну важливу обчислювальну частину цифрових двійників. Хмара являє собою набір повністю керованих та інтегрованих сервісів, які дозволяють легко та безпечно підключати, керувати та отримувати дані з глобально розподілених пристроїв у великій кількості, обробляти та аналізувати ці дані в реальному часі та здійснювати операційні зміни і вживати необхідних заходів.

Ці платформи являються зручними для смарт-об’єктів, що зберігають свої дані в “хмарі” та обробляють їх в реальному часі. Також вони є зручними і для користувачів, особливо тих, які використовують великі об’єми зібраної інформації.

На етапі аналізу дані аналізуються та візуалізуються. Вчені та аналітики даних можуть використовувати розширені аналітичні платформи та технології для розробки ітеративних моделей, що дають змогу створювати статистичні дані та рекомендації для прийняття рішень. Аналітичні дані представлені через інформаційні панелі з візуалізацією, висвітлюють недопустимі відмінності в продуктивності цифрової моделі двійника та аналога фізичного світу, та вказують на області, які потенційно потребують розслідування та зміни.

1.9.5 Сервіси

В цілому, сервіси IoT можуть бути розподілені на чотири класи: Identity-related Services, Information Aggregation Services, Collaborative-Aware Services and Ubiquitous Services.

Identity-related Services є самим основним і важливим сервісом, який використовують в інших видах послуг. Кожна програма, яка переносить об'єкти реального світу у віртуальний світ повинна ідентифікувати ці об'єкти.

Information Aggregation Services збирає та об’єднує інформацію, що надходить з сенсорів, для обробки та взаємодії з програмним забезпеченням.

Collaborative-Aware сервіси діють на вершині Information Aggregation Services і використовують отримані дані для прийняття рішень і реагування відповідним чином.

Ubiquitous Services спрямовані на забезпечення сервісів Collaborative-Aware

будь-де, в будь-який час і для будь-кого [14].

1.9.6 Дія

Блок “Дії” - це місце, на якому впровадження попередніх кроків може бути подано назад до фізичного ресурсу та цифрового процесу, щоб вплинути на цифрового двійника. Статистичні дані проходять через декодери, а потім надходять у виконавчі елементи активів. Вони відповідають за механізми управління, оновлення, контролюють ланцюжки постачань та поведінку замовлення, тощо. Ця взаємодія завершує зв'язок замкнутого циклу між фізичним світом і цифровим двійником.

Висновки до розділу 1

В першому розділі було вирішено такі завдання:

- Розгянуто основні поняття технології «Інтернету речей», IoT (Internet of Things).
- Досліджено архітектуру та основні компоненти IoT. Приведено характеристики кожного складового елемента.
- Досліджено значення цифрового двійника в архітектурі IoT та аналіз його складових блоків.

Інтерне́т рече́й — це мережа, що складається з взаємозв'язаних фізичних об'єктів або пристроїв, які мають елементи збору інформації (датчики), а також програмне забезпечення. Поєднання цих елементів дозволяє здійснювати передачу, збір і обмін даними між фізичним світом і комп'ютерними системами.

Цифровий двійник в складі Інтернету речей концептуально належить до мереж наступного покоління, тому його архітектура багато в чому схожа з відомою архітектурою NGN. Цифровий двійник складається з набору різних інфокомунікаційних технологій, що забезпечують функціонування Інтернету речей, і його архітектура показує, як ці технології пов'язані один з одним.

2. АСПЕКТИ БЕЗПЕКИ

2.1 Основні ризики

При розробці системи важливо знати загрози, яким вона може піддаватися, а по завершенні розробки і створення архітектури передбачити належні кошти її захисту. Важливо спланувати стратегію безпеки на самому початку розробки продукту, адже знаючи, як зловмисники можуть скомпрометувати систему, можна спочатку усунути відповідні ризики.

Нижче наведено огляд найбільш поширених ризиків:

- Розкриття інформації**.** Якщо на пристрої працює змінене програмне забезпечення, з нього може бути витік даних стороннім особам. Зловмисник може використовувати отриманий матеріал ключа, щоб втрутитися в обмін даними між пристроєм і контролером, польовим або хмарним шлюзом, а потім отримувати відомості.
- Відмова в обслуговуванні. Пристрій може бути відключено або включено в режимі, в якому неможливо підтримувати зв'язок (це можуть робити навмисно з багатьма виробничими установками). Пристрій можуть перевести в стан, при якому неможливо підтримувати зв'язок.
- Незаконна зміна. Для пристрою можуть налаштувати інше невідоме системі управління стан (параметри, які не належать до еталонних), через що пристрій буде передавати дані, які можна неправильно інтерпретувати.
- Несанкціоноване підвищення привілеїв. Це випадок, коли пристрій, який виконує певну функцію, використовують для примусового виконання інших дій. Наприклад, клапан, запрограмований відкриватися наполовину, можуть налаштувати відкриватися повністю.
- Спуфінг (Spoofing). Зловмисник може отримати матеріал криптографічного ключа з програмного забезпечення або обладнання пристрою, а потім отримати доступ до системи за допомогою іншої фізичної або віртуального

пристрою, використовуючи посвідчення пристрою, з якого було вкрадено матеріал[15].

2.2 Безпека зв'язку

Канал зв'язку повинен бути захищений, для цього застосовуються технології шифрування і перевірки автентичності, щоб пристрої знали, чи можуть вони довіряти віддаленій системі. Нові криптографічні технології, такі як ECC (Elliptic Curve Cryptography), працюють в десять разів краще попередників. Не менш важливим завданням тут є управління ключами для перевірки достовірності даних та достовірності каналів їх отримання. Провідні центри сертифікації (CA) уже вбудували «сертифікати пристроїв» в більш ніж мільярд пристроїв IoT, надавши можливість виконувати перевірку автентичності широкого спектру пристроїв, включаючи стільникові базові станції, багато іншого.

Шифрування, перевірка справжності і керованість незмінно є основою стійкої безпеки. Є відмінні бібліотеки з відкритим вихідним кодом, які виконують шифрування навіть в пристроях IoT з обмеженими обчислювальними ресурсами. Але, на жаль, більшість компаній, як і раніше піддаються ризикам, допускаючи помилки при управлінні ключами для IoT.

Транзакції на 4 млрд доларів в день електронної торгівлі захищені простою і надійною моделлю довіри, яка обслуговує мільярди користувачів і понад мільйон компаній по всьому світу. Ця модель довіри допомагає системам безпечно проводити перевірку достовірності систем інших компаній і взаємодіяти з ними по зашифрованих каналах зв'язку. Модель довіри сьогодні є критичним фактором безпечного взаємодії в комп'ютерних середовищах і грунтується на дуже короткому списку довірених центрів сертифікації (CA). Ці ж CA встановлюють сертифікати в мільярди пристроїв щороку. Сертифікати пристроїв дозволяють, наприклад, перевіряти справжність мобільних телефонів для безпечного підключення до базових станцій, перевіряти справжність інтелектуальних лічильників для електроенергетики, а також приставок в індустрії кабельного телебачення. Надійні

CA дозволяють легко і безпечно генерувати, видавати, реєструвати, контролювати і відкликати сертифікати, ключі та облікові дані, які мають вирішальне значення для надійної перевірки автентичності. З огляду на реалізовані обсяги сертифікатів безпеки для IoT, більшість сертифікатів пристроїв продаються великими партіями за вельми скромну суму грошей за одиницю (в доларовому вираженні йдеться про десятки центів за сертифікат).

Чому перевірка справжності присторою має значення? Небезпечно використовувати дані від неперевірених пристроїв або неперевірених сервісів. Такі дані можуть пошкодити або скомпрометувати систему, передати контроль над обладнанням зловмисникам. Використання надійної перевірки автентичності для обмеження небажаних підключень допомагає вберегти системи від подібних небезпек і зберегти контроль над пристроями і сервісами. Незалежно від того, чи з'єднується пристрій з якимось іншим пристроєм або відбувається обмін даними з віддаленим сервісом, наприклад, хмарним, зв'язок завжди повинен бути захищеним. Всі взаємодії вимагають надійної перевірки автентичності і взаємної довіри. Виходячи з цих міркувань, економія на сертифікатах пристроїв видається спірною.

На щастя, безліч стандартів було розроблено для спрощення розгортання надійної перевірки автентичності всіх ланок ланцюга обміну даними. Стандарти існують для форматів сертифікатів. Надійні центри сертифікації підтримують не лише стандартизовані формати, а так же і кастомні формати. У більшості випадків сертифікатами можна легко керувати віддалено за допомогою стандартних протоколів, таких як Simple Certificate Enrollment Protocol (SCEP), Enrollment over Secure Transport (EST) і Online Certificate Status Protocol (OCSP). Завдяки надійному центру сертифікації, який надає можливість обробляти сертифікати, ключі та облікові дані, фактичну перевірку справжності можна робити за допомогою потужних стандартів Transport Layer Security (TLS) і Datagram TLS (DTLS) - родинни SSL. Взаємна перевірка справжності, коли обидві кінцеві точки перевіряють один одного, має вирішальне значення для якісного захисту систем. В якості додаткового бонусу, одного разу виконавши перевірку достовірності за TLS або DTLS, дві кінцеві точки можуть обмінюватися ключами шифрування або

отримувати їх для обміну даними, які неможливо розшифрувати підслуховуючими пристроями. Для багатьох додатків IoT потрібно абсолютна конфіденційність даних, це вимога легко виконується використанням сертифікатів і протоколів TLS / DTLS. Однак коли конфіденційність не є обов'язковою вимогою, справжність переданих даних може перевірятися будь-якою стороною, якщо вони були підписані під час їх появи на датчику - такий підхід не обтяжує канал шифруванням, що переважно в архітектурі multi-hop, та має вирішальне значення для якісного захисту системі. В якості додаткового бонусу, одного разу виконавши перевірку достовірності за TLS або DTLS, дві кінцеві точки можуть обмінюватися ключами шифрування або отримувати їх для обміну даними, які неможливо розшифрувати підслуховуючими пристроями. Для багатьох додатків IoT потрібно абсолютна конфіденційність даних, це вимога легко виконується використанням сертифікатів і протоколів TLS / DTLS[15].

Часто виникають питання щодо вартості та продуктивності чіпів IoT для криптографічних операцій. Тут потрібно взяти до уваги, що Elliptic Curve Cryptography (ECC) в 10 разів швидше і ефективніше, ніж традиційне шифрування навіть в умовах обмежених обчислювальними ресурсами пристрою. Така швидкість і ефективність досягаються без зниження рівня безпеки. ECC навіть продемонстрував рівень захисту industry best practice, еквівалентний RSA 2048, в тому числі на надзвичайно обмежених в ресурсах чіпах - на 8-bit 1-MHz процесорах і 32-bit 1-KHz процесорах, при споживанні лише мікроват енергії. DTLS, варіант TLS був розроблений спеціально для малопотужних пристроїв, які періодично працюють між циклами сну. І нарешті, ціна таких 32-розрядних чіпів становить всього кілька десятків центів (при розрахунку в доларах), тому ціну або потужність чіпів не вийде використовувати в якості аргументу для зниження вимог щодо захисту нижче розумних порогових значень, коли безпека має значення. В силу описаних факторів пропонуються наступні рекомендації по довжині ключа для перевірки справжності пристрою IoT, де безпека має значення:

- мінімум 224-bit ECC для сертифікатів кінцевих об'єктів з перевагою 256-bit і 384-bit;

- мінімум 256-bit ECC для кореневих сертифікатів з перевагою 384-bit.

Сьогодні ми не можемо уявити собі таку незручність, як ручне встановлення сертифікатів в наші браузери для кожного веб-сервера, в той же час, ми не можемо сліпо вірити будь-якому сертифікату. Ось чому кожен браузер має кілька коренів довіри, за якими верифікуються всі сертифікати. Вбудовування цих коренів в браузери дало можливість масштабувати захист на мільйони серверів в Інтернеті. Оскільки мільярди пристроїв стають онлайн щорічно, в рівній мірі важливо, щоб в них вбудовувалися і коріння довіри, і сертифікат пристрою.

Дані, пов'язані з IoT, повинні зберігатися в безпеці весь час. Наше життя часто залежить від правильності, цілісності і належного функціонування систем більше, ніж від конфіденційності даних. Перевірка справжності інформації, пристроїв і походження інформації можуть мати вирішальне значення. Дані часто зберігаються, кешуються і обробляються декількома вузлами, а не просто передаються з точки А в точку Б. З цих причин дані завжди повинні бути підписані в той момент, коли вони були вперше зафіксовані і збережені. Це допомагає знизити ризики будь-якого втручання в інформацію. Підписання об'єктів даних, як тільки вони були зафіксовані, і ретрансляція підписів з даними, навіть після їх дешифрування, є все більш поширеною і успішною практикою.

2.3 Захист пристроїв

Середовище пристроїв - це фізичний простір навколо пристрою, в межах якого до нього можна отримати фізичний доступ і / або цифровий доступ через однорангову локальну мережу. Під локальною мережею мається на увазі унікально ідентифікована ізольована від загального доступу з Інтернету мережа, до якої можна встановити мережевий міст. Нею може бути будь-яка технологія бездротового радіозв'язку ближньої дії, що дозволяє здійснювати спеціальний робочий обмін даними між пристроями. Нею може бути технологія мережевої віртуалізації, яка створює ілюзію локальної мережі, або загальнодоступні операторські мережі, в яких для однорангового обміну даними потрібні два пристрої, які взаємодіють в просторі

загальнодоступної мережі.

Захист пристроїв - це в першу чергу забезпечення безпеки і цілісності програмного коду. Підписання коду потрібно для підтвердження правомірності його запуску, також необхідний захист під час виконання коду, щоб атаки не перезаписали його під час завантаження. Підписання коду криптографічним ключем гарантує, що він не буде зламаний після підписання і безпечний для пристрою. Це може бути реалізовано на рівнях додатків і прошивок, навіть на пристроях з монолітними прошивками. Всі критично важливі пристрої, будь то датчики, контролери або щось ще, повинні бути налаштовані на запуск тільки підписаного коду.

Пристрої повинні бути захищені і на наступних етапах, вже після запуску коду. Тут допоможе захист на основі хоста, який забезпечує захищеність системи, розмежування доступу до системних ресурсів і файлів, контроль підключень, захист від вторгнень, захист на основі поведінки і репутації. Також в цей довгий список можливостей хостового захисту входять блокування, протоколювання і оповіщення для різних операційних систем IoT. Останнім часом багато засобів хостового захисту були адаптовані для IoT і тепер добре відпрацьовані і налагоджені, не вимагають доступу до хмари і дбайливо витрачають обчислювальні ресурси пристроїв[16].

2.3.1 Захист програмного коду IoT

При включенні кожен пристрій завантажується і запускає певний виконуваний код. Нам вкрай важливо бути впевненими в тому, що пристрої будуть робити тільки те, на що їх запрограмували, а стороннє втручання не зможе перепрограмувати їх. Тобто першим кроком у захисті пристроїв - є захист коду, щоб гарантовано завантажувався і запускався тільки потрібний нам код. На щастя, багато виробників вже вбудували можливості безпечного завантаження в свої чіпи. Схожим чином справи йдуть і з високорівневих кодом - різні перевірені часом клієнтські бібліотеки з відкритим вихідним кодом, на кшталт OpenSSL, можуть використовуватися для

перевірки підпису і дозволу коду тільки з авторизованого джерела. Внаслідок цього все більшого поширення набувають підписані прошивки, образи завантаження і більш високорівнева вбудований код, в тому числі підписані базові програмні компоненти, куди входять будь-які операційні системи. Все частіше зустрічаються не просто підписані прикладні програми, а взагалі весь код на пристрої. Такий підхід гарантує, що всі критичні компоненти систем IoT: датчики, механізми, контролери та реле сконфігуровані правильно - на запуск тільки підписаного коду і ніколи не запустять непідписаний код.

Доброю манерою було б дотримуватися принципу «ніколи не довіряти не підписаному коду». Логічним продовженням було б «ніколи не довіряти не підписаним даними і, тим більше, не підписаним конфігураційним даними». Використання сучасних засобів перевірки підпису і поширення апаратної реалізації безпечного завантаження, ставлять серйозне завдання перед багатьма компаніями - управління ключами і контроль доступу до ключів для підпису коду і захисту програмно-апаратних засобів. На щастя, деякі центри сертифікації пропонують хмарні сервіси, які роблять простіше, безпечніше і надійніше адміністрування програм для підписування коду і гарантують суворий контроль, хто може підписувати код, відкликати підписи.

Виникають ситуації, коли програмне забезпечення потрібно оновити, наприклад, в цілях безпеки, але при цьому необхідно врахувати вплив оновлень на заряд батареї. Операції перезапису даних збільшують споживання енергії і скорочують період автономної роботи пристрою.

З'являється необхідність підписати і оновити окремі блоки або фрагменти таких оновлень, а не монолітні образи цілком або бінарні файли. Тоді програмне забезпечення, підписана на рівні блоків або фрагментів, можна оновлювати з набагато більш тонкої деталізацією, не жертвуючи безпекою або зарядом батареї. Для цього не потрібна обов'язково апаратна підтримка, таку гнучкість можна досягти від передзавантажувального середовища, яка може працювати на безлічі embedded-пристроїв.

Якщо час автономної роботи настільки важливий, чому б просто не

конфігурувати пристрій з незмінної прошивкою, яку ніхто не може змінити або оновити? На жаль, ми змушені припустити, що пристрої в польових умовах схильні до реверс-інжинірингу для шкідливих цілей. Після його проведення виявляються і експлуатуються уразливості, які необхідно виправляти якомога швидше. Обфускація і шифрування коду можуть істотно уповільнити процес реверс-інжинірингу та відбити бажання продовжувати атакувати у більшості зловмисників. Але ворожі спецслужби або міжнаціональні деструктивні організації все-таки здатні це зробити навіть для програм, захищених за допомогою обфускації і шифрування, перш за все тому, код повинен бути дешифрований для запуску. Такі організації знайдуть і скористаються уразливостями, які не були вчасно виправленні. У зв'язку з цим можливості віддаленого оновлення (OTA) мають вирішальне значення і повинні бути вбудовані в пристрої до того, як вони покинуть завод. OTA-оновлення software і firmware дуже важливі для підтримки високого рівня захищеності пристрою. Проте, обфускація, сегментоване підписання коду і OTA-оновлення в кінцевому рахунку повинні бути щільно об'єднані між собою для ефективної роботи [15].

До речі, і сегментоване, і монолітне підписання коду використовують модель довіри на основі сертифікатів, описану в попередньому розділі «Безпека зв'язку», а використання ECC при підписанні коду може забезпечити ті ж самі переваги високого рівня безпеки в поєднанні з високою продуктивністю і низьким енергоспоживанням. У цій ситуації пропонуються наступні рекомендації по довжині ключа для підпису коду IoT, де безпека має значення:

- мінімум 224-bit ECC для сертифікатів кінцевих об'єктів з переважним 256-bit і 384-bit;

- мінімум 521-bit ECC для кореневих сертифікатів, оскільки, як правило, очікується, що підписаний код буде використовуватися роками або навіть десятиліттями після підписання, а підписи повинні бути досить сильними, щоб залишатися надійними протягом такого тривалого часу.

2.3.2 Ефективність хостового захисту

У попередньому розділі ми розглянули перший аспект захисту пристроїв, який визначає основні принципи управління ключами, перевірки автентичності для IoT, підписання коду і конфігурації для захисту цілісності пристрою, основи OTA-управління таким кодом і конфігурацією. Однак, після захисту зв'язку і реалізації безпечного завантаження, необхіден захист на етапі експлуатації. Хостовий захист вирішує цю задачу.

Цифрові двійники стикаються з багатьма загрозами, в тому числі шкідливим кодом, який може поширюватися через перевірені з'єднання, скориставшись уразливими або помилками в конфігурації. В таких атаках часто експлуатуються кілька слабких місць:

- невикористання перевірки підпису коду і безпечну завантаження;
- погано реалізовані моделі перевірки, які можна обійти.

Ці недоліки часто використовуються для установки бекдор, сніфферів, програмного забезпечення для збору даних, можливості передачі файлів для витягання конфіденційної інформації з системи, а іноді, навіть, для маніпулювання поведінкою системи. Особливо тривожить здатність деяких зловмисників експлуатувати вразливості для встановлення шкідливих програм прямо в пам'ять вже працюючих систем. Причому, іноді вибирається такий спосіб зараження, при якому шкідлива програма зникає після перезавантаження пристрою, але встигає нанести величезної шкоди. Це працює, тому що багато систем майже ніколи не перезавантажуються. Для відділу безпеки в цьому випадку ускладняється можливість виявлення уразливості в системі і розслідування походження атаки. Іноді такі атаки відбуваються через IT-мережу, підключену до промислової мережі або до мережі IoT, в інших випадках атака відбувається через Інтернет або через прямий фізичний доступ до пристрою. Як ви розумієте, не важливо, який був вихідний вектор інфекції, але якщо він не виявлений, то перше скомпрометований пристрій як і раніше залишається довіреним і стає провідником для зараження іншої мережі, будь то автомобільна мережа транспортного засобу або ціла виробнича

мережа заводу. Таким чином, безпека повинна бути комплексною.

На щастя, в поєднанні з надійною підписом коду і моделлю перевірки, хостовий захист може допомогти захистити пристрій від безлічі небезпек. В такому захисті використовується ряд технологій, в тому числі харденінг, розмежування доступу до системних ресурсів, пісочниця, захист на основі репутації і поведінки, захист від шкідливих програм і, нарешті, шифрування. Залежно від потреб конкретної системи комбінація цих технологій може забезпечити найвищий рівень захисту для кожного пристрою [16].

Харденінг, розмежування доступу до ресурсів і пісочниця захистять всі «двері» в систему. Вони обмежують мережеві підключення до додатків і регламентують вхідний і вихідний потік трафіку, захищають від різних експлойтів, переповнення буфера, цілеспрямованих атак, регулюють поведінку додатків, при цьому дозволяють зберегти контроль над пристроєм. Такі рішення ще можуть використовуватися для запобігання несанкціонованого використання мобільних носіїв, блокування конфігурації та налаштувань пристрою і навіть для деескалації користувальницьких привілеїв, якщо потрібно. Хостовий захист має можливості аудиту і оповіщення, допомагаючи відстежувати журнали і події безпеки. Технології на основі політик можуть працювати навіть в середовищах без підключення до інформаційної мережі або при обмеженій обчислювальної потужності.

Технологія захисту на основі репутації може використовуватися для визначення сутності файлів по їхньому віку, поширеності, розташування і для виявлення небезпек, що не виявляються іншими засобами, а також давати уявлення про те, чи слід довіряти новому пристрою навіть при успішній перевірці автентичності. Таким способом можна ідентифікувати загрози, які використовують мутуючий код або адаптують свою схему шифрування, просто відокремлюючи файли з високим ризиком від безпечних, швидко і точно виявляючи шкідливі програми, незважаючи на всі їхні хитрощі.

Зрозуміло, поєднання застосовуваних технологій буде залежати від конкретної ситуації, але вони можуть об'єднуватися для захисту пристроїв, навіть в середовищах з обмеженими обчислювальними ресурсами.

2.4 Контроль пристроїв

Отже, ми знаємо, що реверс-інжиніринг пристроїв рано чи пізно буде проведено, уразливості будуть виявлені, а для пристроїв необхідно буде надавати поновлення OTA (віддалено). Звичайно, механізми оновлення додають складності архітектурі, тому багато інженерів намагаються уникати їх на свій страх і ризик. На щастя, хороший механізм OTA може використовуватися для багатьох цілей, не тільки для виправлень програмного забезпечення і функціональних оновлень, але також:

- Оновлення конфігурації.
- Управління телеметрією безпеки для аналітики захищеності.
-Управління телеметрією для контроля правильності функціонування пристро.
- Діагностики та відновлення.
- Управління обліковими даними доступу до мережі.
- Управління правами / привілеями і безлічі інших завдань.

Звичайно, все перераховане вище має виконуватися безпечно і надійно, тут потрібно найбільш ретельний підхід до підписання коду і організації передачі файлів. Існують стандарти управління довкіллям software і firmware на кожному пристрої, включаючи конфігурацію. Багато виробників, зокрема, Open Mobile Alliance (OMA), підтримують такі стандарти. Деякі з рішень масштабуются для управління мільярдами пристроїв.

Управління безпекою кожного пристрою може припускати управління конфігурацією за допомогою хостового захисту, яку ми розглянули вище. Природно, деякі технології безпеки передбачають поновлення OTA для контенту безпеки, наприклад, чорні і білі списки, евристика, сигнатури IPS і дані про репутацію. Також існують технології безпеки, засновані на політиках, яких оновлення потрібне тільки при перевстановлення на пристрої програмного забезпечення, наприклад, для додавання функціональних можливостей. Проте обидва типи технологій можуть генерувати телеметрію безпеки, яка має велике

значення при зіткненні з цілеспрямованими атаками.

Зрозуміло, компоненти безпеки не єдині в цифрових двійниках, якими необхідно керувати безпечно і надійно. Більшість пристроїв генерують телеметрію або дані з датчиків, які потрібно також безпечно і надійно збирати і передавати в місця зберігання та аналізу. Багато пристроїв вже містять в собі функції контролю, якими потрібно керувати через конфігураційні параметри, а ті в свою чергу безпечно і надійно зберігати і оновлювати. На щастя, інфраструктури управління пристроями, які використовують загальноприйняті безпечні протоколи, можуть застосовуватися і для захищеного управління основними функціями пристрою, контентом безпеки і телеметрії пристрою. Фактично подібні моделі адаптуються для OTA-керування автомобілями і використовуються для безпечного і надійного управління торговими автоматами. Деякі з інфраструктур управління комбінують агентські та без агентські протоколи управління, тоді як пристрої випускаються з підтримкою стандартизованого управління для спрощення функцій контролю. Окремі інфраструктури управління можуть додатково поєднувати всі ці методи управління з розумінням інформації, отриманої від аналізаторів трафіку.

У ситуації, що склалася цифрові двійники повинні мати вбудовані можливості оновлення через бездротові мережі. Відсутність цієї можливостей залишить пристрої не захищеними від загроз і вразливостей протягом всього терміну служби. Зрозуміло, оновлення OTA може застосовуватися ще для управління конфігураціями пристроїв, контентом безпеки, обліковими даними, а також для розширення функціональних можливостей пристроїв, збору телеметрії і даних програмного оточення, для доставки оновлень і багато чого іншого. Однак з додатковою функціональністю або без неї базові можливості оновлення і управління захищеністю повинні бути передбачені ще на етапі проектування цифрових двійників.

2.5 Контроль взаємодій в мережі

Деякі загрози зможуть подолати будь-які вжиті заходи, незалежно від того,

наскільки добре все захищено. Тому вкрай важливо мати аналітику безпеки в мережах, де використовуються цифрові двійники. Системи для аналітики безпеки допоможуть краще зрозуміти мережу, помітити підозрілі, небезпечні або зловмисні аномалії.

Все більше систем повинні зв'язуватися один з одним, тому постає питання, «чому довіряти?». Сертифікати пристроїв можуть містити інформацію про походження і тип пристрою. Проте на питання про те, чи потрібно довіряти цьому пристрою, в кінцевому підсумку повинні будуть відповідати інші служби, наприклад, засновані на репутації, або на каталогах. Такий каталог здатний не тільки відслідковувати інформацію про безпеку для кожного пристрою і систем, але ще відстежувати і керувати привілеями і повноваженнями, якими пристрої та системи наділяють один одного. Фактично кожен з нас оточеним великою кількістю пристроїв, а такі каталоги допомагають розібратися з пристроями, що відіграють важливу роль для користувача. Дана технологія робить можливим швидкий пошук віддаленого пристрою через каталог і, сприяє прискоренню прийняття рішення про використання даних з чужого пристрою.

Концепція каталогів вельми перспективна, але на сьогодні не являється основоположною технологією, ні ключовим елементом в контролі взаємодій в мережі. Ця перспективна концепція включена в список для того, щоб дати попередній огляд викликів, що стоять перед багатьма компаніями, і наводимо приклад, як можна впоратися зі складними масштабними завданнями. Деякі компанії вже зіткнулися з подібного роду проблемами, оскільки вони несуть відповідальність за захист мільйонів пристроїв.

2.6 Еволюція парадигми

Більшість Цифрових Двійників являють собою закриті системи. Покупці не зможуть додавати програмне забезпечення безпеки після того, як пристрої покинуть завод. Таке втручання анулює гарантію, а часто просто не представляється можливим. З цієї причини, захисні функції повинні бути спочатку вбудовані в

пристрої, щоб вони були безпечними за своєю архітектурою. Для більшої частини індустрії така «безпека всередині», тобто вбудована при виготовленні пристрою на заводі - це новий спосіб забезпечення захисту, це стосується і класичних технологій безпеки, таких як шифрування, перевірка справжності, перевірка цілісності, запобігання вторгнень і можливості безпечного оновлення. З огляду на тісний зв'язок апаратного і програмного забезпечення в моделі, іноді простіше, щоб програми для захисту використовували розширення функцій апаратної частини і створювали «зовнішні» рівні безпеки. Вже сьогодні багато виробників чіпів вбудовують функції безпеки в обладнання. Але апаратний рівень - це всього лише перший шар, необхідний для комплексного захисту зв'язку і пристроїв. Комплексний захист вимагає інтеграції функцій управління ключами, захисту на основі хоста, інфраструктури OTA і аналітики безпеки, про що ми згадували раніше. Відсутність навіть одного з наріжних каменів у фундаменті безпеки залишить широкий простір діям зловмисників. необхідний для комплексного захисту зв'язку і пристроїв. Комплексний захист вимагає інтеграції функцій управління ключами, захисту на основі хоста, інфраструктури OTA і аналітики безпеки, про що ми згадували раніше. Відсутність навіть одного з наріжних каменів у фундаменті безпеки залишить широкий простір діям зловмисників. необхідний для комплексного захисту зв'язку і пристроїв. Комплексний захист вимагає інтеграції функцій управління ключами, захисту на основі хоста, інфраструктури OTA і аналітики безпеки, про що згадувалось раніше [15]. Відсутність навіть одного з елементів у фундаменті безпеки залишить широкий простір дій зловмисників.

2.7 Аналітика безпеки та реакція на погрози

Незалежно від того, наскільки добре ви захистили пристрій, код, зв'язок, і, навіть застосовуючи кращу з можливих інфраструктур управління OTA, в розпорядженні деяких зловмисників цілком достатньо ресурсів і можливостей для подолання захисту. Таким чином, стратегічні загрози вимагають стратегічних технологій для мінімізації негативних наслідків. Аналітика безпеки може

використовувати телеметрію безпеки з пристроїв і мережевого обладнання, щоб давати чітке уявлення про те, що відбувається в обчислювальному середовищі, включаючи виявлення прихованих загроз. Ці ж дані використовуються в аналітичних системах в рамках вирішення завдань оптимізації роботи цифрового двійника.

Моніторинг і аналітика часто можуть бути розгорнуті в якості тимчасового рішення в середовищах, де розгортання інших засобів захисту займе досить великий час. Давайте розглянемо приклади. Legacy-прилади в промислових системах управління (виробництво, нафта і газ, комунальні послуги) не можна модифікувати до заміни системи цілком. Автоматизовані автомобілі, чиї мікроконтролери глибоко вбудовані, вже знаходяться на дорозі, і очевидно, їх не можна просто демонтувати і замінити на нові. У середовищі охорони здоров'я виробники зовсім забороняють лікарням модифікувати обладнання для додавання захисних функцій. У таких випадках рішення для виявлення аномалій можуть бути надзвичайно корисними. Багато мереж характеризуються чітко визначеними шаблонами поведінки, а відхилення в таких системах легко ідентифікуються. Широке різноманітність промислових протоколів і протоколів IoT ускладнює ситуацію, але нові технічні рішення, які використовують просунуте машинне навчання, успішно вирішують аналітичні завдання. Засоби захисту в пасивному режимі «виявлення» будуть вести себе менш агресивно, ніж в активному режимі «запобігання», так як помилкові спрацьовування не впливатимуть на роботу системи загалом [1].

Іншим варіантом захисту є шлюзи, наприклад, між застарілими і більш сучасними та захищеними середовищами. Оскільки атака в одній частині може передаватися по всій мережі, то її потрібно зупиняти раніше. Однонаправлені шлюзи даних використовуються для гарантованої односпрямованої передачі між відкритими мережами і мережами з обмеженим доступом, між високо пріоритетними об'єктами розподіленого моніторингу та централізованої аналітики, шлюзами між промисловими і загальними IT-мережами, між головним блоком транспортного засобу та інформаційно-розважальними системами.

У більшості випадків замовники можуть співпрацювати з компаніями, які

спеціалізуються на захисті інформації, для використання їх існуючої інфраструктури big data аналітики безпеки і великомасштабних систем збору подій безпеки по всьому світу для того, щоб отримувати, аналізувати і обмінюватися інформацією про всілякі мережі і екосистеми. Досить активно така діяльність ведеться в різних галузях, в першу чергу в сфері рітейлу і критичною інфраструктури, так як це дає гарантії швидкого оновлення системи цілком для захисту від будь-яких виникаючих загроз.

Найчастіше експертні знання в області безпеки, необхідні в аналітиці для виявлення складних загроз, можуть виходити за рамки можливостей компаній, які не спеціалізуються в області IT та ІБ. З цих причин багато організацій звертаються до аутсорсингу, щоб можна було покластися на експертів, що виконують моніторинг і аналітику. У деяких випадках компанії будують свої власні сховища телеметрії безпеки і надають доступ до цього сховища декільком партнерам по аналітиці для спільного пошуку цілеспрямованих атак. Деякі аналітичні продукти і платформи надають API і SDK для забезпечення спільного доступу, безпечного взаємодії та обміну даними, наприклад, можна надавати права заданому списку партнерів до певних даних і обмежувати доступ для інших.

У прикладі з об'єднаними промисловими і IT-мережами ми рекомендується створювати єдину площину даних, що охоплює обидва середовища, для отримання в пріоритетному порядку дані про різні погрози та мінімізацію ризиків проникнення загроз з одного середовища в іншу. Такі рішення повинні працювати з різними виробниками, всілякими пристроями та протоколами, щоб клієнти отримували цілісне уявлення про свою мережі.

Принцип «виявлення і реагування» може доповнювати технології посиленого захисту для відображення переважної більшості атак, а також для мінімізації ризиків збитку від найбільш грізних супротивників[14].

2.8 Використання блокчейн для підтримки IoT-додатків

Використання приватних блокчейнів має практичну цінність, якщо вони можуть взаємодіяти з публічним блокчейном, який вже задіяний. Для цього створюється платформа і публічний блокчейн, який може ефективно використовуватися з декількома приватними блокчейнами.

Іншими словами, шляхом використання блокчейна, стало можливим здійснювати мікроплатежі за рамками P2P-розрахунків публічного блокчейна. У контрольованій мережі приватного блокчейна, ми впроваджуємо Машинну Валюту для двосторонніх договорів і платежів між елементами цифрового двійника і, тим самим сприяємо більш доступним, надійним і безпечним процесам.

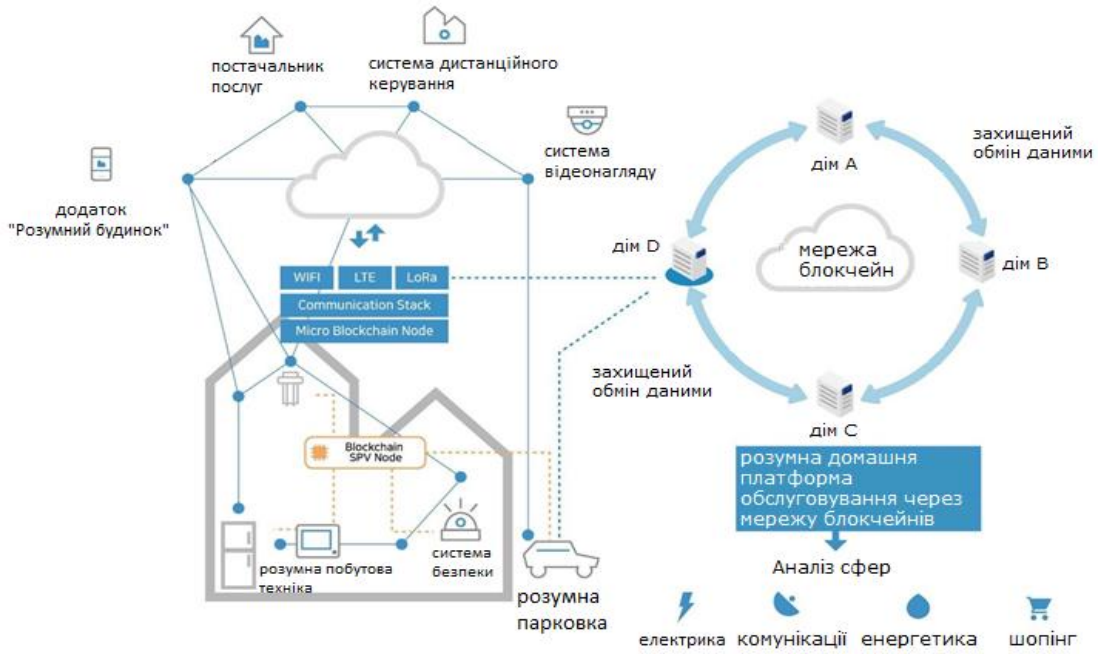

Рисунок 2.1 Приклад приватного блокчейна з використанням IoT

Наприклад, користувач хоче керувати пристроєм за певними можливостями або бюджетом. Користувач встановлює певне значення (величину або суму; тут мається на увазі значення IoT-контракту) в пристрої управління, встановленому в смартфоні, комп'ютері, смарт-ТВ або пульті керування, потім переміщує дані на

відповідний пристрій. Далі, частина пристрою, що обробляє фактичні дані, передає значення контракту в вимірювальні пристрій і починає операцію. В цей час активований пристрій буде функціонувати, поки не передасть сигнал із зазначенням, що певне значеня контракту досягло рівня, зазначеного вимірювальним пристроєм. Пристрій, що виконує певні дії передає дані, коли досягнуто певного значення, або коли пристрій контролю запитує підтвердження поточного робочого стану, щоб користувач міг підтвердити дані. В цей час в процесі аутентифікації необхідно підтвердити комунікаційні процеси між пристроєм контролю користувача, пристроем, що виконує певні дії і вимірювальним пристроєм. Крім того, дозволений контракт може використовуватися для здійснення платежу, а у випадку з приватним блокчейном, токен користувача може застосовуватися в якості платіжного токена для активації пристрою.

2.9 Інтеграція з публічними блокчейнами

З огляду на те, що первісна блокчейн-технологія стає популярною завдяки своїй децентралізації, прозорості, легкості у використанні і надійності, блокчейн використовують в таких сферах, як криптовалюта, перевірка присутності, ринки прогнозів і міжнародні фінанси. Через збільшення кількості транзакцій і даних необхідно розглянути питання про те, як публічні блокчейни можуть вийти за рамки своїх можливостей.

Підключення між публічними блокчейнами вже виконано за допомогою обміну даних через біржі. Тобто в разі зареєстрованого блокчейна на біржах, обмін буде відбуватися з перемиканням. Перевага цього підходу полягає в тому, що люди можуть просто ідентифікувати і використовувати послуги, так як вони виконуються таким же чином, як обмін валютами, наприклад, Корейська вона і долар США.

Публічні блокчейни можуть приєднуватися за ієрархічним принципом. Наприклад, є блокчейни нижнього рівня, що використовуються по регіонах або по округах, і ці результати інтегруються в блокчейні середнього

рівня. В цьому випадку блокчейн середнього рівня набирає масштаби міста / області і може мати кілька блокчейнов нижнього рівня. Іншими словами, всі транзакції, зібрані на блокчейні нижнього рівня, передаються на блокчейн середнього рівня. Блокчейн верхнього рівня може розглядатися в якості структури, в якій інформація по всіх блокчейнах нижнього рівня інтегрована з блокчейном державного масштабу. Цей простий приклад може використовуватися в різних формах блокчейн-мережі у вигляді дерева. У цьому конкретному випадку, взаємодія між публічними блокчейнами може бути пов'язана через окремий сервіс, який виступає в якості біржі. Таке формування може бути з'єднувальною ланкою блокчейна.

Приватна блокчейн-мережа або дозволений блокчейн - це блокчейн з привілеями доступу. Його конфігурація означає, що до нього не може отримати вільний доступ будь-який вузол, в порівнянні з публічним блокчейном. Таким чином, щоб отримати доступ до приватного блокчейну з публічного блокчейна, необхідний з'єднуючий вузол. Цей вузол повинен мати всю інформацію по конфігурації приватного блокчейна, щоб надати такий же рівень доступу, тим часом дозволяючи розміщення в публічному ланцюжку. Підводячи підсумки, можна сказати, що до приватного блокчейну можна отримати доступ, тільки коли адміністратор приватного блокчейна дає доступ через попередню аутентифікацію і реєстрацію.

У випадку зі спеціальної транзакцією, адміністратор може наділити окремими привілеями і відправити їх на вузол або пристрій. Для використання приватного блокчейна, користувач повинен бути зареєстрований після такої аутентифікації в приватному блокчейні.

Токени можуть використовуватися в рамках певного підприємства або для певної мети. У цьому випадку, співвідношення обміну між двома блокчейнами може визначатися / змінюватися через біржу.

Щоб знайти способи ефективного здійснення транзакцій, що виконуються на широкомасштабних IoT-пристроях, вченими було досліджено структуру / швидкість обробки транзакцій IoT-а і мережу Ефіріум нового покоління. У тестовій мережі

вчинення великих транзакцій від десятків сотень до сотень тисяч транзакцій в секунду показало, що оброблялася лише обмежена кількість транзакцій, в залежності від швидкості фізичної мережі / фізичного обчислювального ресурсу. Тому кращим рішенням для управління транзакціями високого рівня - це обробка за допомогою множинних, окремих приватних блокчейнів і інтеграція інформації через окремі коріння блокчейна.

Зокрема, публічні блокчейни в глобальному середовищі мають досить обмежену швидкість транзакцій (біткоіни - близько 2 транзакцій / в секунду, а Ефіріум - близько 5 транзакцій / в секунду) через проблеми з мережею і синхронізацією блоків на декількох вузлах. Hdac (Платформа цифрового обслуговування наступного аокоління) також не може оптимізувати швидкість в глобальному Інтернет-середовищі через зміни швидкості мережі і затримок синхронізації блоків. Тому в майбутньому блокчейн для обробки великої кількості транзакцій може стати мережею блокчейнів з ієрархічною або розподіленою структурою, що складається з численних приватних блокчейнів для різних цілей [17].

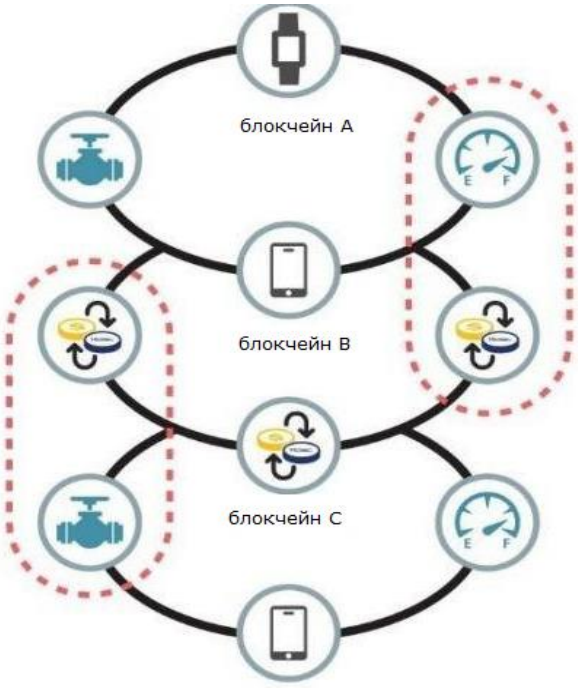

Рисунок 2.2 Інтеграція блокчейнів за ієрархічним принципом

2.10 Надійність IoT-пристроїв в блокчейні

Сполучна структура IoT і мережева структура блокчейнів дуже схожі, див. Малюнок нижче.

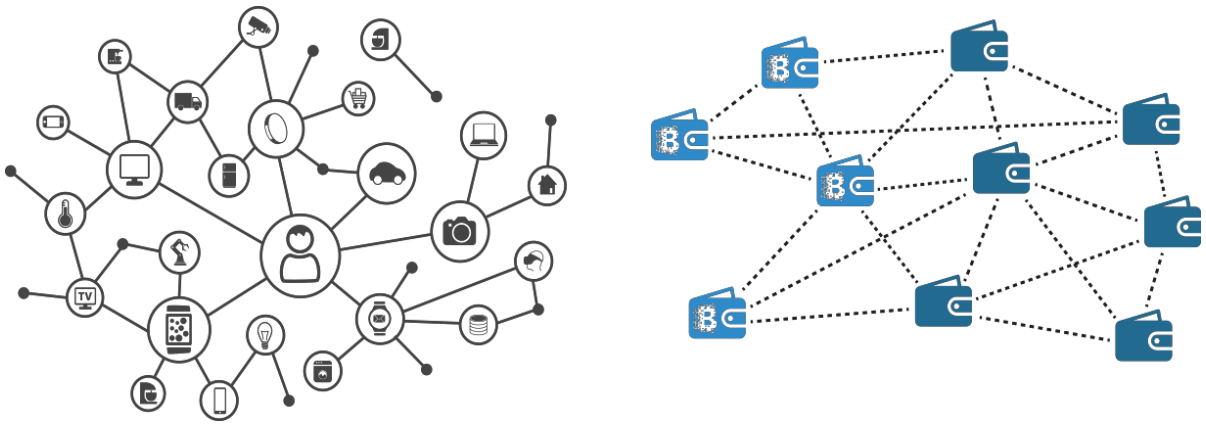

Рисунок 2.3 Порівняння структури мережі IoT і блокчейнів

Інтеграція блокчейнів з IoT спрощує впровадження конфіденційності і цілісності, що є обов'язковим для надійного підключення і безпечної обробки між пристроями. Для цього під'єднані пристрої реагують на фальсифікацію і дії з метою зміни повідомлень, тим самим підвищуючи надійність комунікації. Зокрема, блокчейн може справлятися із зовнішніми атаками завдяки комплексному математичному шифруванню реєстру транзакцій в рамках блоку. Крім того, блокчейн використовує децентралізований метод, який заважає хакерам визначити мету атаки. Ці характеристики зводять до мінімуму вплив одиничних атак на цифрові двійники.

Надійні функції, надані для IoT-пристроїв, перераховані нижче:

- P2P і децентралізована структура розподіляє цілі для атак, ускладнюючи хакерам проводити ідентифікацію індивідуальних користувачів для атак. У випадку з приватним блокчейном, при обмеженнях на розробку розподілених обчислень, проблему безпеки можна вирішити, захистивши мережу за допомогою методу «Безпечного IP», який значно скорочує загрозу зовнішніх атак.
- Можна забезпечити прозорість усіх аспектів транзакцій, записуючи їх в

розподіленому реєстрі, де при необхідності до них можна легко отримати доступ. Це гарантує цілісність відомостей про транзакції для своєчасного реагування на підробку або злом, скорочуючи витрати на вирішення спорів, так як всі учасники підтверджують деталі транзакції.

- Процедури аутентифікації і авторизації потрібні для IoT-пристроїв, які є предметами транзакцій.
- У випадку з публічним блокчейном, можна збільшити ефективність його створення і підтримки відповідно до розподілу. Крім того, широкомасштабна децентралізація публічного реєстру покращує ефективність, скорочуючи вартість транзакцій, тим самим більш ефективно розміщуючи публічні і приватні ресурси.

В результаті, мережа, що використовує блокчейн, надасть надійне середовище для адміністратора і користувача, щоб обмінюватися різними даними і значеннями [17].

2.11 Конфігурація блокчейн-мережі IoT

Блокчейн-мережа IoT - це приватний блокчейн, зареєстрований після аутентифікації, який може працювати на блокчейн-мережі. Тому можна сказати, що він відрізняється від публічного блокчейна своїм доступом до мережі.

Блокчейн-мережа IoT містить наступні компоненти:

- Вузол блокчейн: записує всі блоки з транзакціями в якості повного вузла. Зберігає інформацію про конфігурацію контролю взаємодії користувача з пристроєм і пристрою з пристроєм, виставлення рахунків і управління, здійснюваного адміністратором.
- Адміністратор: Людина, чи система, яка проводить реєстрацію користувачів, шлюзів і пристроїв в блокчейні, а також надає до них доступ. Налаштування безпечно зберігаються в повному вузлі блокчейна і передаються відповідним користувачам, шлюзам і пристроям через мережу. Кожен користувач і пристрій підтримує налаштування, пов'язані з ними. Вони також можуть систематично інтегруватися з існуючим операційним середовищем IoT.

- Користувач: особа або пристрій, що діє в якості простого вузла, який не зберігає блоки.
- Шлюз: одиниця для контролю фіктивних пристроїв або датчиків. Може аналізувати деталі IoT-контракту і потім передавати фіктивним пристроям або датчикам. Кожен пристрій або датчик з'єднаний з індивідуальною адресою.
- Пристрій: пристрій, підключений до шлюзу або простого вузлу, який не зберігає блоки. Відповідає індивідуальним адресам, також може аналізувати деталі IoT-контракту і керувати ними [16].

Як показаний про на рисунку 2.4, користувач відправляє IoT-контракт, який прикріплений з програмою до шлюзу або пристрою. Пристрій аналізує і керує отриманим IoT-контрактом. Користувач може відправляти транзакції, які забезпечують доступ або контролюють дозволені шлюзи або пристрої.

Рисунок 2.4 Структура блокчейн-мережі IoT

Користувач в блокчейні IoT повинен мати доступ до пристрою відповідно до прав доступу. Користувач також повинен контролювати спеціальне обладнання відповідно до налаштувань або просто дивитися статус обладнання. Звичайним

користувачам може заборонятися доступ до певного обладнання. Це дозволяє адміністратору надавати дозвіл на доступ за адресами користувачів, пристроїв або шлюзів. Таке налаштування права доступу зберігається у всіх повних вузлах блокчейн-мережі, а також розподіляється між усіма вузлами, шлюзами і пристроями.

Доступ і контроль користувачів і пристроїв, а також платіжні повноваження безпечно записуються на блокчейні. IoT-контракт може виконуватися після верифікації цього повноваження в порівнянні з записом здійснення транзакції. У звичайному операційному середовищу цифрових двійників такі дозволи видаються часто. Тому блокчейн IoT може бути використаний в поєднанні з існуючим IoT-середовищем.

Представлені наступні типи зіставлення повноважень:

- Зіставлення користувача з пристроєм / шлюзом
- Зіставлення користувача з користувачем
- Зіставлення пристрою / шлюзу з пристроєм

Існують наступні права зіставлення:

Права доступу: право доступу до обладнання. Може визначати мінімальний рейтинг доступу. Користувач або пристрій матимуть рейтинг, що означатиме доступ тільки для спеціального рейтингу. Якщо доступ неможливий, всі права, зазначені нижче, будуть недоступні.

Право перегляду: право на перегляд поточного стану, докладні повноваження можуть визначатися як окремий рядок, і інтерпретується пристроєм для визначення можливості його застосування.

Право контролю / запису: право контролювати пристрій або змінювати його стан. Детальні повноваження можуть визначатися окремо, і інтерпретуватися пристроєм для визначення можливості його застосування.

Права на платіж: визначаються права щодо сум, які підлягають оплаті і автоматичних платежів. Метод обмеження максимальної суми платежу і максимальної одноразової суми платежу протягом певного періоду є ефективним.

Інші права: так як цей метод залежить від пристрою, тут не вказані жодні заходи контролю пристроїв.

Усі транзакції, що використовують цю мережу блокчейн, передаються відповідно до прав доступу. Тобто, якщо користувач А не має доступ до пристрою В, але транзакція проходить від А до В, пристрій В і всі вузли блокчейн відхилять цю транзакцію. У цьому випадку про помилку повідомляється в вузол виявлення вторгнень, і адміністратор може відразу ж перевірити подробиці.

Після того, як адміністратор проведе конфігурацію повноважень між першим користувачем з пристроєм, або пристроєм з пристроєм, права кожного користувача і пристрою можуть коригуватися, якщо внесені зміни в мережу блокчейн. Крім того, при додаванні або видаленні пристрою, необхідно провести відповідне зіставлення. Основні права можуть визначатися в разі, коли додається новий пристрій відповідно до налаштувань.

Процес зіставлення прав доступу в залежності від пристрою в деяких випадках може бути досить складним. Тому можна спробувати згрупувати користувачів, шлюзи і пристрої для ефективного управління зіставленням прав, а також надати програмний інтерфейс [API] або команду у вигляді скрипта для контролю комплексних зіставлень. У деяких випадках зіставлення користувача з пристроєм може бути виражено в більш складній формі, з урахуванням місця розташування, часу і простору.

Даний метод зіставлення користувача з пристроєм вже використовується в IoT- індустрії, його можна буде використовувати в існуючій системі управління для досягнення максимального ефекту, зменшуючи модифікації через відповідне підключення до блокчейну [17].

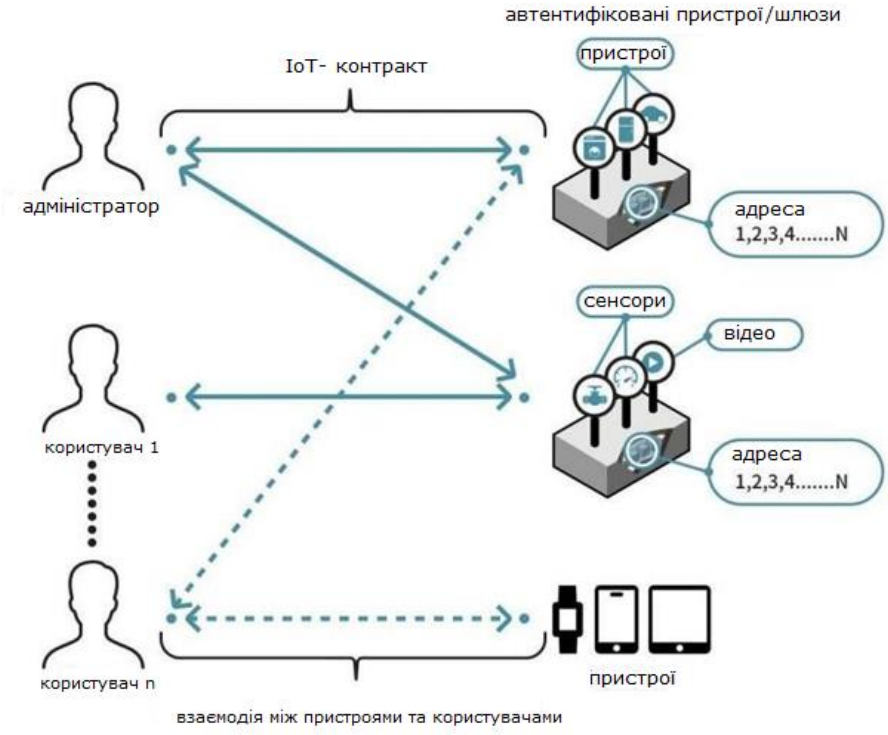

Рисунок 2.5 Зіставлення користувача і пристрою на приватному блокчейні

2.12 IoT-контракт

IoT-контракт - концепція, в якій об'єкт смарт-контракту поширюється на цифрові двійники. Програміст може створити програму, яка контролює функціонування пристрою. Тобто він створює смарт- контракт для IoT і відправляє IoT-контракт на певний пристрій для виконання автоматизованої роботи, платежів і взаєморозрахунків. IoT- контракт - це транзакція, яка передає команди управління між «Користувачем-пристроєм» або «пристроєм-пристроєм», і щоб використовувати цю транзакцію, спочатку необхідно провести аутентифікацію користувача з пристроєм або пристрої з пристроєм, як це буде описано нижче.

Перед використанням цієї транзакції, користувач і пристрій повинні бути спочатку зареєстровані в мережі блокчейн через процедуру аутентифікації. Якщо мова йде про користувача, тільки авторизований користувач, який пройшов двухфакторну аутентифікацію, може отримати доступ до мережі блокчейн. В якості методу аутентифікації користувача, можна додати ID, пароль, одноразовий пароль

або біометричну аутентифікацію (відбитки пальців, райдужна оболонка ока, розпізнавання особи і т.д.) для проведення верифікації.

Так як пристрій складно зареєструвати сам по собі, адміністратор повинен спочатку ідентифікувати і зареєструвати свій пристрій. За допомогою декількох методів можна визначити унікальний ID пристрою. По-перше, не виникне ніяких проблем, якщо пристрій буде мати унікальний ID (такий як, як у чіпа безпеки). Однак якщо все йде не так, можна зареєструвати хеш унікальною відповідної інформації, MAC-адресу, ID центрального процесора, ID диска, OS-зображення, адреса електронного гаманця і т.д., щоб пристрій міг автоматично відключитися від блокчейн-мережі, а адміністратор міг би отримати інформацію про порушення.

Залежно від IoT-контракту, коли статус діючої програми на пристрої змінюється, він може керувати пристроєм, здійснювати автоматичні платежі між пристроями або передавати інформацію про статус або дані на заздалегідь задане місце розташування. У цей час відбувається автоматичний платіж між пристроями (Машинна Валюта), платіж може бути здійснений тільки на адреси, дозволені для передачі на зіставленні користувача з пристроєм і пристрою з пристроєм. Пристрій А, який отримує команди, може передавати статус або контролювати інформацію, що передається іншому пристрою В відповідно до змісту програми, а також може контролювати пристрій В.

Контроль можливий тільки після реєстрації установки і отриманні прав контролю в блокчейн-мережі. Надаючи обслуговування IoT-контракту через Hdac, можливий контроль стану, платежі і управління між користувачами і пристроями, а також між одними пристроями. Завдяки цьому можна здійснювати міжмашинні платежі.

Автоматизовану програму можна додати до IoT-контракту, щоб користувач міг контролювати пристрій, отримувати інформацію про статус пристрою, автоматично здійснювати платежі, якщо виконані певні умови і т.д. Ця програма може передавати прості дані в форматі JSON відповідно до умов пристрою, вона може мати програмований тип API, керуючий більш складною інформацією.

У пристроях з високими характеристиками, до IoT-контракту додається більш

складне програмування на вищому рівні і передається пристрою, воно може інтерпретуватися і оброблятися через віртуальну машину, що функціонує в рамках пристрою. Найефективніший і зручний спосіб контролю IoT-пристроїв щодо швидкості та безпеки - це використання API для розробки програм щодо пристрою.

Що стосується гнучкості, інтерпретуюча програма або віртуальна машина можуть бути ефективними. При використанні цих методів немає необхідності змінювати пристрій за допомогою зміни контрольної програми з боку користувача, тому їх легко і зручно застосовувати. Однак для їх використання потрібно відносно високий рівень продуктивності комп'ютера і пам'яті, тому вони не можуть завантажуватися на пристрої з низькими характеристиками. Наприклад, є 100 пристроїв класу А, і політика управління цими пристроями змінилася, то буде ефективніше і безпечніше змінити з боку користувача IoT-контракт, у якого є права контролю, ніж змінювати або оновлювати 100 пристроїв.

З огляду на те, що транзакція повинна здійснюватися в режимі реального часу, в IoT відповідну кількість пристроїв зареєстровано в блокчейні. У разі, коли мережа забезпечує необхідну продуктивність обробки, може оброблятися близько 500 транзакцій / в секунду. Кількість транзакцій або час реагування, яким можна керувати, може сильно залежати від продуктивності мережі і пристроїв.

Зокрема, якщо транзакція вимагає високого рівня безпеки, вона може передаватися на пристрій через безпечний канал окремо від звичайної мережі, такий як VPN (віртуальна приватна мережа). Іншими словами, вона може бути недоступною через проміжну публічну мережу. У приватному блокчейні пристрій і повний вузол можуть бути організовані в співвідношенні N: 1, де пристрої безпеки використовуються для поліпшення безпеки сегмента мережі. При необхідності для користувача програма і блок даних можуть бути зашифровані і передані завдяки безпеки пристрою [17].

Висновки до розділу 2

В другому розділі отримано наступні результати:

- Досліджено основні ризики, яким може піддаватися мережа.
- Сплановано стратегію забезпечення безпеки, починаючи з розробки, закінчуючи інтерацією всієї системи
- Проаналізовано інтеграцію приватного блокчейна з технологією IoT.
- Досліджено переваги використання IoT контракту.

Безпека в IoT унікальна, тому що луди повинні довіряти технології. Унікальність полягає в тому, що повинна відбуватися ідентифікація, аутентифікація, зберігання та обробка інформації, в тому числі і фінансової. Також повинна забезпечуватись безпека доставки товарів, мультифакторна аутентифікація, автоматична фіксація передачі прав від одного контрагента до другого.

Всього п’ять кроків дозволять зробити дану технологію біль безпечною:

1) Оцінка цілей і ризиків.
2) Управління цілісністю системи.
3) Шифрування конфіденційних даних.
4) Для важливих систем використовувати апаратне забезпечення і стандарти.
5) Захист пристроїв з обмеженими можливостями за допомогою оверлейної мережі.

3. МЕТОДОЛОГІЯ РОЗРАХУНКУ ПРОДУКТИВНОСТІ МЕРЕЖІ

3.1 AWS IoT

AWS IoT забезпечує безпечне двостороннє спілкування між пристроями, підключеними до Інтернету, такими як датчики, виконавчі пристрої, вбудовані мікроконтролери або інтелектуальні пристрої та Cloud AWS. Це дає змогу збирати телеметричні дані з кількох пристроїв, а також зберігати та аналізувати дані.

AWS IoT складається з наступних компонентів:

- *Шлюз пристрою.* Дозволяє пристроям безпечно та ефективно спілкуватися з AWS IoT.
- *Брокер повідомлень.* Забезпечує безпечний механізм для пристроїв і програм IWT AWS для публікації та отримання повідомлень одне від одного. Для публікації та підписки може використовуватись протокол MQTT або MQTT через WebSocket, чи інтерфейс HTTP REST.
- *Двигун правил.* Забезпечує обробку повідомлень та інтеграцію з іншими службами AWS. Може використовуватись мова на базі SQL, щоб вибрати дані з корисних інформаційних повідомлень, а потім обробляти та надсилати дані іншим службам, таким як Amazon S3, Amazon DynamoDB та AWS Lambda. Також може використовуватись брокер повідомлень, щоб повторно опублікувати повідомлення іншим абонентам.
- *Служба безпеки та ідентифікації.* Забезпечує спільну відповідальність за безпеку. Пристрої повинні зберігати свої облікові дані в безпеці, щоб надсилати їх до брокера повідомлень. Брокер повідомлень та двигун правил використовують функції безпеки AWS для надійного надсилання даних на пристрої або інші служби AWS.
- *Реєстр.* Організує ресурси, пов'язані з кожним пристроєм у хмарі. При реєстрації пристрою відбувається прив’язка до трьох спеціальних атрибутів кожного з них. Можлива прив’язка сертифікатів та клієнтських ідентифікаторів, щоб покращити здатність попереджати та усувати неполадки.

- *Реєстр групи.* Дозволяє керувати кількома пристроями одночасно, поділивши їх на групи. Групи також можуть містити підгрупи, що створює певну ієрархію. Будь-які дії, які виконуються на батьківській групі, застосовуватимуться до її дочірніх груп, а також до всіх пристроїв у ньому та у всіх його дочірніх групах. Дозволи, надані групі, будуть застосовуватися до всіх пристроїв та у всіх його дочірніх групах.

- *Цифровий двійник.* Документ JSON, який використовується для зберігання та отримання поточної інформації про стан для пристрою. Забезпечує стійкі уявлення про пристрої в хмарі. Оновлену інформацію про стан пристрою можна опубліковувати в службі, а пристрій синхронізує цей стан при підключенні. Пристрої також можуть опублікувати свій поточний стан у тінь для використання додатками або іншими пристроями.

- *Служба автентифікації за замовчуванням.* Ви можете визначити власні авторизатори, які дозволяють керувати вашою власною стратегією автентифікації та авторизації за допомогою спеціальної служби автентифікації та функції лямбда. Користувальницькі авторизатори дозволяють AWS IoT перевіряти автентичність пристроїв та дозволяти операції за допомогою автентифікації та стратегій авторизації токенів носіїв. Користувальницькі авторизатори можуть реалізовувати різні стратегії автентифікації (наприклад, перевірка JWT, виклик провайдера OAuth тощо) і повертати документи політики, які використовуються шлюзом пристрою для авторизації операцій MQTT.

- *Jobs service.* Дозволяє визначити набір віддалених операцій, які надсилаються і виконуються на одному або декількох пристроях, підключених до AWS IoT. Наприклад, ви можете визначити роботу, яка наказує пристроям завантажити та встановити оновлення програм, перезавантаження, обертання сертифікатів або виконання віддалених способів усунення несправностей. Для виконання цих дій необхідно вказати опис віддалених операцій, які потрібно виконати, і список цілей, які повинні виконувати їх. Цілі можуть бути окремими пристроями або групами [18].

3.1.1 Принцип роботи AWS IoT

AWS IoT дозволяє підключеним до Інтернету пристроям підключитися до Cloud AWS і дозволяє додаткам у хмарі взаємодіяти з пристроями, підключеними до Інтернету. Звичайні додатки IoT збирають та обробляють телеметрію з пристроїв або дозволяють користувачам дистанційно керувати пристроєм.

Пристрої звітують про стан, публікуючи повідомлення в форматі JSON на теми MQTT. Кожна тема MQTT має ієрархічне ім'я, яке ідентифікує пристрій, стан якого оновлюється. Коли повідомлення публікується в темі MQTT, повідомлення надсилається брокеру повідомлень AWS IoT MQTT, який відповідає за надсилання всіх повідомлень, опублікованих у MQTT-темі, для всіх клієнтів, які підписалися на цю тему.

Зв'язок між пристроєм та AWS IoT захищено за допомогою сертифікатів X.509. Сертифікат може бути створено автоматично, або завантажено користувачем. У будь-якому випадку сертифікат повинен бути зареєстрований та активований за допомогою AWS IoT, а потім скопійовано на ваш пристрій. Коли ваш пристрій спілкується з AWS IoT, він подає сертифікат AWS IoT як посвідчення.

Користувачем створюються правила, які визначають одну чи кілька дій для виконання на основі даних у повідомленні. Наприклад, можна, оновлювати або запитувати дані з таблиці DynamoDB або запустити певну функцію. Правила використовують вирази для фільтрування повідомлень. Коли правило збігається з повідомленням, двигун правил викликає дію, використовуючи вибрані властивості. Правила також містять роль IAM, яка надає AWS IoT дозвіл на ресурси AWS, які використовуються для виконання дії. Архітектура AWS IoT зображена на рисунку 3.1

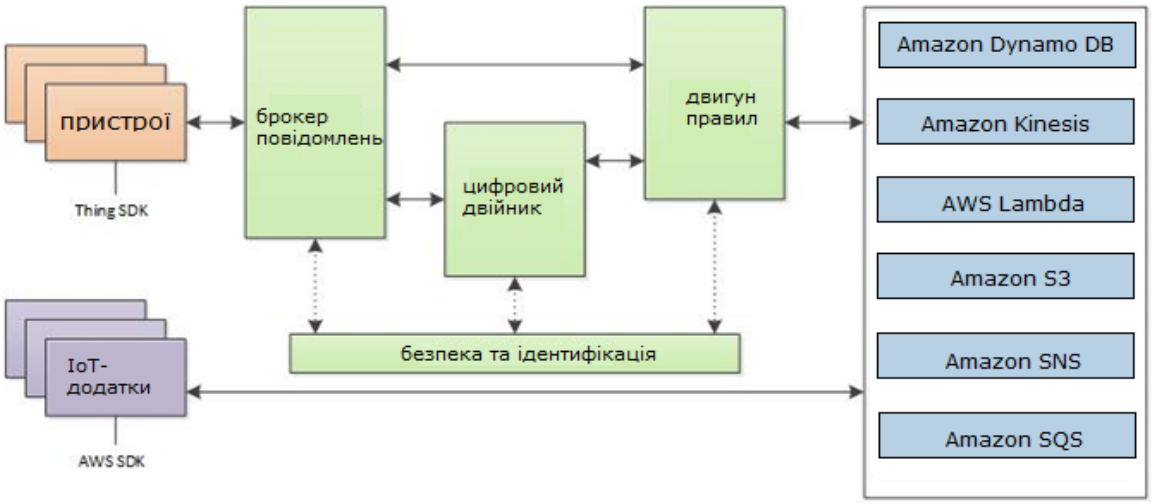

Рисунок 3.1 Архітектура AWS IoT

Кожен пристрій має цифрового двійника, який зберігає та витягує інформацію про стан. Кожен елемент у інформації про стан має дві записи: стан, останній з якого повідомляється пристроєм, і бажаний стан запитується програмою. Програма може запитати поточну інформацію про стан для пристрою. Двійник відповідає на запит, надаючи документ JSON з інформацією про стан (як повідомляється, так і за бажанням), метадані та номер версії. Програма може керувати пристроєм, надіславши запит на зміну свого стану. Цифровий двійник приймає запит на зміну стану, оновлює інформацію про стан та надсилає повідомлення про те, що інформація про стан була оновлена. Пристрій отримує повідомлення, змінює стан і повідомляє про його новий стан [18].

3.1.2 Принцип роботи сервісів

Для організації роботи сервісу, перш за все потрібен акаунт AWS. Як тільки AWS IoT Button буде підключена до мережі Wi-Fi і отримає виділений для неї сертифікат і приватний ключ, буде виконано її безпечне підключення до AWS IoT Core. При натисканні кнопка буде публікувати повідомлення в темі. За допомогою сервісу правил AWS IoT можна налаштовувати маршрутизацію подій одиничним,

подвійним і тривалим натисканням кнопки до будь-якого сервісу AWS. Крім того, можна налаштувати відправку повідомлень самому собі через Amazon SNS або зберігання даних про натисканнях в таблиці Amazon DynamoDB. Можна навіть запрограмувати власну логіку за допомогою функції AWS Lambda, написаної на Node.js, Python або Java, а потім налаштувати функцію на підключення до сторонніх сервісів або інших пристроїв, підключеним до IoT.

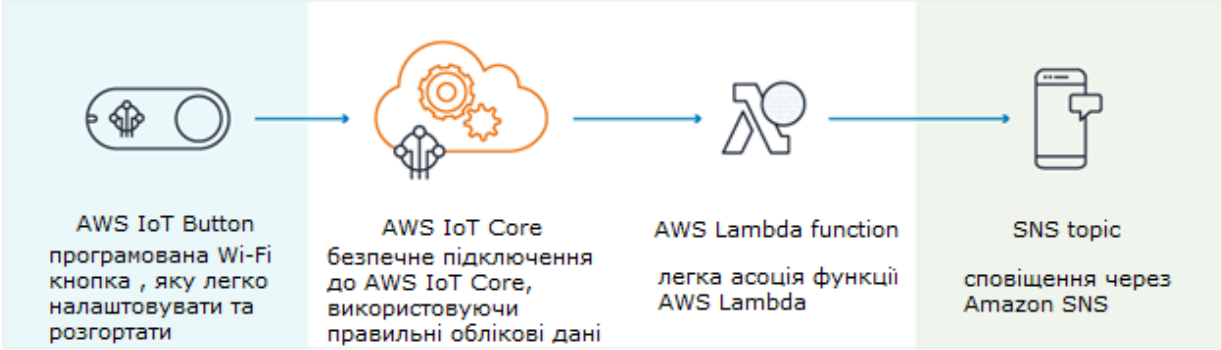

Рисунок 3.2 Основні блоки AWS IoT

AWS IoT безпосередньо інтегрується з наступними службами AWS:

• *Amazon Simple Storage Service* - забезпечує масштабоване зберігання в Cloud AWS. Сучасні компанії повинні вміти просто та безпечно збирати, зберігати та аналізувати дані у величезному масштабі. Amazon S3 - це об'єкт зберігання, призначений для зберігання та вилучення будь-яких обсягів даних з будь-яких джерел: веб-сайтів і мобільних додатків, корпоративних додатків, а також даних з датчиків або пристроїв IoT. Сервіс гарантує надійність зберігання на рівні 99,999999999% і використовується для зберігання мільйонів прикладних програм, що застосовуються у всіх галузях промисловості. S3 відкриває найширші можливості для забезпечення безпеки та відповідає самим строгим нормативним вимогам. Сервіс надає клієнтам гнучкість у керуванні даними, забезпечуючи оптимізацію витрат, контроль доступу та відповідність вимогам. S3 надає функціональні можливості для виконання запитів до сервісу без вилучення, що дозволяє використовувати потужні аналітичні інструменти для безпосередньої обробки даних, що зберігаються в S3.

• *Amazon DynamoDB* - надає керовану базу даних NoSQL. це швидкий і

гнучкий сервіс баз даних NoSQL. Він підходить для будь-яких додатків, що вимагають стабільної роботи із затримкою не більше кількох мілісекунд на будь-який масштаб. Ця повністю керована обласна база даних підтримує роботу на базі як документів, так і пар «ключ-значення». Гнучка модель даних, стабільна продуктивність та автоматичне масштабування пропускної здатності роблять цей сервіс відмінною платформою для мобільних і інтернет-додатків, ігор, реклами, IoT та багатьох інших додатків. Amazon DynamoDB Accelerator (DAX) - це повністю керований високоактивний кеш пам'яті, який дозволяє зменшити час відгуку DynamoDB з мілісекунд до мікросекцій навіть при обробці мільйонів запитів у секунду.

• *Amazon Kinesis*-Дозволяє в реальному часі обробляти потокові дані в масовому масштабі. За допомогою Amazon Kinesis можна просто збирати, обробляти і аналізувати потокові дані в режимі реального часу, щоб своєчасно отримувати аналітичні результати і швидко реагувати на нову інформацію. Надані службою Amazon Kinesis основні можливості дозволяють економічно обробляти поточні дані на будь-якому масштабі, а також забезпечують гнучкість при виборі інструментів, які оптимально відповідають вимогам програми. Amazon Kinesis дає можливість в режимі реального часу збирати такі дані, як відео- та аудіопотоки, додатки журналів, події навігації користувачів по веб-сайтам і телеметричні дані Інтернету для машинного навчання, аналізу та інших додатків. Amazon Kinesis дозволяє обробляти і аналізувати дані за моментом поступу і реагувати миттєво, а не чекати, поки всі дані будуть зібрані, щоб розпочати їх обробку.

• *AWS Lambda* - запускає ваш код на віртуальних серверах від Amazon EC2 у відповідь на події. AWS Lambda дозволяє запускати програмні коди без виділення серверів та керування ними. За допомогою Lambda можна запускати практично будь-які види програм і серверних сервісів, при цьому виконувати які-небудь операції адміністрації не потрібно. Просто завантажте програмний код, і Lambda забезпечить всі ресурси, необхідні для його виконання та масштабування, з високою ступенем доступності. Можна налаштувати автоматичний запуск програмного коду з інших сервісів AWS або безпосередньо з будь-якого мобільного або веб-додатка.

• *Amazon Simple Notification Service* - це гнучка, повністю керована служба передачі повідомлень по моделі «публікація-підписка» та повідомлення для мобільних пристроїв, що дозволяє координувати доставку повідомлень для підписаних кінцевих точок і клієнтів. За допомогою SNS можна надсилати повідомлення більшій кількості отримувачів, включаючи розподілені системи та сервіси, а також мобільні пристрої. Сервіс дозволяє надійно надсилати повідомлення з усіма кінцевими точками при будь-якому масштабі. Розпочати роботу з SNS можна за кілька хвилин, використовуючи Консоль управління AWS, інтерфейс командного рядка AWS або AWS SDK всього з трьома простими API. SNS дозволяє забути про складності та додаткові витрати, пов'язані з обслуговуванням виділеної ПО та інфраструктури для відправлення повідомлень, а також з керуванням такими.

• *Amazon Simple Queue Service*-зберігає дані в черзі, які потрібно завантажити за допомогою додатків. Служба простої черги Amazon (SQS) - повністю керована послугами повідомлень, що дозволяє легко ізолювати та масштабувати мікросервіси, розподілені системи та безсерверні додатки. Одна з сучасних рекомендацій по розробці ПО - створення архітектури, в якій кожний окремий компонент рішення виконує свою конкретну задачу. Така підхід дозволяє підвищити масштабність та надійність готової системи. SQS дозволяє легко та економічно ізолювати компоненти обласного додатка та координувати їх роботу. За допомогою SQS можна відправляти, зберігати та отримувати повідомлення компонентів програмного забезпечення в будь-якому масштабі без втрати повідомлень і необхідності забезпечити постійну доступність інших сервісів [18].

3.2 Фактори, що впливають на продуктивність Web-служб

Розглядаються фактори, що впливають на продуктивність Web-служб, а також приводиться методика розробки простих кількісних моделей для оптимального вибору протоколу по критерію продуктивності, що залежить від числа користувачів. Web-служба - це послуга, що отримується через Internet, яка виконує поставлені

завдання або виробляє транзакції [29]. Web-служби мають модульну структуру і є самодостатніми модулями, які можна описати, опублікувати і виконати через Internet. Web-служба може являти собою бізнес-процес, додаток і навіть обчислювальний ресурс. До прикладів Web-служб можна зарахувати все, починаючи з послуг електронних платежів і закінчуючи послугами поширення вмісту і зберігання інформації. Web-служби можна автоматично викликати з додатків, а не тільки з браузерів.

Бізнес стає все більш залежним від використання Web-служб, тому їх робочі характеристики (тобто продуктивність) стають життєво важливими. Чим більше інформації і послуг можна отримати від компанії, тим більше запитів він отримує. А чим більше запитів отримує Web-служби, тим більше ймовірність, що користувачі занадто довго чекатимуть реакції на свій запит. І тоді, як правило, Web-користувачі (можливі покупці) будуть розчаровані і будуть відмовлятися від надання послуг.

Час відгуку і пропускна здатність - це два найбільш важливих показника продуктивності Web-систем. Зазвичай під пропускною спроможністю Web-служби розуміють швидкість обслуговування запитів, що вимірюється кількістю операцій в секунду. Через великого розмаїття розмірів запитуваних Web-об'єктів, пропускну здатність також вимірюють в бітах в секунду (біт / с, bps).

Незважаючи на те, що затримка установки зв'язку впливає на сприйняття користувачами сайту, в той же час вона є нетривалим, одноразово діючим фактором продуктивності сайту, тобто він вимірюється один раз на запит і не відображає якість всього сеансу [18].

Отже, до найбільш часто використовуваних показників продуктивності Web-служб відносяться:

- пропускна здатність, запитів / с;
- пропускна здатність, Мбіт / с;
- час відгуку при наскрізний передачі даних;
- навантаження центрального процесору;

3.3 Опис системи та розрахунок пропускної здатності

В офісному центрі розташовані кулери з водою, на які прикріплено кнопку Amazon Dash. Коли закінчується вода, користувач натискає кнопку. Сповіщення надходить на смартфон адміністратора. Адміністратор в свою чергу проходить багатофакторну автентифікацію, вибирає зручну дату і час та підтверджує замовлення. В цифровому двійнику кнопки «зашита» вся необхідна інформація: починаючи від цифрового ідентифікатора, закінчуючи складом та специфікацією необхідної води.

Запит обробляється сервером, та відправляється кур'єру. Коли вода доставлена, менеджер підтверджує оплату і після цього відбувається оплата.
аналіз було проведено на реалізації сервісу IoT, що розгорнутий на інфраструктурі Amazon, який може бути реалізований на транспортному рівні протоколом MQTT чи HTTP.

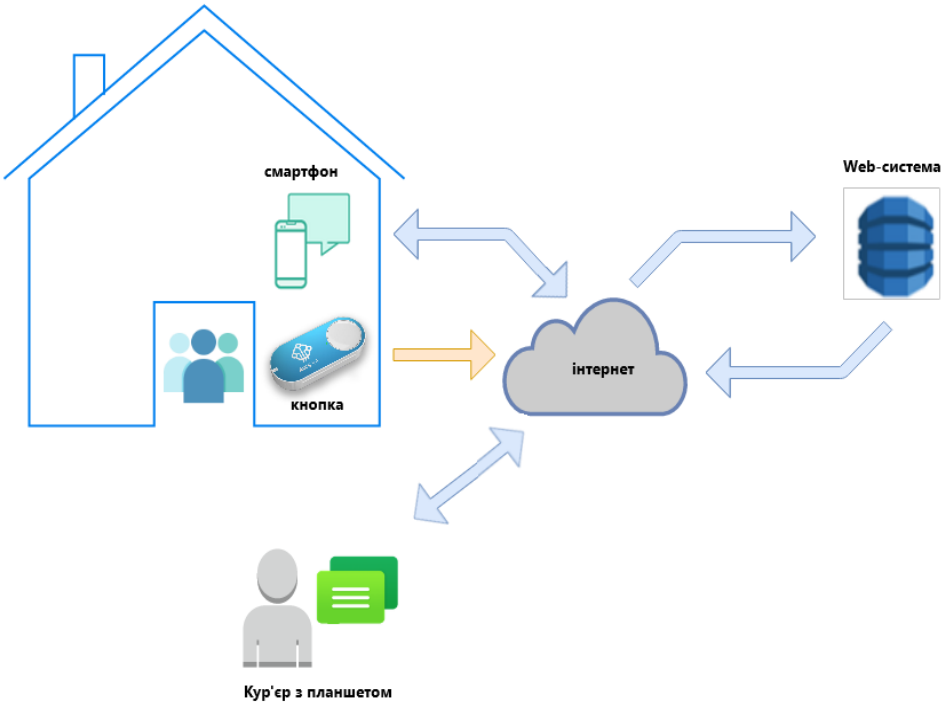

Рисунок 3.3 Схематичне відображення бізнес-процесу

Нижче наведено типи запитів, що приймають участь у повному циклі від

натискання кнопки, до підтвердження замовлення:

1 тип – замовлення води.

2 тип – підтвердження замовлення та вибір зручної дати доставки.

3 тип – багатофакторна автентифікація.

4 тип – підтвердження доставки.

5 тип – оплата.

Кожен з наведених типів може мати різний об’єм даних. Нижче наведено об’єм інформації простого цифрового двійника та детального.

а) Простий варіант (10 В):

1) ідентифікаційний номер - 0225536658

б) Детальний (2 kB):

1) ідентифікаційний номер – 0548946844

2) назва – «Сніжинка»

3) тип води - столова (із вмістом солей до 1 г/дм3), має низькій рівень мінералізації, можна вживати без обмежень.

4) склад води

5) дата розливу – 05.03.2018 р. 05:55:20

6) адреса – м. Київ, вул. Комарова, буд.15

7) ємність - 18,9 л. (19 л.)

8) державні санітарні норми - ДСан ПІН 2.2.4-171-10. Каламутність НОФ - 0, смак та присмак (бал) - 0, запах при І 20°С-60°С (бал) - 0, водневий показник од Рн 7,3, нітрати не більше 0,1 (мг/дмЗ), загальна жорсткість - 0,9(ммоль/дмЗ), амоній не більше 0,1 мг/дмЗ.

9) вимоги до бутля - кожен бутель питної води фасованої повинен мати етикетку, на якій повинно бути зазначено: назва води питної, (природна), негазована; дата виготовлення; строк придатності до споживання чи дата закінчення строку придатності до споживання; умови зберігання; показники якості; найменування, місцезнаходження виробника і місце виготовлення питної води; назва нормативного документа, який визначає вимоги щодо якості питної води.

В таблицях 3.1 та 3.2 наведено розміри повідомлень для різних типів запитів.

Таблиця 3.1 – Варіації розміру повідомлень при використанні HTTP

	Розміри повідомлень, Біт									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	1290	2290	6290	8290	9290	11290	16290	20290	36290	40290
2	5290	7290	12290	15290	18290	20290	25290	30290	50290	77290
3	10290	12290	16290	17290	19290	22290	24290	26290	30290	33290
4	3290	6290	10290	12290	14290	17290	20290	28290	33290	100290
5	100290	110290	120290	140290	165290	170290	200290	250290	260290	270290

Таблиця 3.2 – Варіації розміру повідомлень при використанні MQTT

	Розміри повідомлень, Біт									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	1002	2002	6002	8002	9002	11002	16002	20002	36002	40002
2	5002	7002	12002	15002	18002	20002	25002	30002	50002	77002
3	10002	12002	16002	17002	19002	22002	24002	26002	30002	33002
4	3002	6002	10002	12002	14002	17002	20002	28002	33002	100002
5	100002	110002	120002	140002	165002	170002	200002	250002	260002	270002

Мережевий трафік має пульсуючий характер [20]. "Пульсуючий" відноситься до того факту, що дані передаються випадковим чином, з піковими швидкостями, що перевищують середні швидкості в 8-10 разів [21]. Також вважається, що мережевий трафік пульсує в декількох масштабах часу. Припустимо, що фірма обслуговує 1000000 клієнтів. Кожен клієнт замовляє воду 1 раз на тиждень. Навантаження має піковий характер. Замовлення відбуваються з 06:00 до 09:00 та з

18:00 до 22:00. З цього витікає, що кожного дня тривалість замовлень складає 7 годин.

Пропускна здатність сервера, виражена в запитах/с, складатиме:

$$\frac{1000000}{7*7*3600} = 5{,}66\frac{запитів}{с}, \quad (3.1)$$

Однак, цей показник не дає ніякого представлення щодо пропускної здатності мережі, що використовується в період спостереження і не дає представлення про розміри типів запитів. Таким чином, для оцінки пропускної здатності мережі чи мережевого адаптера потрібно розрахувати пропускну здатність в Біт/с.

На рисунку 3.4 наведено ймовірності того, що кожен з типів запитів відбудеться.

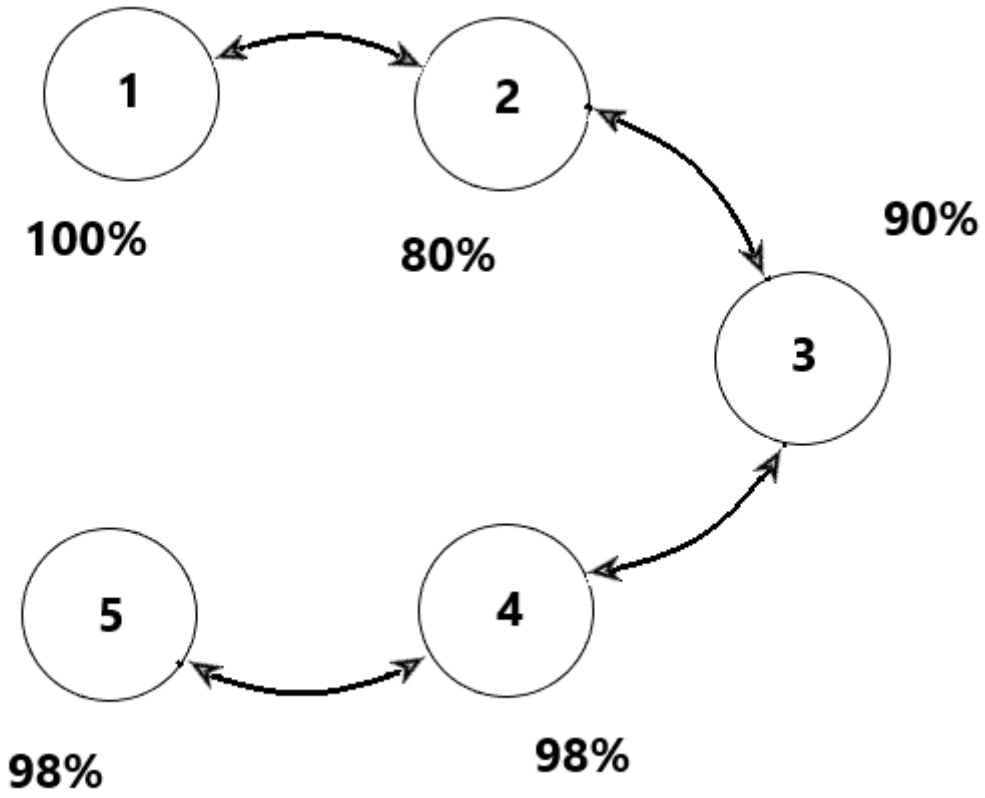

Рисунок 3.4 Ймовірності виконання запитів

Щоб визначити, яку частку складає кожен з типів в усій передачі, проведемо наступні розрахунки:

$$\frac{1}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)} = 0.255, \quad (3.2)$$

$$\frac{0.8}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)} = 0.204, \quad (3.3)$$

$$\frac{(0.8*0.9)}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)} = 0.184, \quad (3.4)$$

$$\frac{(0.8*0.9*0.98)}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)} = 0.180, \quad (3.5)$$

$$\frac{(0.8*0.9*0.98*0.98)}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)} = 0.177, \quad (3.6)$$

Розрахуємо пропускну здатність для кожного типу запиту:

$$Пропускна\ здатність = \frac{загальна\ кількість\ запитів*частка\ кожного\ з\ типів*середній\ розмір}{час\ спостереження}, \quad (3.7)$$

Розрахунки для всіх типів першого варіанту:

$$\frac{1000000*0,255*(1290*8)}{176400} = 14918,3\ [Біт/□] \quad (3.8)$$

$$\frac{1000000*0,204*(5290*8)}{176400} = 48941,5\ [Біт/с] \quad (3.9)$$

$$\frac{1000000*0,184*(10290*8)}{176400} = 85866,67\ [Біт/с] \quad (3.10)$$

$$\frac{1000000*0,180*(3290*8)}{176400} = 26857,14\ [Біт/с] \quad (3.11)$$

$$\frac{1000000*0,177*(100290*8)}{176400} = 805049,00\ [Біт/с] \quad (3.12)$$

Загальна пропускна здатність – це сума пропускних здатностей для кожного з типів:

$$Загальна\ пропускна\ здатність = 14918{,}37 + 48941{,}5 + 85866{,}67 + 26857{,}14 + 805049 = 981632{,}7\ [Біт/с] \quad (3.13)$$

Отже пропускна здатність для першого варіанту складає 981,633 Кбіт/с ;

Розрахунки для інших варіантів виконані за аналогією в Excel, результати наведені в таблицях 3.3 та 3.4

Таблиця 3.3 – Пропускна здатність при використанні HTTP

	Пропускна здатність, Біт/с HTTP									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	14918,37	26482,99	72741,5	95870,75	107435,4	130564,6	188387,8	234646,3	419680,3	465938,8
2	48941,5	67444,9	113703,4	141458,5	169213,6	187717	233975,5	280234	465268	715063,9
3	85866,67	102556	135934,7	144279,4	160968,7	186002,7	202692,1	219381,4	252760,1	277794,1
4	26857,14	51346,94	84000	100326,5	116653,1	141142,9	165632,7	230938,8	271755,1	818693,9
5	805049	885321,1	965593,2	1126137	1326818	1366954	1607770	2009131	2089403	2169675
∑	981632,7	1133152	1371973	1608072	1881089	2012381	2398458	2974332	3498867	4447166

Таблиця 3.4 – Пропускна здатність при використанні MQTT

	Пропускна здатність, Біт/с MQTT									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	11587,76	23152,38	69410,88	92540,14	104104,8	127234	185057,1	231315,6	416349,7	462608,2
2	46277,01	64780,41	111038,9	138794	166549,1	185052,5	231311	277569,5	462603,5	712399,5
3	83463,4	100152,7	133531,4	141876,1	158565,4	183599,5	200288,8	216978,1	250356,8	275390,8
4	24506,12	48995,92	81648,98	97975,51	114302	138791,8	163281,6	228587,8	269404,1	816342,9
5	802737,1	883009,3	963281,4	1123826	1324506	1364642	1605458	2006819	2087091	2167363
∑	968571,4	1120091	1358912	1595012	1868027	1999320	2385397	2961270	3485805	4434104

Таблиця 3.5 – Загальна пропускна здатність при використанні MQTT

	Пропускна здатність, Біт/с									
Тип протоколу	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
MQTT	968571,4	1120091	1358912	1595012	1868027	1999320	2385397	2961270	3485805	4434104
HTTP	981632,7	1133152	1371973	1608072	1881089	2012381	2398458	2974332	3498867	4447166
Δ	1,013485	1,011661	1,009611	1,008188	1,006992	1,006533	1,005475	1,004411	1,003747	1,002946

Рисунок 3.5 Відношення загальної пропускної здатності

З отриманих результатів можна зробити висновок, що мережеве з’єднання повинно мати мінімальну пропускну здатність 5 Мбіт/с, рекомендовано 10 Мбіт/с. А при збільшенні розмірів запитів, різниці між протоколами майже немає. Тому при великих файлах доцільніше використовувати HTTP, бо він не потребує додаткових налаштувань при розгортанні. При передачі запитів, та відповідей невеликих розмірів, доцільніше використовувати MQTT.

3.4 Середній час відгуку.

Визначення затримок, та “вузьких місць” відіграє важливу роль для того, щоб робити зміни в програмному забезпеченні, модернізації обладнання або установки, використовувати більш швидкісні ліній передачі даних. Затримки можна поділити на 3 масштабні складові:

1. Процесор. Центральний процесор відповідальний за обробку даних з пристроїв, а також за ініціалізацію зв’язку.

2. Мережа. Мережа породжує затримки під час доставки інформації від клієнта до сервера і назад від сервера до клієнта. Ці затримки є функціями різних компонентів на шляху між клієнтом і сервером, таких як модеми, маршрутизатори, лінії передачі, мости і комутатори.

3. Сервер. Час перебування на сервері, R'_{Server} - це час, що витрачається на виконання запиту. Воно включає в себе час обробки і час очікування на різних компонентах сервера, таких як процесор, диск і мережевий адаптер.

Коли запитуваний документ не перебуває в кеші клієнта, час відгуку R на запит являє собою суму часу перебування запитів на всіх ресурсах:

$$R_{miss} = R'_{CPU} + R' \text{Network} + R' \text{Server} \quad , \tag{3.14}$$

де R'_{CPU} – час обробки процесором.

$R'_{Network-}$ час передачі в мережі.

R'_{Server} - час перебування на сервері.

Розглянемо середній час відгуку для запропонованих протоколів. Припустимо,

що документи не кешуються. Середній час перебування на сервері (R' $_{Server}$) – 3,6 с., а час, затрачений процесором на обробку отриманих даних (R' $_{CPU}$) – 0,3с. Пропускна здатність мережі складає 100 Мбіт/с.

Час передачі в мережі (R' $_{Network}$) дорівнює розміру даних в бітах, поділеному на пропускну здатність мережі. Продемонструємо розрахунки для першого варіанта. Розрахунки для інших варіантів було проведено в програмі Excel, результати наведено в таблиці 3.6

$$R'_{\text{Network}(HTTP)} = \frac{120450*8}{100000000} = 0{,}009639, [\text{c}] \quad (3.15)$$

$$R'_{\text{Network}(MQTT)} = \frac{119010*8}{100000000} = 0{,}009521, [\text{c}] \quad (3.15)$$

$$R_{miss(HTTP)} = 0.009639 + 0.3 + 3.6 = 3{,}909636, [\text{c}] \quad (3.16)$$

$$R_{miss(MQTT)} = 0.009521 + 0.3 + 3.6 = 3{,}909521\ [\text{c}] \quad (3.17)$$

$$\Delta = \frac{R_{miss(HTTP)}}{R_{miss(MQTT)}}; \quad (3.18)$$

$$\Delta = \frac{3{,}909636}{3{,}909521} = 1.000029467 \quad (3.19)$$

Таблиця 3.6 – Середній час відгуку

	Варіант									
Протокол	1	2	3	4	5	6	7	8	9	10
HTTP	3,9096	3,9110	3,9132	3,9154	3,9181	3,9193	3,9229	3,9284	3,9328	3,9417
MQTT	3,9095	3,9109	3,9131	3,9153	3,9180	3,9192	3,92280	3,92832	3,9327	3,94160
Δ	1,000029467	1,000029456	1,000029439	1,000029423	1,000029403	1,000029394	1,000029367	1,000029326	1,000029293	1,000029227

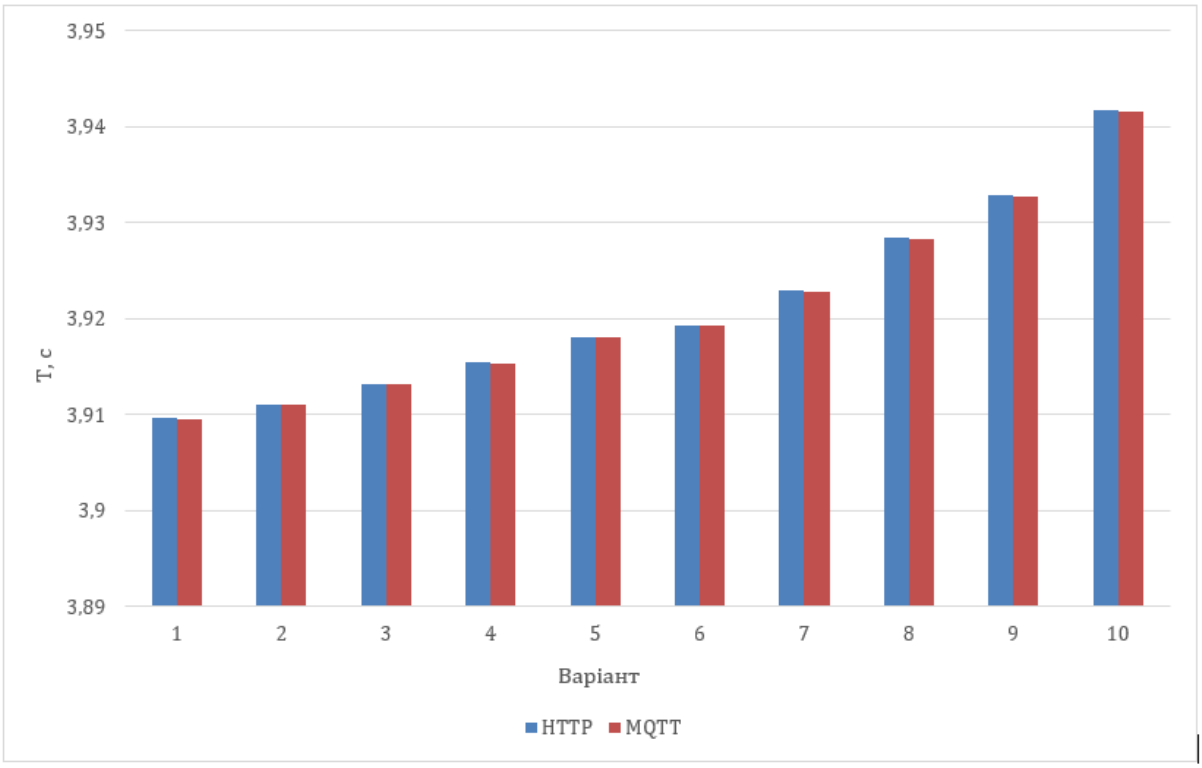

Рисунок 3.6 Порівняння середнього часу відгуку для протоколів

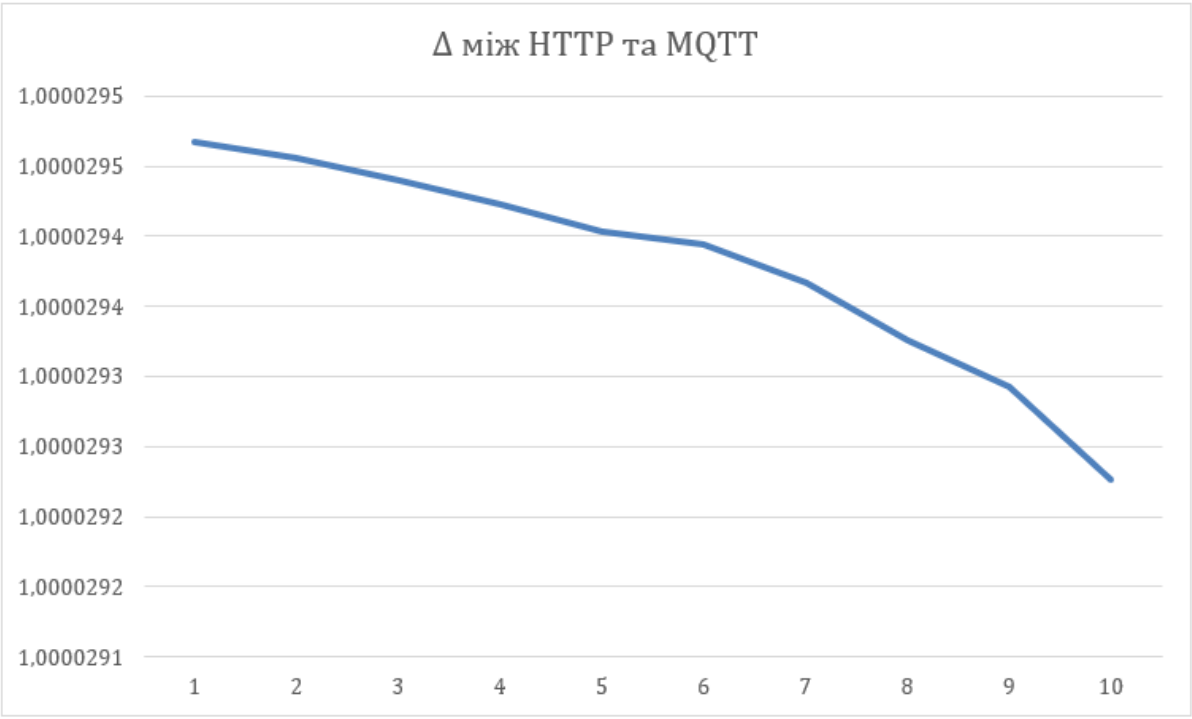

Рисунок 3.7 Відношення середнього часу відгуку

3.5 Пропускна здатність при використанні кешуючого проксі-сервера.

Проксі-сервер, кеш і сервер дзеркального відображення - це три технології, які використовуються для поліпшення характеристик продуктивності і безпеки Web. Згадані технології призначені для скорочення часу доступу до Web-документів, збільшення пропускної здатності мережі, необхідної для передачі Web-документів, потреби в серверах, які містять найбільш популярні документи, а також підвищення безпеки електронних служб [21].

Проксі-сервер являє собою особливий тип Web-сервера. Він має можливість виступати в якості і сервера, і клієнта (див. рис. 3.8). Проксі-сервер працює як агент, що представляє сервер для клієнта і клієнта для сервера. Проксі-сервер приймає запити від клієнтів і направляє їх на Web-сервери. Як тільки проксі-сервер отримує відповіді від віддалених серверів, він передає їх клієнтам. Спочатку проксі-сервери були створені для надання доступу в Web користувачам з приватних мереж, які могли отримувати доступ в Internet тільки через брандмауер.

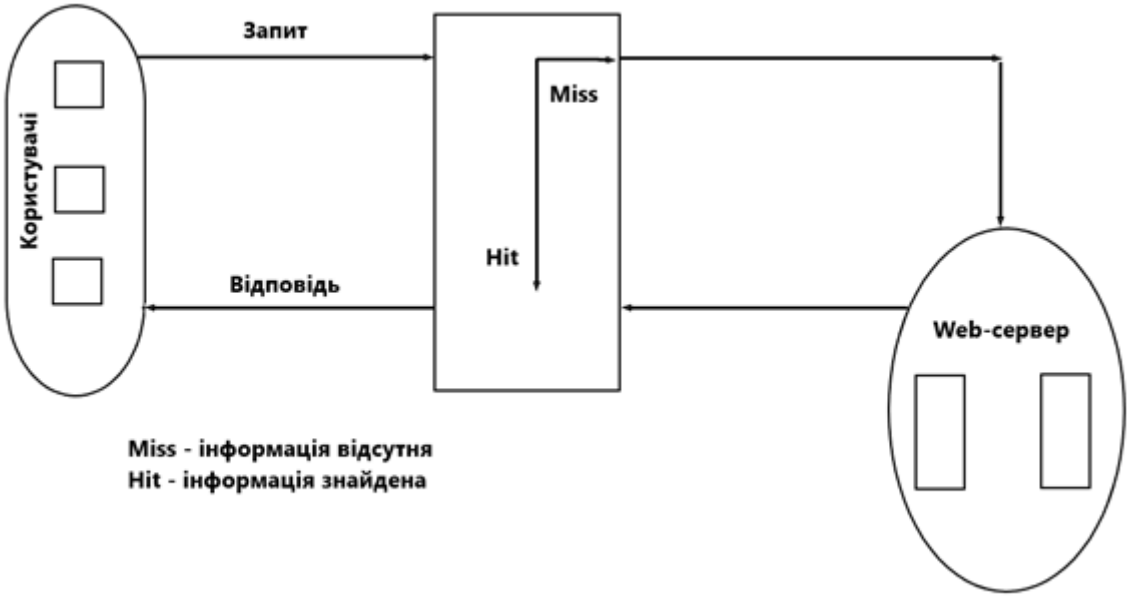

Рисунок 3.8 Архітектура проксі-сервера для Web

Проксі-сервер робить набагато більше, ніж просто комутує відповіді. Проксі-сервери для Web можуть бути налаштовані так, щоб кешувати прийняті відповіді,

стаючи, таким чином, кешуючими проксі-серверами. У великих розподілених інформаційних системах, подібних World Wide Web, кешування - це ключ до високої продуктивності. Основна ідея кешування досить-таки проста: зберегти необхідні документи в локальних файлах або проксі-серверах для подальшого використання. Таким чином, відпадає необхідність у завантаженні документа при подальшому його запиті. Кешування зводить до мінімуму час доступу за рахунок максимально близького розташування даних до їх споживачів. Отже, кешування збільшує швидкість доступу і скорочує мережевий трафік, оскільки часто запитуваний документ повертається з найближчого кешу швидше, ніж з віддалених серверів. Крім того, кешування скорочує навантаження на сервер і збільшує коефіцієнт готовності за рахунок дублювання документа на декількох серверах [24].

Розрахуємо, яке значення пропускної здатності можна економити, використовуючи кешуючий проксі-сервер.

$$Економія\ пропускної\ здатності = Пропускна\ здатність * Результативність * \% кешованого\ трафіку, \quad (3.20)$$

Протокол HTTP передбачає кешування даних, що дозволяє зекономити частину пропускної здатності мережі. В цілях аналізу показника кешування, розглянемо випадок, коли результативність складає 68% і 42% всього трафіку можна зекономити, використовуючи кешування. Пропускна здатність HTTP складає 4,4471 Мбіт/с, а MQTT – 4.434 Мбіт/с.

$$Економія\ ПЗ = 4{,}4471 * 0{,}42 * 0{,}60 = 1{,}12\ [Мбіт/с] \quad (3.21)$$

$$Прорускна\ здатність = 4{,}4471 - 1{,}12 = 3{,}327\ [Мбіт/с] \quad (3.22)$$

Так, як протокол MQTT не використовує кешування, то використання HTTP буде давати виграш в розмірі:

$$4{,}434 - 3{,}327 = 1{,}107\ [Мбіт/с] \quad (3.23)$$

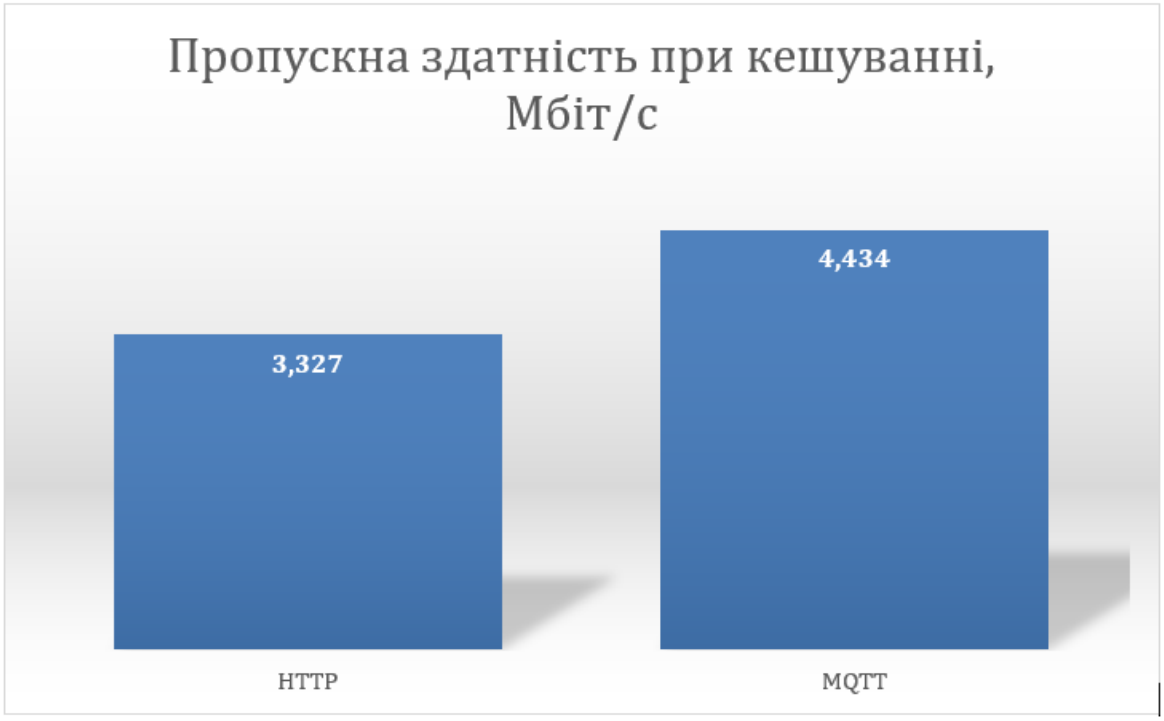

Рисунок 3.9 Порівняння пропускної здатності протоколів при використанні кеш-серверу

3.6 Розрахунок використання ЦП серверу додатків при обробці запитів

Сервер додатків (application server) являє собою програмне забезпечення, обробляє всі прикладні операції між споживачами, і базою даних компанії. У загальному випадку, сервер додатків отримує запит клієнта, виконує деяку бізнес-логіку і взаємодіє з сервером транзакцій і / або сервером бази даних. Як правило, сервери додатків демонструють такі характеристики:

1) зберігають і виконують логіку додатків, написаних на різних мовах програмування (наприклад, Java, C або C ++);

2) керують великими обсягами транзакцій з серверними базами даних;

3) забезпечують сумісність з усіма існуючими мережевими стандартами, включаючи HTTP, HTML, XML, CGI, Netscape Server API (NSAPI), Microsoft

Internet Server API (ISAPI) і Java;

4) працюють з більшістю популярних Web-серверів, браузерів і систем управління базами даних.

Сервери додатків можуть бути реалізовані кількома різними способами: у вигляді CGI-сценаріїв, FastCGI-сценаріїв, Java-сервлетів, серверних додатків і серверних сценаріїв [22].

Припустимо, що Web-сервіс отримує в середньому 20 відвідувань в секунду. 0,8 запитів проходять всі етапи. Кожна з цих транзакцій генерує HTTP – запити (CGI-сценарії), які виконуються на Web-сервері. Потрібно знати навантаження на центральний процесор, що створюється сценаріями. Навантаження, що визивається HTTP – запитом було змодельовано наступним чином. Середня потреба в обслуговуванні ЦП HTTP – запитом (D_{cpu}^{HTTP}) складає 12 мс. Використовуючи закон для необхідності в обслуговуванні, отримуємо:

$$U_{cpu}^{HTTP} = X_{HTTP} * D_{cpu}^{HTTP}\ , \tag{3.24}$$

де X_{HTTP} – пропускна здатність сервера в HTTP - запитах, виконаних за секунду,

U_{cpu}^{HTTP} – використання ЦП, що пов’язане з виконанням HTTP - запитів.

Інтенсивність вхідного потоку (λ_{HTTP}) може бути розрахована, як:

$$\lambda_{HTTP} = 20 * 0.8 = 16\ \left[CGI - \frac{\text{запитів}}{\text{с}}\right], \tag{3.25}$$

Вважаємо, що поток постійний $X_{HTTP} = \lambda_{HTTP}$. Таким чином:

$$U_{cpu}^{HTTP} = 16 * 0.012 = 19\% \tag{3.26}$$

З отриманих результатів ми бачимо, що на обробку HTTP – запит витрачається 19% ЦП.

Розглядаючи MQTT, використовуються Java-сервлети. Сервлет, який підключається до теми MQTT, і пересилає всі повідомлення, які він отримує через WebSocket. Згідно статистики, сервлети на 30% менш ресурсномісткі, ніж CGI-додатки. Потреби в обслуговуванні ЦП зі сторони сервлетів складає:

$$D_{cpu}^{MQTT} = D_{cpu}^{HTTP} * 0.7 = 12 * 0.7 = 8.4 \text{ [мс]}, \quad (3.27)$$

Використовуючи формулу № 3.24, отримаємо:

$$U_{cpu}^{HTTP} = 16 * 0.0084 = 13\%, \quad (3.28)$$

Отримані результати наглядно демонструють, що Java-сервлети (MQTT) знижують використання ЦП з 19% до 13%.

Рисунок 3.10 Використання ЦП серверу, при обробці запитів

3.7 Час обслуговування в різних мережах

Повідомлення від клієнта до сервера повинно пройти через декілька рівнів протоколу та може передаватися через одну або більшу кількість мереж (рис. 3.11.).

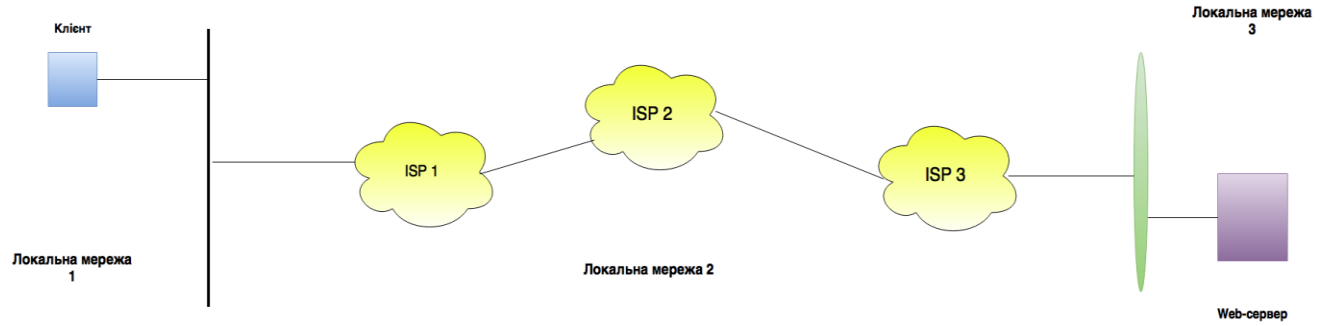

Рисунок 3.11 З’єднання між клієнтом і сервером

Повідомлення від клієнта до Web-сервера повинно пройти через три мережі з швикостями 10 Мбіт/с, 100 Мбіт/с, і 16 Мбіт/с відповідно. Повідомлення, що генеруються додатком, повинні пройти через стек протоколів, що включають, принаймні, транспортний протокол (TCP чи UDP), Internet-протокол (IP) та мережевий протокол. Протокольні об’єкти кожного рівня спілкуються між собою за рахунок обміну протокольними одиницями обміну (ProtocolDataUnit, PDU), що складаються з заголовка та області даних [23].

Протокольні одиниці обміну по різному називаються в різних протоколах і зазвичай відводять максимально можливий розмір під область даних. На мережевому рівні максимальний розмір області даних носить назву максимального розміру блоку корисного навантаження пакету(maximumtransmissionunit, MTU). Для мережі першого ISP (InternetServiceProvider)MTU = 1 500 байт, для мережі другого ISPMTU = 4 472 байт, а для мережі третього ISP MTU = 4 444 байт.

Таким чином, маршрутизатори повинні фрагментувати дейтаграми при переході до мережі з більш низьким значенням MTU. Фрагменти збираються заново на рівні Internet-протокола (IP) на хості місця призначення.

Кожен рівень протоколу добавляє свій власний заголовок, а іноді і хвіст (тобто заключну частину). Час обслуговування повідомлення мережею – це час, необхідний на передачу даного повідомлення через мережу. Згаданий час еквівалентний відношенню кількості байт, необхідних для передачі повідомлення, включаючи заголовок та хвіст(службова інформація) до пропускної здатності

мережі. Розмір заголовка протоколу залежить від протоколів, що використовуються та від фрагментації повідомлення, яка може знадобитися на мережевому рівні.

Для ілюстрації методики розрахунку часу обслуговування повідомлення мережею, розглянемо декілька прикладів.

Приклад перший:

Клієнт на рис. 3.12 посилає 300-байтний запит Web-серверу та отримує 10 000-байтну відповідь. Взаємодія між клієнтом та сервером здійснюється через TCP-з'єднання.

Запит від клієнта до сервера розміщується в області даних TCP-сегмента, який надходить в область даних IP-дейтаграми. IP-дейтаграма інкапсулюється Ethernet-, кадрами по мірі її проходження по мережам. Таким чином, до 300-байтного запиту дописується 20-байтний TCP-заголовок. 20-байтний IP-заголовок плюс 2 байт заголовка MQTT в мережі 1, та 18 байт – в мережі 3. Тобто 300-байтний запит перетворюється в 342-байтовий кадр (= 300 + 20 + 20 + 2) в мережі 1та 2 і в 358-байтний кадр (= 300 + 20 + 20 + 18) в мережі 3. Час на передачу кадру по мережі дорівнює розміру кадру в бітах, поділеному на пропускну спроможність мережі (в біт/с). Відповідно, час передачі для кадрів, які містять клієнтський запит, в мережах 1, 2 та 3, відповідно, складає:

$$\frac{342\times8}{10000000} = 0{,}000273, [\text{с}] \quad (3.29)$$

$$\frac{342\times8}{100000000} = 0{,}00002736, [\text{с}] \quad (3.30)$$

$$\frac{358\times8}{16000000} = 0{,}000179, [\text{с}] \quad (3.31)$$

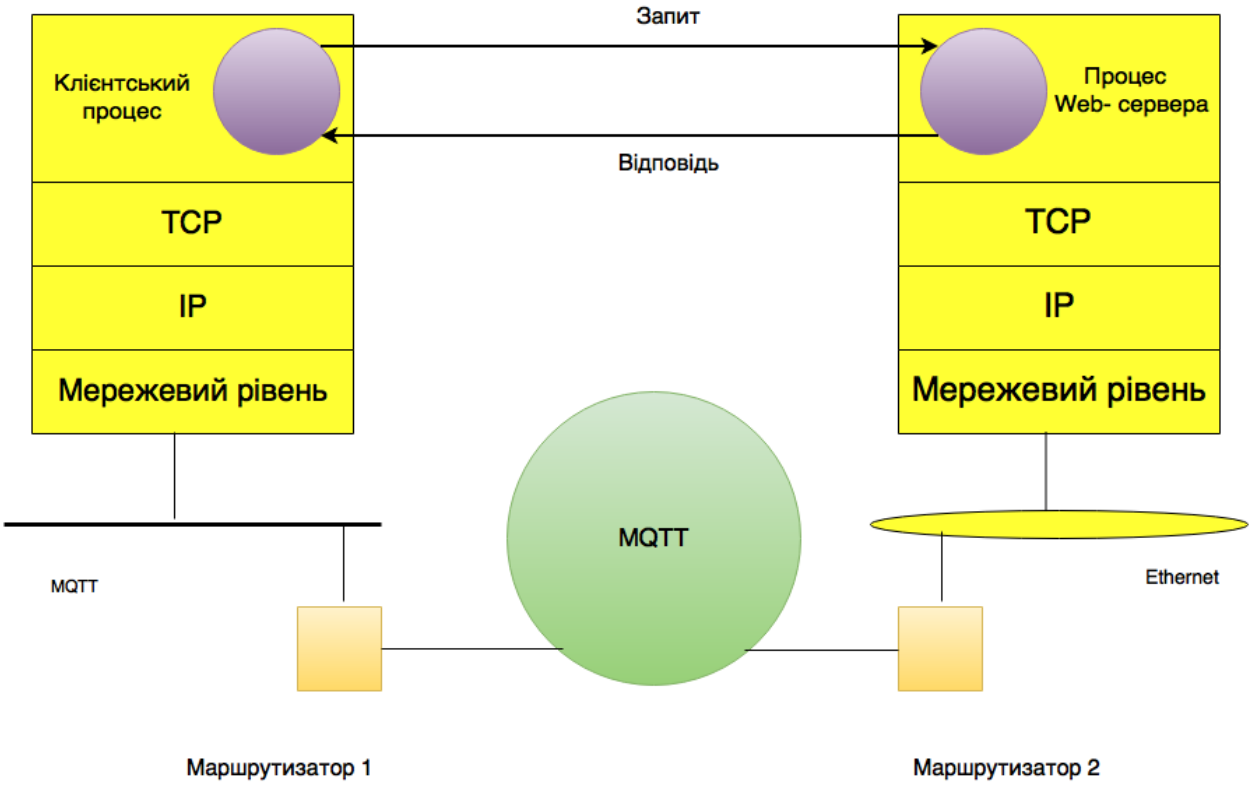

Рисунок 3.12 Взаємодія між клієнтом та сервером

Приклад другий:

Клієнт на рис. 3.11. посилає 10 000-байтний запит Web-серверу.

Таким чином, до 10 000-байтного запиту дописується 20-байтний TCP-заголовок. 20-байтний IP-заголовок плюс 2 байт заголовкаMQTTв мережі 1, та 18 байт – в мережі 3. Тобто 10 000-байтний запит перетворюється в 10042-байтовий кадр (= 10000 + 20 + 20 + 2) в мережі 1та 2 і в 10058-байтний кадр (= 10000 + 20 + 20 + 18) в мережі 3. Відповідно, час передачі для кадрів, які містять клієнтський запит, в мережах 1, 2 та 3, відповідно, складає:

$$\frac{10042\times8}{10000000} = 0{,}008033, [\text{с}] \quad (3.32)$$

$$\frac{10042\times8}{100000000} = 0{,}00080336, [\text{с}] \quad (3.33)$$

$$\frac{10058\times8}{16000000} = 0{,}005029, [\text{с}] \quad (3.34)$$

Приклад третій:

Клієнт на рис. 3.11. посилає 50-байтний запит Web-серверу. Тобто 50-байтний запит перетворюється в 92-байтовий кадр (= 50 + 20 + 20 + 2) в мережі 1та 2 і в 108-байтний кадр (= 50 + 20 + 20 + 18) в мережі 3. Відповідно, час передачі для кадрів, які містять клієнтський запит, в мережах 1, 2 та 3, відповідно, складає:

$$\frac{92\times8}{10000000} = 0{,}0000736, [\text{с}] \quad (3.35)$$

$$\frac{92\times8}{100000000} = 0{,}00000736, [\text{с}] \quad (3.36)$$

$$\frac{108\times8}{16000000} = 0{,}000054, [\text{с}] \quad (3.37)$$

Розглянемо ситуацію, коли замість MQTT використовується HTTP, при відповідних запитах довжиною 300, 10 000, 50 байт. Середня довжина заголовку HTTP складає 290 байт. Таким чином, до 300-байтного запиту дописується 20-байтний TCP-заголовок. 20-байтний IP-заголовок плюс 290 байт заголовка HTTP в мережі 1, та 18 байт – в мережі 3. Тобто 300-байтний запит перетворюється в 630-байтовий кадр (= 300 + 20 + 20 + 290) в мережі 1та 2 і в 648-байтний кадр (= 300 + 20 + 20 + 18+290) в мережі 3.

Час передачі клієнтського запиту складатиме:

$$\frac{630\times8}{10000000} = 0{,}000504, [\text{с}] \quad (3.38)$$

$$\frac{630\times8}{100000000} = 0{,}0000504, [\text{с}] \quad (3.39)$$

$$\frac{648\times8}{16000000} = 0{,}000324, [\text{с}] \quad (3.40)$$

Проведемо аналогічні розрахунки для запиту розміром 10 000 байт та 50 байт. 50-байтний запит перетворюється в 380-байтний в мережі 1 та 2, та в 398-байт в мережі 3.

$$\frac{380\times8}{10000000} = 0{,}000304, [\text{с}] \quad (3.41)$$

$$\frac{380\times8}{100000000} = 0{,}0000304, [\text{с}] \quad (3.42)$$

$$\frac{398\times8}{16000000} = 0{,}000199, [\text{с}] \quad (3.43)$$

Запит розміром 10 000 байт, шляхом додавання заголовків перетвориться в 10330- бітовий в мережах 1 і 2, та в 10348-байтний в мережі 3. Час обслуговування складатиме відповідно:

$$\frac{10330\times8}{10000000} = 0{,}008264, [\text{с}] \quad (3.44)$$

$$\frac{10330\times8}{100000000} = 0{,}0008264, [\text{с}] \quad (3.45)$$

$$\frac{10348\times8}{16000000} = 0{,}005174, [\text{с}] \quad (3.46)$$

На рис. 3.13 зображено порівняльну характеристику часу обробки повідомлення запиту розміром 300 байт.

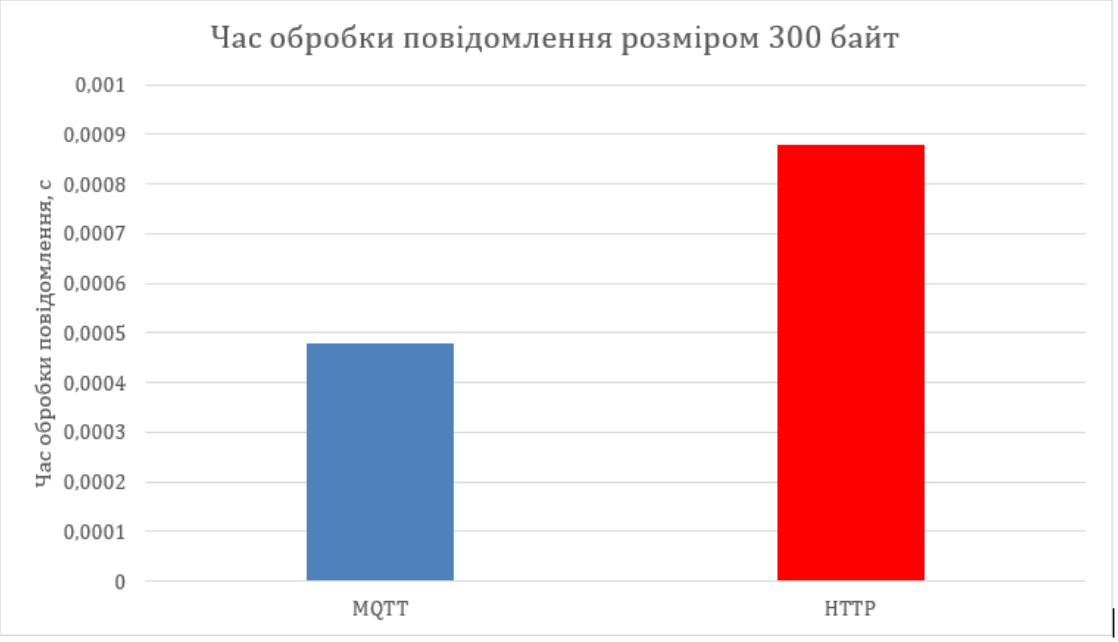

Рисунок 3.13 Час обробки повідомлення розміром 300 байт

На рис. 3.14 зображено порівняльну характеристику часу обробки повідомлення запиту розміром 10 000 байт.

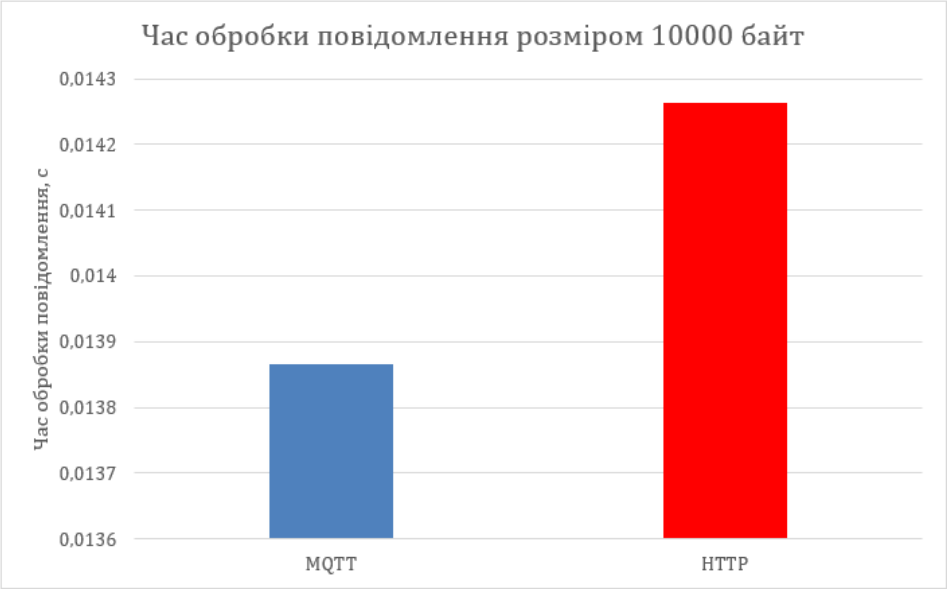

Рисунок 3.14 Час обробки повідомлення розміром 10 000 байт

На рисунку 3.15 зображено порівняльну характеристику часу обробки повідомлення запиту розміром 50 байт.

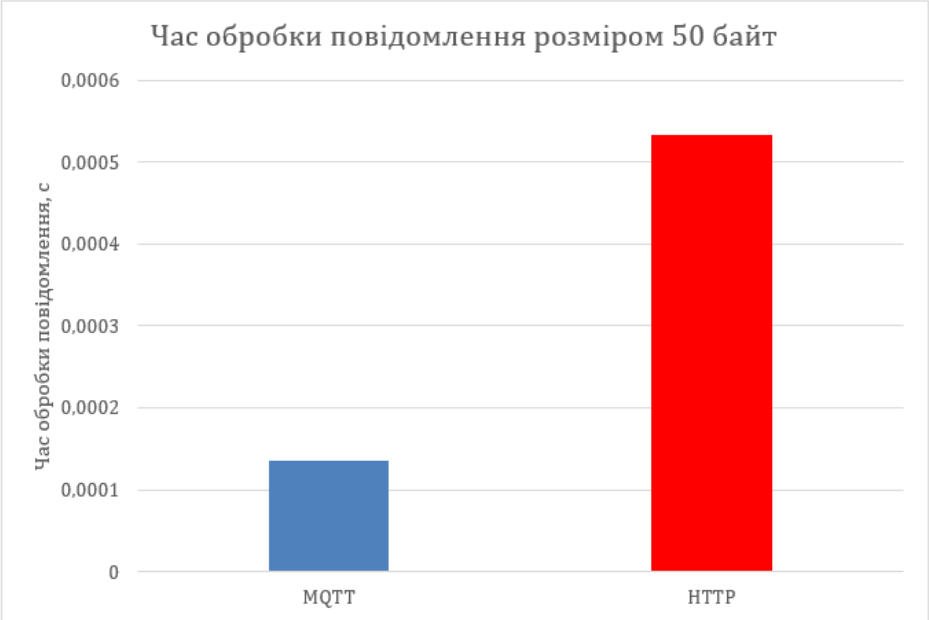

Рисунок 3.15 Час обробки повідомлення розміром 50 байт

У таблиці 3.7 наведено узагальнені результати розрахунків часу обробки запитів.

Таблиця 3.7 - Час обробки запитів

Час обробки запитів			
Розмір повідомлення	Назва мережі	MQTT	HTTP
50 (байт)	ISP1 (10Мбіт/с)	7.36×10^{-5} (с)	30.40×10^{-5} (с)
	ISP2 (100Мбіт/с)	0.0736×10^{-5} (с)	3.04×10^{-5} (с)
	ISP3 (16Мбіт/с)	5.4×10^{-5} (с)	53.34×10^{-5} (с)
300 (байт)	ISP1 (10Мбіт/с)	27.3×10^{-5} (с)	50.40×10^{-5} (с)
	ISP2 (100Мбіт/с)	2.73×10^{-5} (с)	5.04×10^{-5} (с)
	ISP3 (16Мбіт/с)	17.9×10^{-5} (с)	32.40×10^{-5} (с)
10 000 (байт)	ISP1 (10Мбіт/с)	803.3×10^{-5} (с)	826.40×10^{-5} (с)
	ISP2 (100Мбіт/с)	80.33×10^{-5} (с)	82.40×10^{-5} (с)
	ISP3 (16Мбіт/с)	502.9×10^{-5} (с)	517.4×10^{-5} (с)

Тепер роздивимось відповідь сервера клієнту. Нехай після встановлення TCP-з’єднання встановлений MSS (максимальний розмір сегмента), що дорівнює 1 460 байт. Це означає, що серверу потрібно відправити клієнту 7 TCP-сегментів, щоб передати свою 10 000-байтну відповідь. Перші 6 сегментів будуть мати кожний 1 460-байтний блок даних плюс 20-байтний TCP-заголовок. Останній сегмент буде мати блок даних довжиною 1 240 байт (≈ 10 000 – 6 х 1 460). По мірі руху відповіді від сервера до клієнта, до кожного сегменту добавляться ще IP-заголовок і заголовок кадра відповідної мережі.

Розрахуємо час обслуговування для відповіді в мережах 1, 2 і 3. Почнемо з мережі 3, Ethernet. Кожен з перших 6 сегментів згенерує 1518-байтний кадр (з них

1460 байт буде відведено на дані, 20 байт – TCP-заголовок, 20 байт - на IP-заголовок і 18 байт – на службову інформацію кадра Ethernet).

Останній сегмент згенерує 1298-байтний кадр. Отже, час обслуговування для відповіді в мережі 3 складе:

$$\frac{(6\times1518+1298)\times8}{16000000} = 0{,}005203, [\text{с}] \quad (3.47)$$

Розмір заголовку кадра и полоса пропускання мережі 1 та 2 відрізняються від відповідних значень для мережі 3. Таким чином, час обслуговування для відповіді складатиме:

$$\frac{(6\times(1460+20+20+2)+(1240+20+20+2))\times8}{10000000} = 0{,}008235, [\text{с}] \quad (3.48)$$

$$\frac{(6\times(1460+20+20+2)+(1240+20+20+2)\times8)}{100000000} = 0{,}000823, [\text{с}] \quad (3.49)$$

$$\frac{(6\times(1460+20+20+18)+(1240+20+20+18)\times8)}{16000000} = 0{,}00337, [\text{с}] \quad (3{,}50)$$

Час обслуговування 100 000- байтної відповіді:

$$\frac{(69\times(1460+20+20+2)+(720+20+20+2))\times8}{10000000} = 0{,}08352, [\text{с}] \quad (3.51)$$

$$\frac{(69\times(1460+20+20+2)+(720+20+20+2))\times8}{100000000} = 0{,}008352, [\text{с}] \quad (3.52)$$

$$\frac{(69\times(1460+20+20+18)+(720+20+20+18)\times8)}{16000000} = 0{,}05276, [\text{с}] \quad (3.53)$$

Час обслуговування 5 000- байтної відповіді:

$$\frac{(4\times(1460+20+20+2)+(620+20+20+2))\times8}{10000000} = 0{,}005336, [\text{с}] \quad (3.54)$$

$$\frac{(4\times(1460+20+20+2)+(620+20+20+2))\times8}{100000000} = 0{,}0005336, [\text{с}] \quad (3.55)$$

$$\frac{(4\times(1460+20+20+18)+(620+20+20+18)\times8)}{16000000} = 0{,}003375, [\text{с}] \quad (3.56)$$

Розрахуємо час обслуговування для відповіді в мережах 1, 2 і 3, при використанні протоколу HTTP.

Час обслуговування 10 000- байтної відповіді:

$$\frac{(6\times(1460+20+20+290)+(1240+20+20+290))\times8}{10000000} = 0{,}010608, [\text{с}] \quad (3.57)$$

$$\frac{(6\times(1460+20+20+290)+(1240+20+20+290))\times 8}{100000000} = 0{,}001061, [\text{с}] \quad (3.58)$$

$$\frac{(6\times(1460+20+20+18)+(1240+20+20+18)\times 8)}{16000000} = 0{,}00663, [\text{с}] \quad (3.59)$$

Час обслуговування 100 000- байтної відповіді:

$$\frac{(69\times(1460+20+20+290)+(720+20+20+290))\times 8}{10000000} = 0{,}100824, [\text{с}] \quad (3.60)$$

$$\frac{(69\times(1460+20+20+290)+(720+20+20+290))\times 8}{100000000} = 0{,}010082, [\text{с}] \quad (3.61)$$

$$\frac{(69\times(1460+20+20+308)+(720+20+20+308)\times 8)}{16000000} = 0{,}06315, [\text{с}] \quad (3.62)$$

Час обслуговування 5 000- байтної відповіді:

$$\frac{(4\times(1460+20+20+7)+(620+20+20+7))\times 8}{10000000} = 0{,}007744, [\text{с}] \quad (3.63)$$

$$\frac{(4\times(1460+20+20+2)+(620+20+20+2))\times 8}{100000000} = 0{,}000774, [\text{с}] \quad (3.64)$$

$$\frac{(4\times(1460+20+20+18)+(620+20+20+18)\times 8)}{16000000} = 0{,}00484, [\text{с}] \quad (3.65)$$

На рис. 3.16. зображено порівняльну характеристику часу обробки повідомлення відповіді розміром 10 000 байт.

На рис. 3.17. зображено порівняльну характеристику часу обробки повідомлення відповіді розміром 100 000 байт.

На рис. 3.18. зображено порівняльну характеристику часу обробки повідомлення відповіді розміром 5000 байт.

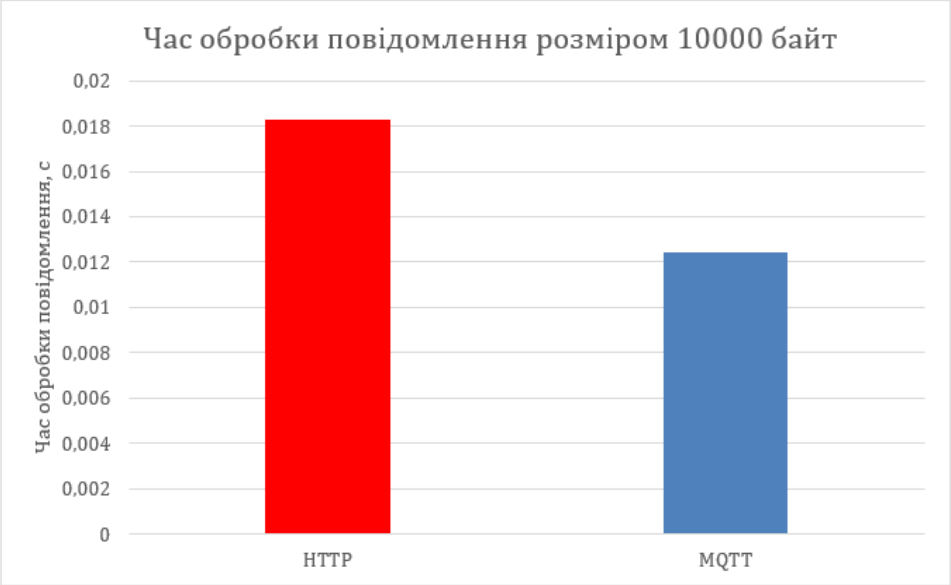

Рисунок 3.16 - Час обробки повідомлення розміром 10 000 байт

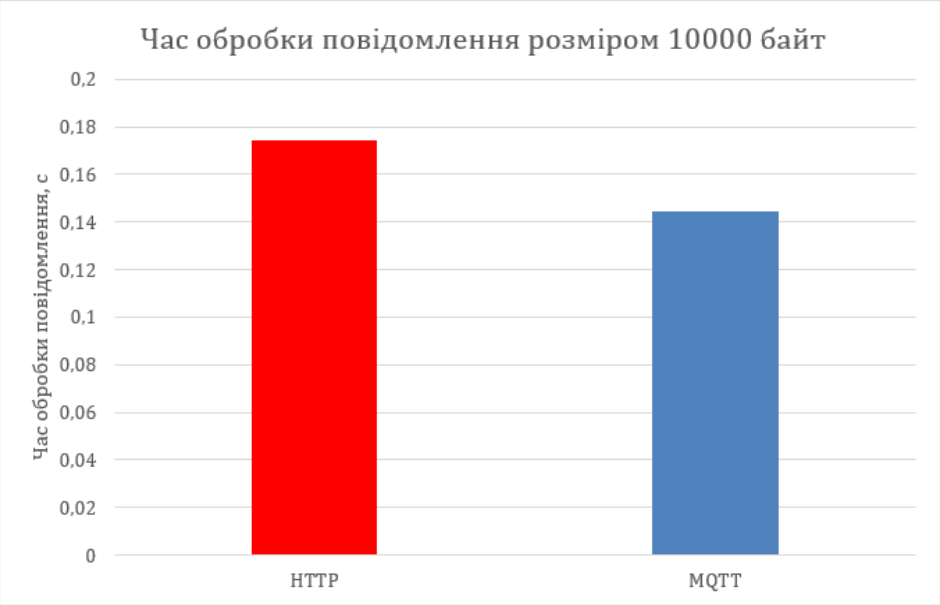

Рисунок 3.17 - Час обробки повідомлення розміром 100 000 байт

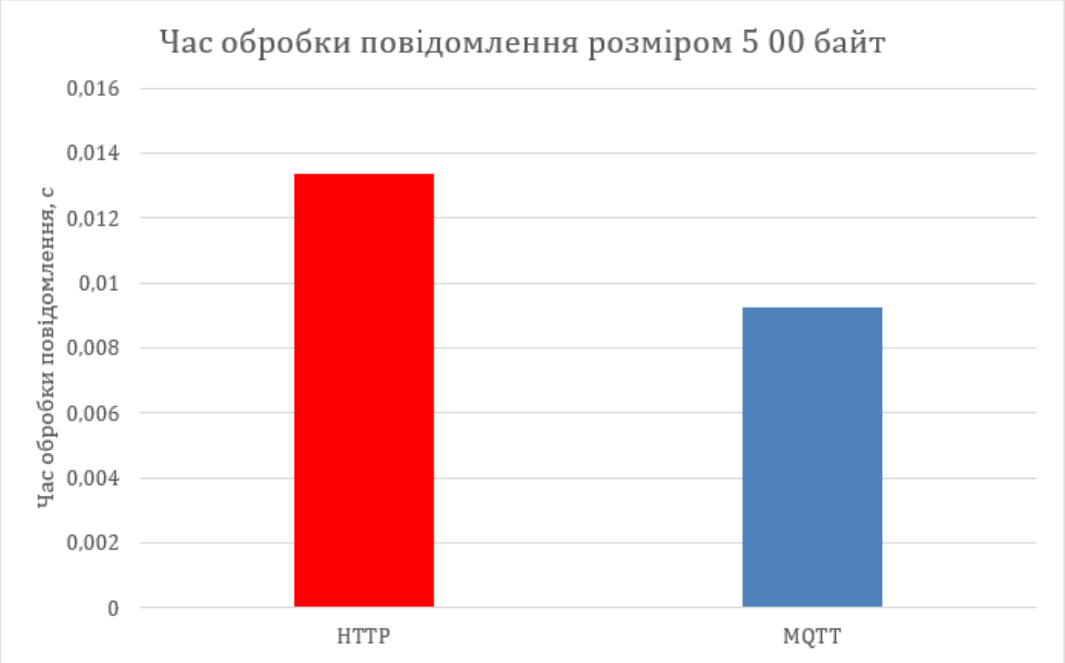

Рисунок 3.18 - Час обробки повідомлення розміром 5000 байт

Відмітимо, що оскільки MSS менше або дорівнює MTU для всіх мереж, то фрагментувати повідомлення не знадобиться. Сучасні IP-стандарти рекомендують хосту джерела повідомлення визначати мінімальне значення MTU по тракту передачі даних до вибору початкового розміру дейтаграми. Це дозволить уникати фрагментації з наступною зборкою повідомлення та прискорює обробку пакетів на проміжних маршрутизаторах і хості місця призначення.

Висновки до розділу 3

В 3 розділі було запропоновано методологію розрахунку продуктивності мережі.

Проведено аналіз сервісу IoT, розгорнутого на інфраструктурі AWS Amazon, що може бути реалізованим на транспортному рівні, протоколам MQTT чи HTTP.

AWS IoT забезпечує безпечне двостороннє спілкування між пристроями, підключеними до Інтернету, такими як датчики, виконавчі пристрої, вбудовані

мікроконтролери або інтелектуальні пристрої та Cloud AWS. Це дає змогу збирати телеметричні дані з кількох пристроїв, а також зберігати та аналізувати їх.

Запропонований метод складається з наступних розрахунків:

1) Розрахунок пропускної здатності для 5 типів запитів, та різних розмірів повідомлень. Отримані результати демонструють, що для передачі маленьких повідомлень краще використовувати протокол MQTT. При передачі великих файлів, краще HTTP, бо він не потребує додаткових налаштувань при розгортанні.

2) Середній час відгуку. З отримах результатів, бачимо, що різниці між протоколами майже не існує.

3) Пропускна здатність мережі при використанні кешуючого проксі-сервера. Для протокола MQTT такий сервер не передбачений. Для протокола HTTP виграш в пропускній здатності складає 1,107 Мбіт/с.

4) Розрахунок використання ЦП серверу являється важливим пунктом, бо це один з ключових елементів мережі та всієї системи. При використанні протоколу HTTP використання ресурсів ЦП склало 19%, а при MQTT – 13%.

5) Під час розрахунку часу обслуговування в мережах, було розраховано час, що витрачався на обробку запитів, для повідомлень розміром 300, 50 та 10 000 байт, для двох протоколів (MQTTі AMQP). Також було розраховано час відповіді для повідомлень розміром 5 000, 10 000 та 100 000 байт відповідно. Отримані результати наведені в таблицях 3.1, та 3.2.

ЗАГАЛЬНІ ВИСНОВКИ

1) Розглянуто основні поняття технології «Інтернету речей», IoT (Internet of Things). Інтернет речей складається з чотирьох основних рівнів, таких як: рівень сенсорів і сенсорних мереж, рівень шлюзів і мереж, сервісний рівень, рівень додатків.

2) Досліджено значення цифрового двійника в архітектурі IoT та аналіз його складових блоків. Цифровий двійник складається з набору різних інфокомунікаційних технологій, що забезпечують функціонування Інтернету речей, і його архітектура показує, як ці технології пов'язані один з одним. Зв’язок між фізичним та цифровим світами, який демонструє рис.1 підкреслює глибокий потенціал цифрового двійника: тисячі датчиків, що приймають безперервні, нетривіальні вимірювання, які транслюються на цифрову платформу, яка, у свою чергу, виконує аналіз в режимі реального часу для оптимізації бізнес-процесу.

3) Безпека і стійкість Інтернету речей - це ефективність оцінки ризиків та їх усунення. Безпека в IoT унікальна, тому що луди повинні довіряти технології. Унікальність полягає в тому, що повинна відбуватися ідентифікація, аутентифікація, зберігання та обробка інформації, в тому числі і фінансової. Також повинна забезпечуватись безпека доставки товарів, мультифакторна аутентифікація, автоматична фіксація передачі прав від одного контрагента до другого.

4) Досліджено методи інтеграції технології IoT з блокчейн. Шляхом використання блокчейна, стає можливим здійснювати мікроплатежі за рамками P2P-розрахунків публічного блокчейна. У контрольованій мережі приватного блокчейна, ми впроваджуємо Машинну Валюту для двосторонніх договорів і платежів між елементами цифрового двійника і, тим самим сприяємо більш доступним, надійним і безпечним процесам.

5) Проведено аналіз сервісу IoT, розгорнутого на інфраструктурі AWS Amazon, та розроблено методологію розрахунку продуктивності мережі. Запропонований метод складається з наступних розрахунків:

1. Розрахунок пропускної здатності для 5 типів запитів, та різних розмірів повідомлень. Отримані результати демонструють, що для передачі маленьких повідомлень краще використовувати протокол MQTT. При передачі великих файлів, краще HTTP, бо він не потребує додаткових налаштувань при розгортанні.

2. Середній час відгуку. З отримах результатів, бачимо, що різниці між протоколами майже не існує.

3. Пропускна здатність мережі при використанні кешуючого проксі-сервера. Для протокола MQTT такий сервер не передбачений. Для протокола HTTP виграш в пропускній здатності складає 1,107 Мбіт/с.

4. Розрахунок використання ЦП серверу являється важливим пунктом, бо це один з ключових елементів мережі та всієї системи. При використанні протоколу HTTP використання ресурсів ЦП склало 19%, а при MQTT – 13%.

5. Під час розрахунку часу обслуговування в мережах, було розраховано час, що витрачався на обробку запитів, для повідомлень розміром 300, 50 та 10 000 байт, для двох протоколів (MQTTі AMQP). Також було розраховано час відповіді для повідомлень розміром 5 000, 10 000 та 100 000 байт відповідно

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Росляков А. В. ИНТЕРНЕТ ВЕЩЕЙ : Учебное пособие / А. В. Росляков, С. В. Ваняшин, А. Ю. Гребешков. - Самара : ПГУТИ, 2015 - 136 с.
2. D. Evans, “The Internet of things: How the next evolution of the Internet is changing everything,”CISCO, San Jose,CA, USA,White Paper, 2011.
3. L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Comput. Netw., vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
4. R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future Internet: The Internet of Things architecture, possible applications and key challenges,” inProc. 10th Int. Conf. FIT, 2012, pp. 257–260.
5. J. Gubbi, R. Buyya, S.Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, Sep. 2013.
6. P. Lopez, D. Fernandez, A. J. Jara, and A. F. Skarmeta, “Survey of Internet of Things technologies for clinical environments,” in Proc. 27th Int. Conf. WAINA, 2013, pp. 1349–1354.
7. D. Yang, F. Liu, and Y. Liang, “A survey of the Internet of Things,” in Proc. 1st ICEBI, 2010, pp. 358–366.
8. Industry 4.0 and the digital twin [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www2.deloitte.com/insights/us/en/focus/industry-4-0/digital-twin-technology-smart-factory.html#endnote-11.
9. J. Gantz and D. Reinsel, “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east,” IDC iView: IDC Anal. Future, vol. 2007, pp. 1–16, Dec. 2012.
10. Васильков, А. Микрокомпьютеры для интернета вещей: от умного дома к поумневшему окружению [текст] / А. Васильков // Компьютерра, 14 июня 2013г.

11. Вишневский, В. Mesh-сети стандарта IEEE 802.11s – технологи и реализация [текст] / В. Вишневский, Д. Лаконцев, А. Сафонов, С. Шпилев // Первая миля. – 2008. – №2-3. – С. 26-31.
12. Восков, Л.С. Web вещей – новый этап развития интернета вещей [текст] / Л.С. Восков , Н.А. Пилипенко // Качество. Инновации. Образование. – 2013. – № 2. – С. 44-49.
13. Гайкович, Г.Ф. Стандартизация в области промышленных сетей. Развитие беспроводных стандартов для АСУ ТП [текст] / Г.Ф. Гайкович // Электронные компоненты. – 2009. – №1. – С. 48-53.
14. Гиббс, М. Интернет вещей – не только для «умных» [текст] / М. Гиббс // Сети/network world. – 2013. – №3.
15. Архитектура безопасности "Интернета вещей" [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/azure/iot-suite/iot-security-architecture.
16. Эталонная архитектура безопасности интернета вещей (IoT) [Електронний ресурс] – Режим доступу до ресурсу: https://www.anti-malware.ru/practice/solutions/iot-the-reference-security-architecture-part-1.
17. Доктор Гринспен Гидеон, "Приватный мультиблокчейн – Технический документ," Coin Sciences , http://www.multichain.com/download/MultiChain-White-Paper.pdf , (2014)..
18. AWS IoT Button [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/iotbutton/.
19. V. A. F. Almeida, D. A. Menasc.e, R. Riedi, F. P. Ribeiro, R. Fonseca, and W. Meira Jr., "Analyzing Web Robots and their Impact on Caching," Proc. Sixth Workshop on WebCa ching and Content Distribution, Boston, Massachusetts, June 20-22, 2001.
20. V. A. F. Almeida, D. A. Menasc.e, R. Riedi, F. P. Ribeiro, R. Fonseca, and W. Meira Jr., "Characterizing and Modeling Robot Workload on E-Business

Sites," Proc. 2001 ACM SIGMETRICS Conf. Measurement Comput. Syst., ACM, Boston, Massachusetts, June 16-20, 2001.

21. V.A.F. Almeida, A. Bestavros, M. Crovella, and A. Oliveira, "Characterizing Reference Locality in the WWW," Fourth Int. Conf. Parallel Distrib. Inform. Syst. (PDIS), IEEE Comput. Soc, Dec. 1996, Miami Beach, Florida, pp. 92-103.
22. J. M. Almeida, V. A. F. Almeida, and D. Yates, "Measuring the Behavior of a World-Wide Web Server," Proc. Seventh Conf. High Perform. Networking (HPN), IFIP, Apr. 1997, pp. 57-72.
23. M. Arlitt and C. Williamson, "Web Server Workload Characterization: the Search for Invariants," Proc. 1996 ACM SIGMETRICS Conf. Measurement Comput. Syst., Philadelphia, Pennsylvania, May 1996, pp. 126-137.
24. T. Berners-Lee, R. Cailliau, H. Nielsen, and A. Pecret, "The World Wide Web," Comm. ACM, vol. 37, no. 8, pp. 76-82, Aug. 1994.