ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Освітньо-професійна програма Інформаційні управляючі системи та технології

Спеціальність 126 Інформаційні системи та технології

Ступінь вищої освіти Магістр

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_____ Юрій УТКІН

«15» грудня 2022 року

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Моделювання якості обслуговування хмарного сервісу архітектури Internet of Things»**

виконав здобувач вищої освіти денної форми навчання

Єгуньков Олександр Олександрович

Керівник кваліфікаційної роботи,
професор , д. т. н. Юрій ПОНОЧОВНИЙ

Полтава – 2022 року

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IoT - Internet of Things

IPv6- нова версія IP-протоколу версії 6

IPv4- четверта версія мережевого протоколу IP

IIRA- Industrial Internet Reference Architecture

IEC- Міжнаро́дна електротехні́чна комі́сія

ОС- Операці́йна систе́ма

UDP - User Datagram Protocol

TCP -Transmission Control Protocol

HTTP - Hyper Text Transfer Protocol

MQTT - Message Queue Telemetry Transport

ВСТУП

Актуальність теми. Розвиток Internet призвів до поширення його супутніх технологій і Інтернет речей (Internet of Things, IoT) є однією з них [1]. Широкого поширення термін Інтернет речей здобув не так давно. Хоча й перший раз застосування даного терміну відбулося в 1999 році при роботі над стандартом розмітки об'єктів за допомогою RFID-міток, призначеним для логістики. Сьогодні, інтернет речей став популярним терміном для опису сценаріїв, у яких інтернет з'єднання і обчислювальна логіка поширюються на велику кількість об'єктів, пристроїв, датчиків і повсякденних об'єктів [2].

Інтернет речей є одним із найбільш перспективних напрямків розвитку інформаційно-комунікаційних технологій. Кількість підключених до мережі Інтернет пристроїв бурхливо зростає, і це вимагає сучасних підходів до побудови високонавантажених серверних систем. Платформи Інтернету речей мають забезпечувати можливість аналізувати різні аспекти даних, що потрібно для оптимізації різноманітних виробничих та інших процесів.

Проблеми та задачі у сфері побудови серверних систем для Інтернету речей можна поділити на загальні, які притаманні багатьом іншим системам обробки великих даних, та специфічні, що виникають лише в цій області [3]. Загальними задачами є побудова та використання ефективних, відмовостійких, масштабованих і розподілених систем роботи з даними. Специфічними для Інтернету речей проблемами є забезпечення надійного керування та моніторингу пристроїв. Сьогодні прискоренню його розвитку сприяють комерційний інтерес і початок роботи над референтними архітектурами для конкретних галузей [4].

Питаннями розвитку методів та моделей оцінювання якості обслуговування, надійності та готовності хмарних сервісів та систем IoT упродовж останніх десятиліть активно займалися В. Харченко, А. Горбенко, К. Тріведі, А. Боярчук. У проведених дослідженнях аналізуються аналітичні моделі хмарних IoT сервісів з прив‘язкою до спеціалізованих засобів моделювання. Проте питання побудови моделей оцінки якості оновлених компонент IoT хмарної архітектури залишається

актуальним.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах ініціативної науково-дослідної теми «Методи та технології оцінювання функційної безпечності FPGA-програмованих логічних контролерів на підприємствах агропромислового комплексу», яка була проведена на кафедрі інформаційних систем та технологій Полтавського державного аграрного університету (2021 р.).

Метою кваліфікаційної роботи є підвищенні точності оцінювання показників якості обслуговування за рахунок розробки моделі компоненти архітектури сервісу IaaS для підтримки системи Internet of Things.

Завданнями кваліфікаційної роботи є:

– аналіз варіантів реалізації систем Internet of Things,

– моделювання архітектур хмарних сервісів,

– побудова та дослідження моделі компоненти архітектури сервісу IaaS для підтримки системи Internet of Things.

– оцінка практичного застосування розробленої моделі на прикладі компоненти хмарної архітектури системи вирішувача, оцінювання ефективності її використання.

Об'єктом дослідження є процеси функціонування систем інтернету речей в умовах відмов компонент хмарного сервісу через зайнятість ресурсів.

Предметом дослідження є модель сервісу IaaS для підтримки системи Internet of Things.

Методи дослідження – проведені в роботі дослідження базуються на методах теорії ймовірності, системного і марковського аналізу, систем масового обслуговування, які використовувалися при розробці комплексу марковських моделей системи Internet of Things.

Інформаційна база кваліфікаційної роботи складається з наукових статей, міжнародних аналітичних видань і звітів, матеріалів наукових конференцій Інтернет-ресурсів, що місять інформацію про архітектуру сучасних вебсистем, а також даних, отриманих від провідних IT-компаній у сфері Internet of Things.

Елементи наукової новизни полягають у розроблені та досліджені аналітичної моделі функціонування архітектури сервісу IaaS для підтримки системи Internet of Things в умовах відмов в обслуговуванні через зайнятість ресурсів.

Практична значущість роботи полягає в можливості повторного застосування та модифікації розробленого програмного коду моделі для оцінювання показників якості обслуговування інформаційних систем з відмовами і обмеженими чергами обслуговування а також показників енергоспоживання та економічної ефективності. Отримані результати можуть бути корисними для IT фахівців при моделюванні спеціалізованих інформаційних управляючих систем.

Апробація результатів дослідження відбувалася шляхом оприлюднення доповідей на наукових конференціях, семінарах.

Публікації. За результатами проведеного дослідження опубліковано тези: «Захист облікових записів за допомогою служб Active Directory», Матер. щорічної студентської наукової конференції Полтавського державного аграрного університету, 10 листопада 2022 р. м. Полтава; «V-подібна модель розробки FPGA пристроїв», Матер. науково-практичної конференції за підсумками виробничої практики здобувачів вищої освіти спеціальності «Інформаційні системи та технології», 23 лютого 2022 р., м. Полтава.

Структура та обсяг кваліфікаційної роботи логічно пов'язані з задачами досліджень. Робота містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг текстової частини дипломної роботи складає 74 сторінки формату А4. Вона містить 11 рисунків і 5 таблиць. В роботі використано 53 науково-технічних джерела.

РОЗДІЛ 1
АНАЛІЗ ВАРІАНТІВ РЕАЛІЗАЦІЇ СИСТЕМ INTERNET OF THINGS

1.1 Аналіз концепції систем «Інтернет речей»

Термін Інтернет речей (Internet of Things, IoT) вперше був використаний Кевіном Ештоном при роботі над стандартом розмітки об'єктів за допомогою RFID-міток, призначеним для логістики, але ідея масового впровадження обчислень в навколишні об‘єкти з'явилася ще в кінці 1980-х років [5]. Важливою віхою була поява протоколу IPv6, завдяки якому і став можливий Інтернет речей. А сьогодні прискоренню його розвитку сприяють комерційний інтерес і початок роботи над референтними архітектурою для конкретних галузей [6]. У Google, наприклад, анонсували розробку Brillo - операційної системи для пристроїв Інтернету речей і «розумного» будинку [7]. З'явилися комерційні апаратні компоненти для міжмашинного зв'язку за стандартами Bluetooth, ZigBee, IPC Global і Wi-Fi малої потужності. У Microsoft оголосили, що Windows 10 зможе працювати на популярних платах прототипування, таких як Raspberry Pi 2 [8]. В Samsung і ряді інших підприємств випустили нове покоління чіпів для «розумних» пристроїв [9]. З'явилося чимало проектів, в яких мережеві мікроконтролери використовуються як центральні керуючі елементи для систем датчиків, приводів і радіоміток. Навіть зараз застосування Інтернету речей ще є ізольованим і, призначеним для конкретних систем і сценаріїв. Коли стала відчутною потреба в референтних архітектурах, з'явилися проекти стандартизації, що ставлять за мету поліпшити і спростити розробку (табл.1.1) [10]. Стандарт Reference Architecture Model Industrie (RAMI) 4.0 виходить за рамки Інтернету речей, так як додатково дає рекомендації, що стосуються особливостей виробництва і логістики. У свою чергу, Industrial Internet Reference Architecture (IIRA) в основному орієнтований на промислові застосування. У проекті IoT-A в деталях враховані аспекти Інтернету речей, що стосуються IT. В області міжмашинного зв'язку є великі проекти стандартизації, що базуються на ефективних, масштабованих, захищених технологічних стеках. Ці

стандарти спираються на традиційну модель OSI, пропонуючи специфікації для канального, прикладного, мережевого і транспортного рівнів.

Таблиця 1.1 – Проекти стандартизації IoT

Категорія	Ініціатива	Опис	Стан
Архітектурні моделі	Reference Architecture Model Industrial 4.0	Референтна архітектура для «розумних» заводів на стандартах Інтернету речей.	Версія 1
	Industrial Internet Reference Architecture	Архітектура від альянсу Industrial Internet Consortium (заснований AT&T, Cisco, General Electric, IBM та Intel	Версія 1.7
	Internet of Things Architecture	IoT-A- деталізована архітектурна модель з акцентом на інформаційних та функціональних аспектах. В рамках цієї ініціативи виконаний детальний аналіз системних вимог	Версія 3.0
	Стандарт на архітектурні основи Інтернет речей (IEEE P2413)	Проектом займається комітет IEEE	2413-2019 - IEEE Approved Draft Standard for an Architectural Framework for the Internet of Things (IoT)
	Arrowhead Framework	Великий європейський проект розробки оптимальних методів автоматизації процесів з допомогою вбудованих систем	Проект оновлюється
Стандарти для IoT	Комітет M2M Європейського інституту телекомунікаційних стандартів	Займається розробкою стандартів зв'язку для Інтернет речей	Доступний цілий ряд готових стандартів
	Сектор стандартизації телекомунікацій Міжнародного союзу електрозв'язку (ITU-T)	Координує розробку індикаційних систем для M2M	Доступний цілий ряд готових стандартів
Інші ініціативи	European Reseach Cluster on the Internet of Things(IREC)	IREC займається багатьма питаннями, які відносяться до Інтернету речей, в тому числі підключення об'єктів, до глобальної мережі	Напрацювання регулярно поновлюється
	Smart Appliance	Ініціатива Євросоюзу – акцент на взаємодії «розумної» побутової техніки на принципах Semantic Web	Чернетка Smart Appliance Reference

На сьогоднішній день розроблено дві основні архітектури Інтернету речей - IoT-A і IIRA. Перша описана більш детально і постійно розширюється, друга поки знаходиться на стадії обговорення. IoT-A семантично орієнтована, надає можливість інтерпретації даних для техніко-економічних обґрунтувань і фокусується на загальних концепціях інформатики. IIRA, в свою чергу, акцентується на галузевій діяльності: бізнесі, операціях (прогнозування, моніторинг, оптимізація і т.п.), інформації (аналітика і дані) і додатках (призначених для користувача інтерфейсів, інтерфейсів програмування, правил, логіки). Специфікація RAMI 4.0, розрахована на виробничі застосування, розширює IIRA, доповнюючи функціональний рівень двома концепціями, визначеними в стандартах IEC 62890, 62264 і 61512, - життєвого циклу і потоку створення активів. У IoT-A досить повно охоплені моделювання та структурування управління бізнес-процесами на основі Інтернету речей, віртуальні об'єкти, сервіси IoT і крос-сервісна організація з точки зору функціональності, інформації та областей застосування. Хмарні аспекти, в тому числі серверна архітектура і управління нею, визначаються конкретною реалізацією. Те саме можна сказати про агентські програми і код, що працює на пристроях для конкретних застосувань. У IIRA ці питання теж розглядаються, але ця модель більше фокусується на техніко-економічних обґрунтуваннях і сценаріях застосування.

Обидві архітектури коротко згадують міжмашинний зв'язок, вказуючи, зокрема, що мережевий рівень можна реалізувати за допомогою IPv6, а мережевий і транспортний - на протоколах UDP і CoAP (Constrained Application Protocol). Замість HTTP можна використовувати більш легкий протокол MQTT, що працює поверх TCP/IP. В IoT-A і в IIRA приділено багато уваги орієнтуванню на речі (датчики, приводи, мітки і т.п.). В обох архітектурах на всіх рівнях передбачаються механізми управління і безпеки. В цілому обидва архітектурних шаблони задають і роз'яснюють загальну структуру Інтернету речей, описуючи моделі обробки даних і взаємодії людей і пристроїв з використанням стандартів між машинного зв'язку. У табл. 1.2 перераховані архітектурні рівні і протоколи згідно з нинішніми версіями стандартів.

Таблиця 1.2 – Рівні архітектури та протоколи Інтернету речей [1,11]

Рівень	Протоколи
Семантична орієнтація	Сервісні протоколи, в тому числі OPC Unified Architecture (OPC UA), Universal Plug and Play (UPnP), Devices Profile for Web services (DPWS), Constrained Application Protocol (CoAP), та EXI (Efficient XML Interchange)
Орієнтація на інтернет	Механізми стиковки та конвертації протоколів на основі UDP або TCP з HTTP або MQTT. Підтримка IPv4 або IPv6
Орієнтація на речі	Фізичний и канальний рівні з відповідними комунікаційними протоколами, здатними забезпечити простоту встановлення та супроводу

В рамках різних ініціатив розроблено вже чимало моделей, архітектурних проектів та інструментів, прийшов час об'єднувати різні підходи і галузеві стандарти, інакше виникають перешкоди для реалізації ключових механізмів Інтернету речей. Один з таких механізмів - ефективний, захищений зв'язок. Відповідних технологій сьогодні існує чимало, але перш за все необхідні захищені, прості бездротові мережі малої потужності. Апаратні компоненти, які підходять для створення Інтернету речей, вже доступні у продажу, але поки немає стандартних мережевих операційних систем (ОС) реального часу для різних предметних областей, хоча є величезна різноманітність таких систем.

Для певних сценаріїв застосування Інтернету речей можна, наприклад, задіяти існуючі стандарти на програмовані логічні контролери з циклічним або дієвим викликом завдань. І навпаки, інструментарії розробки, створені для Інтернету речей, можна використовувати в сфері автоматизації промислових процесів. Крім того, практично відсутні стандарти, пов'язані з людино-машинним інтерфейсом і аналізом великих даних. Першим кроком до стандартизації може стати створення семантичних описів для пристроїв, параметрів і призначених для користувача інтерфейсів.

1.2 Безпека в середовищі IoT

В основі безпеки лежить моделювання ризиків. Завдання моделювання ризиків полягає в тому, щоб зрозуміти, як саме зловмисники могли б проникнути в систему, а потім - вжити відповідних заходів для зниження ризиків. Завдяки моделюванню ризиків можливо передбачити та впровадити заходи що до зниження ризиків ще на етапі проектування системи, а не після її розгортання. Це важливо, оскільки модернізація систем безпеки на нескінченну кількість пристроїв, що працюють за межами компанії, технічно нездійсненна. До того ж такий підхід загрожує появою помилок і залишає клієнтів без захисту від будь-яких ризиків.

В світі активно працюють над визначенням і документуванням функціональних вимог до системи, які зроблять взаємодію користувачів з нею ще зручніше та ефективніше [13]. Проте виявлення неочевидних способів, якими зловмисники могли б проникнути в систему, - набагато складніше завдання. Моделювання ризиків дозволяє зрозуміти, які саме дії можуть зробити зловмисники і з якою метою. В ході цього структурованого процесу обговорюються рішення з проектування системи безпеки, а також зміни, внесені в проект на етапі реалізації, які можуть вплинути на безпеку системи. Модель ризиків - це просто документ. Однак він є ідеальним способом забезпечувати безперервну передачу знань і зберігати напрацьований досвід, щоб дати новим фахівцям можливість миттєво зорієнтуватися в новому середовищі. І нарешті, завдяки моделюванню ризиків можливо враховувати також і інші аспекти безпеки. Наприклад, визначити той рівень безпеки, який провайдери готові надати своїм клієнтам. Цей задокументований рівень разом з моделлю ризиків надає дані, які потрібні для тестування конкретного рішення IoT.

Спеціалізовані підключення складаються з багатьох точок потенційної взаємодії, які необхідно врахувати при розробці адекватної системи захисту

цифрового доступу до цих пристроїв. Терміном «цифровий доступ» позначаються всі операції, що виконуються шляхом прямої взаємодії з пристроєм, якщо безпека забезпечується за допомогою фізичного контролю доступу. Наприклад, можна помістити пристрій в приміщення, що закривається на замок. Хоча фізичний доступ не можна заборонити програмними і апаратними засобами, все одно можна вжити певних заходів, щоб запобігти проникненню в систему шляхом фізичного доступу.

При вивченні різних механізмів взаємодії увага повинна в рівній мірі бути спрямована як на управлінні пристроєм, так і на обмін даними між пристроями. Управління пристроєм можна розглядати через інформацію, яка надається пристрою будь-яким учасником обміну даними з метою вплинути на поведінку пристрою таким чином, щоб змінити його стан або стан його середовища. Дані на пристрої - це інформація, яку створює сам пристрій і надає іншому учаснику обміну даними. Це відомості про стан пристрою і зареєстрований стан його середовища.

Щоб забезпечити більш високий рівень безпеки, рекомендується розділити стандартну архітектуру IoT на кілька компонентів (зон) [14]. До них відносяться:

– пристрій;

– польовий шлюз;

– хмарні шлюзи;

– служби.

Поділ на зони являє собою універсальний спосіб сегментування рішення. Кожна з зон часто містить власні дані, а також пред'являє окремі вимоги до перевірки автентичності та авторизації. Поділ на зони можна використовувати, щоб локалізувати збиток і обмежити вплив зон з низькою довірою на зони з більш високою довірою.

Кожна зона відділяється кордоном довіри, яка на рис.1.1 позначена червоним пунктиром. Вона показує перехід даних (інформації) з одного джерела в інший. Під час такого переходу дані (інформація) можуть стати об'єктом підробки (S –

Spoofing), незаконної зміни (T – Tampering), спотворення сенсу (R - Repudiation), розкриття інформації (I - Information Disclosure), відмови в обслуговуванні (D - Denial of Service) і підвищення привілеїв (E - Elevation of Privilege). Всі ці загрози в сукупності називають за першими літерами англійських слів абревіатурою «STRIDE». Компоненти, позначені в межах кожної з меж, також схильні до загроз STRIDE, що дозволяє повністю, у всіх деталях, змоделювати ризики для конкретного рішення. Далі буде розглянуто кожен з цих компонентів і конкретні проблеми безпеки (рис.1.1).

Зона пристрою являє собою фізичний простір, що безпосередньо оточує пристрій. У середовищі пристрою здійснюється фізичний доступ до пристрою і (або) спеціальний робочий цифровий доступ через локальну мережу.

Зона польового шлюзу, Польовий шлюз - це пристрій, програмне забезпечення або серверне програмне забезпечення загального призначення, що слугує засобом реалізації обміну даними, а також, по можливості, системою управління пристроями і центром обробки даних, одержуваних від пристроїв. У зону польового шлюзу входить сам польовий шлюз, а також всі підключені до нього пристрої.

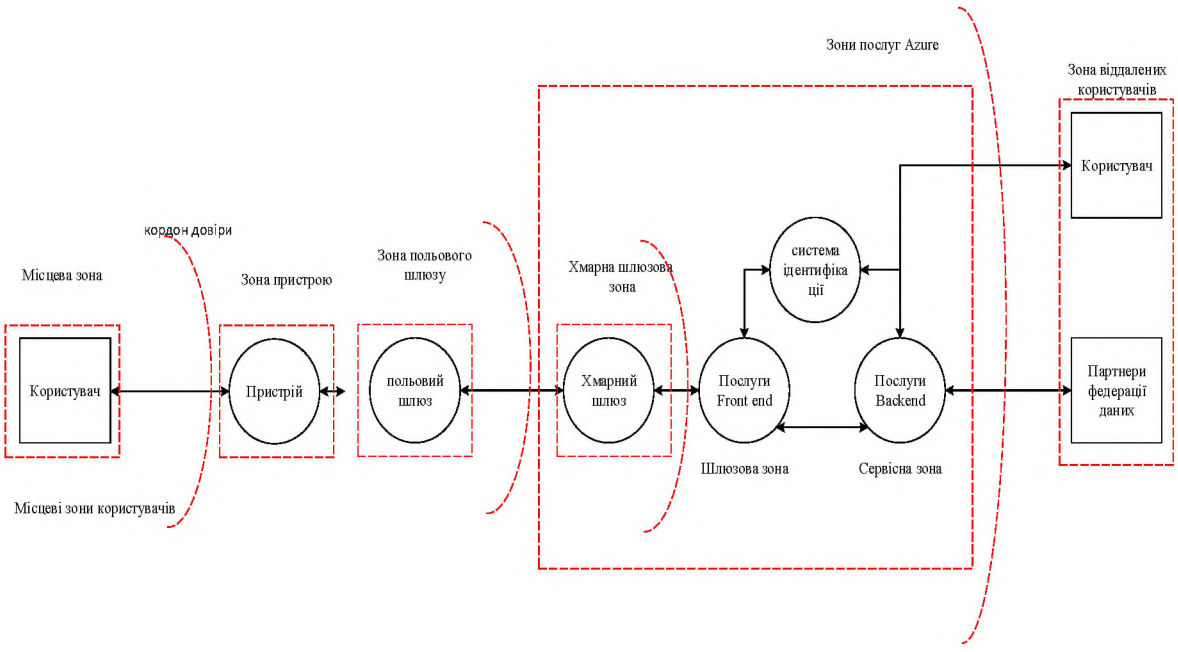


Рисунок 1.1 – Модель ризиків стандартної архітектури IoT

Зона хмарного шлюзу являє собою систему для віддаленого обміну даними між різними пристроями і польовими шлюзами. Дані, як правило, надходять від різних сайтів в загальнодоступній мережі в хмарну систему аналізу і контролю даних, що є федерацією таких систем.

Зона служб визначається як будь-який програмний компонент або модуль, який взаємодіє з пристроями за допомогою польового або хмарного шлюзу, виконуючи збір і аналіз даних, а також здійснюючи контроль і відправку команд.

1.3 Механізми Інтернету речей

Мережа дає важливу модель взаємодії для Інтернету речей - можливість отримувати інформацію про пристрої і, в деяких випадках, контролювати їх з допомогою браузера. Інтернет речей розширить звичайний WWW, підключивши до Всесвітньої мережі широке коло програмно-апаратних комплексів та електронних пристроїв. Пристрої в складі Інтернету речей, які управляються і передають інформацію за допомогою веб-технологій називають «Фізичною всесвітньою павутиною» (Physical Web). Будь-яка взаємодія між такими пристроями здійснюється із застосуванням ідентифікаторів, причому в Інтернеті речей в якості ідентифікаторів для глобальної мережі пристроїв потрібно використовувати 128-розрядні адреси IPv6. Можна також застосовувати уніфіковані ідентифікатори ресурсу URI, що включають в себе покажчики і імена, як більш високорівневу альтернативу, що прокладає міст між пристроями Інтернету речей і традиційними веб-технологіями URL (універсальний покажчик ресурсу, використовується спільно з сервісом доменних імен для маршрутизації і з'єднання з сервісами) та URN (універсальне ім'я ресурсу - Uniform Resource Names). Відмінною особливістю Фізичної павутини є використання URI як основного ідентифікатора.

Розширення визначення Інтернету речей має на меті включити в нього будь-яку людину чи предмет на планеті. Таким чином, концепція Фізичної павутини

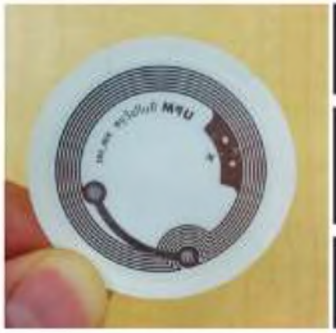
охоплює не тільки розумні пристрої. Для з'єднання з Інтернетом потрібна підтримка мережевих технологій і засобів обробки, але пристрої, у яких їх немає, могли б користуватися ресурсами сусідніх, що виконують роль шлюзів. Це дозволить мільярдам людей і різних пристроїв брати участь в Інтернеті речей, тим більше що у більшості людей сьогодні вже є такий шлюз - смартфони, чисельність користувачів яких вже перевищила мільярд. На рис.1.2 показані дві моделі взаємодії, які можна реалізувати за допомогою смартфонів. При прямій взаємодії смартфон може опитувати стан пристрою поблизу себе і виконувати роль моста між низькорівневими одноранговими протоколами, такими як Bluetooth або Wi-Fi, і протоколами Інтернету на приклад HTTP і TCP. Один із прикладів - фітнес-трекер Fitbit, який завантажує число пройдених кроків через смартфон користувача через 4G-мережу в його хмарний аккаунт. Через веб-сервіс на своїх смартфонах мобільні користувачі, що знаходяться поблизу від об'єкта Інтернету речей, можуть шукати пов'язану з ним інформацію, яка опублікована зацікавленими сторонами, - наприклад, постер фільму, який дозволяє проходячи повз нього людям автоматично звернутися до веб-сторінки кінотеатру і купити електронні квитки.

Рисунок 1.2 – Два способи взаємодії з Інтернетом речей: а - прямий; б - через посередника

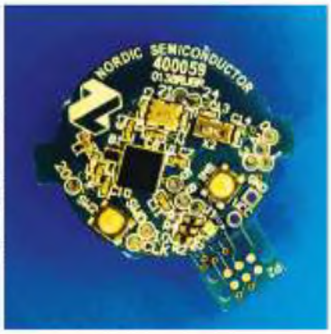
Чому ж Інтернет речей ще не став одним із стандартних інструментів бізнесу, а залишається предметом міркувань і домислів, висловлюваних експертами на заходах електронної галузі, серед яких, наприклад, Consumer Electronics Show? Відповідь полягає в тому, що Інтернет речей існує лише для невеликої кількості технологій, на базі яких вдалося побудувати успішні бізнес-моделі. Як правило, це закриті екосистеми, інтерфейси програмування і формати даних, що суперечить самій ідеї відкритих систем (з орієнтацією на які спочатку створювалися стандарти Інтернету), що грає на руку нинішнім закритим комерційним структурам на зразок Apple AppStore і Facebook. Сьогодні у продажу є системи домашньої автоматизації, що з'єднуються з Інтернетом і зазвичай комплектуються мостом, який управляє ресурсами автоматизації за допомогою протоколів всередині домашньої мережі і через відкриті протоколи спілкується із зовнішнім веб-сервісом. Користуючись настільним комп'ютером або смартфоном як клієнтським пристроєм, можна управляти своїм будинком шляхом взаємодії з онлайн-сервісом. При цьому покупець тим самим фактично безкоштовно надає виробнику пристрою обладнання для відображення призначеного для користувача інтерфейсу.

Для повної реалізації потенціалу Інтернету речей необхідно вирішити проблему масштабування, адже адресний простір Всесвітньої мережі збільшиться на кілька порядків, тому обов'язковою вимогою стане підтримка стандарту IPv6, робота над яким в IETF ведеться вже досить давно. Однак значну частку об'єктів Інтернету речей складуть пасивні пристрої, нездатні безпосередньо зв'язуватися з Мережею по кабельному або радіоз'єднанню, і, щоб такі об'єкти отримали представництво в мережі, знадобляться мітки, смартфони і посередницькі веб-сервіси. З усіх ідей, пов'язаних з Інтернетом речей, ідея підтримки пасивних пристроїв сьогодні найменш опрацьована.

Існує кілька видів міток для фізичних об'єктів, що зв'язують їх з посередницькими веб-сервісами (рис.1.3).


NFC-мітка


QR-код



мітка Bluetooth Low Energy

Рисунок 1.3 – Різні види електронних міток, які підходять для фізичної павутини

На початку 2000-х років ідентифікаційні радіомітки (RFID) вважалися однією з головних технологій, які наближують можливість створення Інтернету речей. В організації EPC Global розробили стандарт UHF RFID, розраховуючи забезпечити можливість подальшої автоматизації транзакцій в роздрібній торгівлі і замінити штрих-коди на мітки, які машина може зчитувати з відстані до трьох метрів. Однак після серії випробувань спільно з великими торговельними мережами, в тому числі з Walmart і Tesco, стандарт не набув поширення, оскільки мітки часто не зчитувалися через перешкоди і незручності доступу. Нові перспективи у RFID виникли з появою технології зв'язку ближньої дії (NFC), призначеної для виконання електронних платежів, і хоча лише деякі моделі смартфонів сьогодні мають приймачі NFC, ймовірність поширення таких апаратів в подальшому висока. З їх допомогою можна було б зчитувати пасивні NFC-мітки, здатні зберігати URI, - недорогі, компактні і тонкі, що прикріплюються практично до чого завгодно. У вересні 2014 року було оголошено, що iPhone буде підтримувати технологію NFC для сервісу ApplePay, а з огляду на частку ринку Apple, можна припустити, що й інші виробники смартфонів підтримають NFC, це дозволить технології стати однією з основ Інтернету речей. Оптичні мітки - це, зокрема, популярний QR-код, успіх якого безпосередньо обумовлений наявністю пристрою зчитування (камери високої роздільної здатності) у сучасних смартфонах. QR-код розпізнається і перетворюється в число, текст або URI, а самі коди роздруковуються на багатьох

видах продукції, на рекламних плакатах і купонах, а також демонструються в телевізійній рекламі. Але на практиці багато рекламних кампаній, організовані з використанням QR-кодів, виявляються низько ефективними. Причина пов'язана з необхідністю наявності на смартфоні спеціального додатка для зчитування QR-кодів, а також з тим, що досить складно розташувати смартфон таким чином, щоб камера могла сфокусуватися і чітко зняти зображення. Bluetooth Low Energy (BLE) це частина стандарту Bluetooth 4.0, прийнята асоціацією Bluetooth Special Interest Group ще в 2010 році, тому всі смартфони, що вийшли за останні кілька років, мають реалізацію BLE в чіпсетах Bluetooth і підтримують ряд відповідних функцій на рівні ОС. Чіпсет Bluetooth можна зменшити, залишивши тільки підтримку BLE і відмовившись від сумісності зі звичайним Bluetooth. В результаті вийде компактний недорогий компонент, який можна використовувати в якості електронної мітки. Мітки на основі BLE можуть сповіщати про свою присутність, передаючи раз в секунду пакет даних. Енергії на це йде мало - мітка може працювати до року на літієвому елементі ємністю 240 мА*год розміром з монету. Доступність недорогої електроніки для міток і наявність в більшості смартфонів чіпсетів Bluetooth стали хорошим каталізатором застосування BLE. У багатьох відомих IT-компаніях і в значній кількості стартапів експериментують з бізнес-моделями, в яких використовуються подібні мітки. Як і інші технології міток, BLE може сприяти розвитку Інтернету речей, оскільки підходить для повсюдного використання, забезпечує більш високу точність зчитування і може вбудовуватися в виріб, не псуючи його зовнішнього вигляду.

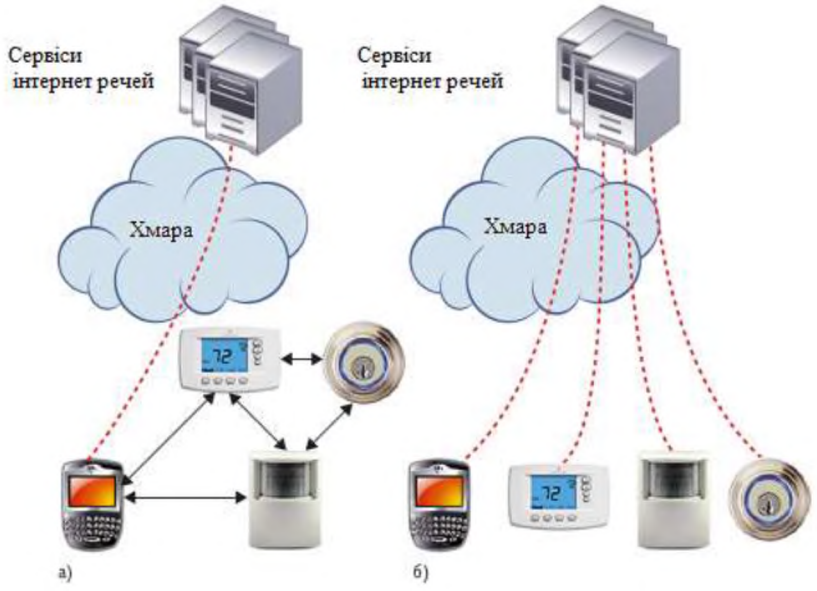
У Фізичній павутині люди і речі мають веб-сторінки, що надають інформацію та засоби для взаємодії з користувачем. Ідея застосування відкритих технологій WWW як посередника для зв'язку з фізичним світом не нова: точки доступу, маршрутизатори, сонячні панелі, лічильники електрики і кав'ярні мають свої веб-сторінки. Перспективи Інтернету речей обумовлені в тому числі широтою можливостей технологічного стека, на якому побудована мережа, і, звичайно, HTTP буде не єдиним протоколом для зв'язку з речами, так само як він не є єдиним

протоколом в Інтернеті. Є маса сценаріїв, коли у стандартних протоколів WWW немає необхідних властивостей, таких, наприклад, якими володіє Real-Time Streaming Protocol. Фізична павутина, подібно до традиційних пошукових систем, повертає текстові описи і відповідні посилання, але оскільки Інтернет речей складають об'єкти фізичного світу, то результати пошуку будуть впорядковані не тільки за допомогою традиційних алгоритмів ранжирування, але і за ступенем близькості до користувача. Вони можуть бути представлені як у вигляді списків, так і в формі географічних карт або поверхових планів. Якщо пошук по Фізичної павутині робиться всередині будинку, то будуть видаватися посилання на термостати, телевізори, аудіо-системи, маршрутизатори та лічильники електрики. Можна отримати посилання на інструкцію до мікрохвильової печі поряд з іншими посиланнями, що дозволяють управляти пристроями у вашому будинку або отримувати інформацію про них. Результатів буде багато, як і при звичайному веб-пошуку, але алгоритми ранжування добре справляються з тим, щоб першими видавати найбільш релевантні результати. Веб-сторінки відмінно підходять для людино-машиної взаємодії, але Інтернет речей у багатьох випадках буде застосовуватися для міжмашинної взаємодії. Формати даних, регламентовані в рамках ініціатив на зразок Schema.org, дозволяють браузерам і хмарним сервісам витягувати відомості про події, організації, людей, місця, продукти і т.п.. Використання структурованих даних також може сприяти єдності призначених для користувача інтерфейсів на різних пристроях: наприклад, користувачеві не доведеться кожного разу заново з'ясовувати, як встановлювати час на черговому побутовому приладі.

Технології загального доступу (Інтернет-технології), в тому числі HTML, Ajax, HTTPS і OpenID, як і структуровані дані, можна застосувати й в Інтернеті речей, але інтернет немає ефективного механізму з'ясування місцезнаходження об'єктів у фізичному світі, і дієвим тут представляється підхід, що складається у

використанні радіомаячків, що транслюють на малу відстань адреси URI за допомогою сигналу дуже низької потужності. Однією з перших до цієї ідеї звернулася компанія HP в рамках проекту Cooltown, в якому URI транслювалися за допомогою інфрачервоних маячків, а тепер малопотужний маячок можна реалізувати на базі BLE - він міг би періодично надсилати короткі пакети-оголошення з URI. Усунути розрив між фізичним і віртуальним світом можна було б, забезпечивши всі предмети радіо-маячками, які широкомовно транслювали б URI поряд з іншою інформацією, допомагаючи виявити себе. Здавалося б, на це знадобляться колосальні кошти, але радіо-маячки постійно дешевшають, і схоже, що всі майбутні «розумні» предмети будуть ними забезпечені. URI-маячки ближнього радіусу дії роблять більш реальною можливість взаємодії з Фізичної павутиною і можуть застосовуватися ширше, ніж NFC і QR-коди

Сьогодні більшість сервісів в WWW засновані на централізованому підході, але не можна стверджувати, що він же підійде і для Інтернету речей. На рис 1.4 порівнюються централізований і розподілений підходи - хмарні сервіси відповідають першому з них.



а - однорангова схема зі шлюзом для виходу на сервіс Інтернету речей;

б - централізований сервіс Інтернету речей.

Рисунок 1.4 – Варіанти реалізації Інтернету речей:

Так як Інтернет речей об'єднає безліч пристроїв з обмеженими мережевими можливостями, децентралізований принцип підключення може стати більш поширеним.

У світі комп'ютерів напрям змінювався не раз: централізований мейнфрейм, децентралізовані персональні комп'ютери з автономними додатками і, нарешті, централізовані хмарні сервіси, до яких тяжіє галузь перш за все тому що ними простіше керувати. У числі їх переваг називаються економія за рахунок масштабу при побудові ЦОД, автоматичне резервне копіювання всіх даних і фізичні засоби безпеки. Однак сучасні клієнтські пристрої відрізняються великою потужністю і гнучкістю: ноутбуки сьогодні оснащуються високопродуктивними багатоядерними процесорами, а смартфони мають потужні мікропроцесори. Можна користуватися нескладними клієнтськими пристроями, які зв'язуються з потужними хмарними сервісами або повномасштабними локальними додатками, яким цілком вистачає локальних ресурсів. Вибір залежить від того, на які компроміси між затримкою, безпекою, приватністю даних і вартістю згоден користувач. Якщо допустимі затримки при взаємодії і перебої в зв'язку, то більш привабливою моделлю будуть хмари. До того ж у зв'язку із зростанням кількості шкідливих програм на домашніх комп'ютерах хмарна схема представляється більш перспективною З огляду на це, для архітектури Інтернету речей було б доцільним реєструвати кожен пристрій в хмарному сервісі і зв'язуватися тільки з ним. Взаємодіючи з таким сервісом, користувачі або машини могли б визначати стан пристрою або управляти його поведінкою.

Хмарна модель - проста і зрозуміла, але не завжди доцільно надавати малопотужним пристроям повноцінні засоби зв'язку з Інтернетом. Устаткування, здатне безпосередньо з'єднуватися з Інтернетом, вимагає адаптера кабельної мережі Ethernet, приймача Wi-Fi або стільникового модему, що збільшує його вартість і витрати енергії. Доцільніше було б задіяти один апаратний міст, що підтримує Wi-Fi і здатний зв'язуватися з більш простими периферійними пристроями Інтернету речей. Можливість такого зв'язку сьогодні забезпечується цілим рядом стандартів, включаючи ZigBee, LoWPAN, Bluetooth Classic і BLE,

причому явного фаворита серед них немає. Як і у випадку радіоміток, шанси виграти в сегменті малопотужних пристроїв є у BLE, але на це піде якийсь час.

В Інтернеті речей, на відміну від традиційного Інтернету, можна з користю задіяти близькість пристроїв. Корисні не тільки знання про присутність об'єктів поблизу – пристрої, що знаходяться поруч можуть з'єднуватися один з одним, виконуючи завдання, які не під силу якомусь одному з них. Комп'ютер, який виявляє поблизу пристрій з екраном більшої площі, може з'єднатися з ним за бездротовою технологією при відтворенні фільму. Комп'ютер може також використовувати компонент, якого у нього немає: датчик, камеру або мишу, - якщо знайде його у сусідньому пристрої. Спільне використання ресурсів пристроїв, що знаходяться поблизу, стане важливим аспектом Інтернету речей, але для цього потрібні засоби, що дозволяють пристроям знаходити один одного, переконатися в можливості довіряти один одному і встановити з'єднання. Крім того, знадобляться підтримувані на обох сторонах з'єднання стандарти на протоколи і формати даних. Таку стандартизацію сьогодні важко провести, оскільки в Інтернеті зараз набагато більше комерційних підприємств, ніж на початку розвитку мережі. Традиційний Інтернет працює на чітко окресленому наборі стандартних протоколів, тоді як світ Інтернету речей більш розпливчастий - перспективи багатьох стандартів і рішень неясні. Наприклад, асоціація Digital Living Network Alliance працює над тим, щоб електроніка різних виробників могла виявляти один одного і ділитися контентом і сервісами. Однак сьогодні технології DLNA вбудовані лише в невелику кількість виробів з підтримкою мережевих функцій.

Для вирішення проблеми затримки при взаємодії з хмарними сервісами робляться спроби комбінувати різні ідеї. В рамках одного з підходів, що отримав назву «периферійні обчислення» (edge computing), частину хмарної обробки переносять ближче до пристроїв, яким потрібен мінімальний час відгуку, тим самим зменшуються кількість транзитних ділянок мережі і, відповідно, затримка. Приклад периферійних обчислень - концепція «хмарок» (cloudlet), що передбачає можливість швидко організувати у фіксованій інфраструктурі сервіси реального часу, якими смартфони і інші мобільні пристрої можуть користуватися через Wi-

Fi. Такі сервіси знаходяться всього в одній транзитній ділянці мережі від мобільних пристроїв. Це досягається за допомогою віртуальної машини, що працює на потужній локальній робочій станції, на яку динамічно доставляється спеціалізоване ПЗ і сервіси для поточного завдання. Коли воно виконано, ресурси вивільняються, дозволяючи створити в хмарці один або кілька нових екземплярів віртуальної машини. Хмарка, таким чином, дає змогу пристроям Інтернету речей в режимі реального часу взаємодіяти з хмароподібний сервісами, які діють далеко від ЦОД. Практично будь-який новий продукт, що відноситься до Інтернету речей, супроводжується мобільним додатком для управління - додатки для смартфонів є найбільш практичним на сьогодні способом обміну інформацією з «розумними» пристроями, а мобільні додатки давно стали нормою, і люди розраховують на те, що зможуть користуватися додатками для управління пристроями з Інтернету речей. Однак у міру розвитку ландшафту Інтернету речей можуть виникнути певні проблеми. Сьогодні на смартфонах зазвичай встановлено по кілька десятків додатків, але коли пристроїв будуть мільйони, почнуться складності. У найближчому майбутньому ми почнемо щодня проходити повз тисячі розумних пристроїв, що підтримують можливість електронної взаємодії, і встановлювати окремий додаток для роботи з кожним з них немислимо. До того ж доведеться постійно видаляти додатки під час встановлення нових, оскільки буде багато пристроїв, взаємодія з якими відбувається лише одноразово. Замість кількох великих додатків, що використовуються повсякденно для багатьох завдань, довелося б тримати величезну кількість додатків для малих пристроїв, що надають мінімальні інтерактивні можливості, - наприклад, для вимикачів. У багатьох випадках інтерактивних функцій взагалі не буде - може лише надаватися невеликий обсяг інформації - скажімо, відомості про те, коли до зупинки підійде наступний автобус. Через величезну кількість потенційних застосувань Інтернету речей відповідні додатки будуть сильно поступатися по функціональності сьогоднішнім програмами для смартфонів, що було б нераціонально, з огляду на потужність останніх. Для створення Фізичної павутини Інтернету речей потрібно зробити так, щоб будь-який користувач з будь-яким смартфоном або планшетом міг,

підійшовши до будь-якого пристрою, почати взаємодіяти з ним, не потребуючи спеціального додатку. Для цього знадобиться розширення мережі, яке дозволить будь-якому розумному пристрою транслювати URI в ефір. Ця передача повинна відбуватися на коротку дистанцію, щоб будь-який розумний пристрій міг отримати перелік сусідніх і взаємодіяти з ними. Відповідний механізм виявлення повинен оптимально працювати як з нативними, так і з веб-додатками, ставши новою платформою, універсальною мовою спілкування, що дозволяє пристроям будь-якого типу надавати дані інших пристроїв і взаємодіяти з ними. Нинішня модель нативних додатків не пропонує засобів підтримки усього розмаїття сценаріїв застосування Інтернету речей - необхідно розширити мережу і інші традиційні системи, створивши можливість спрощення взаємодії пристроїв і зниження вимог до ресурсів.

Створення будь-яких фізичних виробів починається з простих прототипів, які доопрацьовуються на основі відгуків і даних про функціонування в реальному світі. IoT-рішення розробляти складніше, оскільки прототипи потрібні також для програмних і мережевих компонентів. Завдання ще більше ускладнюється наявністю безлічі варіантів контексту, в якому IoT-рішення можуть використовуватися.

В галузі мобільних продуктів накопичений великий досвід вирішення таких проблем, оскільки мобільні середовища можуть приводити до зміни контексту, впливаючи на якість сервісу. В результаті створення мобільних рішень зазвичай починається з простих прототипів, які вдосконалюються в міру доопрацювання. Зазвичай цей процес включає наступні етапи:

– Створення простого інтерактивного web-додатку з використанням середовища розробки для швидкого визначення основних аспектів взаємодій. Всі виклики сервісу замінюються заглушками з використанням контенту, досить реалістичного для моделювання того, як інформація повинна відтворюватися і як її необхідно використовувати.

– Інтеграція простих аспектів сервісу, що дозволяє визначити, наскільки відповіді коректні і доречні контексти.

– Перехід до проектування інтерфейсів за межами функціональних прототипів з урахуванням методів взаємодій і потреб у навчанні та зворотного зв'язку.

– Продовження розробки та інтеграції можливостей аж до випуску продукту.

Ітеративний підхід до розробки прототипів IoT-рішення дозволяє гарантувати придатність сервісу до використання і реалізацію необхідних аспектів. Ітеративний підхід можна застосовувати не тільки для створення прототипів IoT-сервісу, а й для розробки прототипів використовуваного в рішенні апаратного забезпечення.

В якості основи IoT-рішення можна використовувати Raspberry Pi, Arduino, ESP8266 або інші подібні компоненти. Звичайно, бувають вагомі причини для використання конкретних процесорів або системних плат. Однак якщо створюють підключений до Інтернету садовий датчик, то можливостей ATMEGA328 (контролер Arduino) або ESP8266 вистачить для потреб проекту з запасом. Використовуючи готові компоненти, отримують переваги від масштабу вже існуючої інфраструктури, а також від наявності готових рішень типових проблем.

В даний час апаратні засоби легко піддаються інженерному аналізу. У IoT-рішенні цінним є сервіс, а не конкретні аватари. Тому дорогі спеціалізовані апаратні компоненти будуть підвищувати вартість аватарів без істотного підвищення цінності сервісу. Потрібно дотримуватися модульного підходу до розробки компонентів і плати контролера. Тоді, якщо буде потрібно перейти на більш потужну чи менш дорогу системну плату, не доведеться перебудовувати все з нуля.

Завершальним етапом у створенні прототипу є забезпечення запису в журнал відомостей про використання системи і які виникають помилки. Такий журнал дозволяє розуміти, що в системі працює, а що ні, і швидше виконувати налагодження, що є найважливішою вимогою на етапі створення прототипу. Журнал роботи можна зберігати аж до етапу виробництва.

Висновки до розділу 1

У першому розділі виконано аналіз сучасного стану галузі Інтернету речей - революційної концепції, що обіцяє серйозні зміни суспільству в цілому, зміни у повсякденному житті, надання звичайним користувачам абсолютно нового рівню комфорту.

Проаналізовано стандарти та перспективи їх розвитку в сфері IoT, розглянуто особливості рівнів архітектури та протоколів Інтернету речей. На основі моделі ризиків стандартної архітектури IoT виділено чотири зони безпеки: пристроїв, польових і хмарних шлюзів, служб. Поділ на зони являє собою універсальний спосіб сегментування рішення. Кожна з зон часто містить власні дані, а також пред'являє окремі вимоги до перевірки автентичності та авторизації. Проаналізовано різні варіанти реалізації Інтернету речей через однорангову систему та централізований сервіс.

На основі проведеного аналізу сформульовані загальне завдання, яке полягає у розробці моделі компоненти архітектури IaaS хмарного сервісу для систем Internet of Things. Загальне завдання розділено на три часткові задачі, дві з яких розглянуті у наступних розділах.

РОЗДІЛ 2
МОДЕЛІ АРХІТЕКТУР ХМАРНИХ СЕРВІСІВ

2.1 Платформа для реалізації інтернет речей Microsoft Azure

Розглянемо конкретний приклад еталонної архітектури Azure IoT, щоб продемонструвати принципи моделювання в середовищі IoT. Ця еталонна архітектура забезпечує керівництво для побудови безпечних і масштабованих рішень, орієнтованих на пристрої, для підключення пристроїв, проведення аналізу та інтеграції з системами Backend. Ці рішення можуть бути побудовані на публічних, приватних, і гібридних компонентах хмари Azure [15].

Мета Azure архітектури - забезпечити потік інформації між тимчасово або постійно підключеними пристроями і бізнес-активами (тобто не універсальними пристроями, такими як персональні комп'ютери, смартфони або планшети) і хмарними Backend -системами з метою аналізу, контролю та інтеграції бізнес-процесів. Рішення спеціально призначені для великомасштабних середовищ IoT з пристроями серійного виробництва з десятками тисяч одиниць промислових датчиків, що формують значні обсяги даних. Модель підходить і для сценаріїв де працює невелика кількість спеціальних пристроїв, але може бути більш вартісною, ніж рішення еквівалентно запуску одного веб-сайту. Архітектура прагне бути нейтральною стосовно конкретних галузей промисловості і сценаріїв використання, а також нейтральна до конкретних підходів до моделювання стану, метаданих і поведінки пристроїв. Однак, хоча архітектура є абстрактною, передбачається, що реалізація архітектури в конкретних рішеннях буде дуже конкретною і узгодженою з конкретними галузевими стандартами або конкретними доменними проектами. Еталонна архітектура забезпечує гнучкість для можливості компоновки і розширюваності, дозволяючи використовувати різні

типи технологій, вибір залежить від конкретних вимог рішення. На рис.2.1 показана концептуальна архітектура високого рівня.

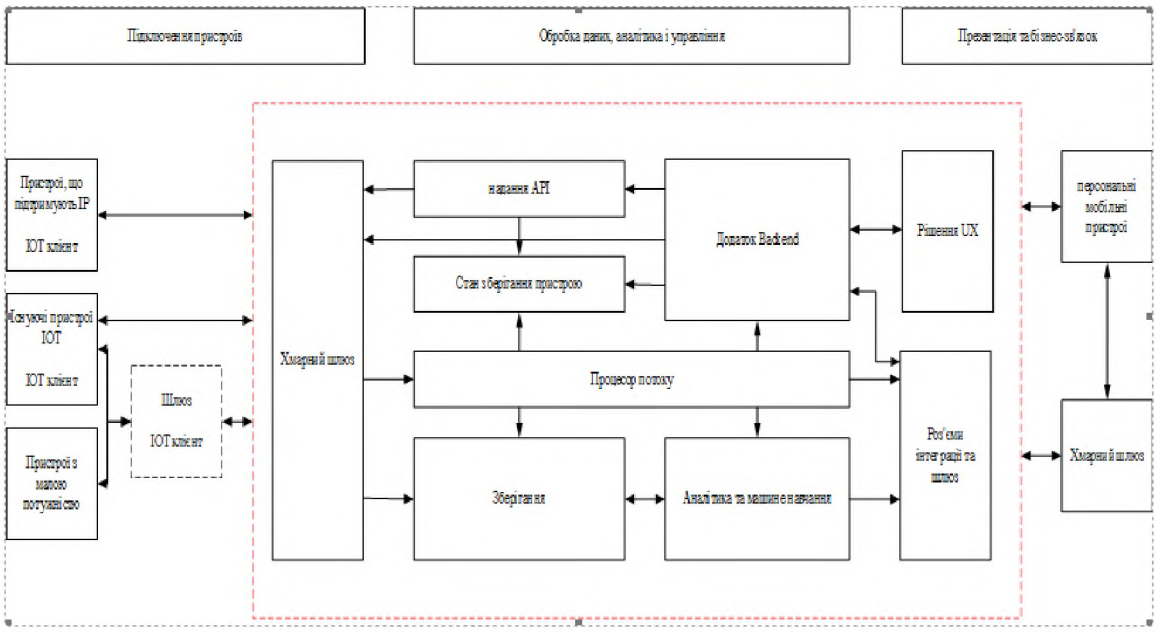


Рисунок 2.1 – Архітектура рішення IoT

Архітектура складається з основних платформ і компонентів рівня додатків для полегшення потреб в обробці в трьох основних областях типового рішення IoT [30]:

– підключення пристроїв,

– обробка даних, аналітика і управління,

– презентація та бізнес-зв'язок.

Пристрої можуть бути підключені безпосередньо або через шлюз; обидва варіанти можуть реалізувати логіку з різними рівнями можливостей обробки. Хмарний шлюз надає кінцеві точки для підключення пристроїв і полегшує двосторонній зв'язок з Backend-системою. Backend містить кілька компонентів для забезпечення реєстрації та виявлення пристроїв, збору даних, трансформації і аналітики, а також бізнес-логіки й візуалізації. Рівень інтеграції бізнесу та презентації відповідає за інтеграцію середовища IoT в бізнес-процеси

підприємства. Рішення IoT пов'язано з існуючими бізнес-додатками і стандартними програмними рішеннями через адаптери або можливості інтеграції корпоративних додатків (EAI) і бізнес-шлюзу (B2B).

Користувачі в сценаріях «бізнес-бізнес» або «бізнес-споживач» будуть взаємодіяти з рішенням IoT і спеціальним призначенням IoT через цей шар. Вони можуть використовувати IoT-рішення або призначені для користувача інтерфейси бізнес-системи, включаючи додатки на персональні мобільні пристрої, такі як смартфони і планшети. Microsoft надає основні можливості платформи, служби Azure IoT і компоненти рішення IoT, а також попередньо сконфігуровані рішення, які пропонують за замовчуванням склад цих елементів для загальних сценаріїв IoT.

2.2 Архітектурні принципи управління

Еталонна архітектура забезпечує ступінь деталізації компонентів, який дозволяє збирати безпечні, складні рішення, що підтримують екстремальний масштаб, і при цьому забезпечують максимальну гнучкість щодо можливих сценаріїв. Це мотивує такі принципи управління в різних областях архітектури.

– Неоднорідність. Пропонована еталонна модель повинна враховувати безліч різноманітних сценаріїв, середовищ, пристроїв, шаблонів обробки і стандартів. Він повинен мати можливість обробляти величезну апаратну і програмну гетерогенність.

– Безпека. Оскільки рішення IoT представляють собою потужний зв'язок між цифровим і фізичним світами, створення безпечних систем є необхідною основою. Ця еталонна модель передбачає заходи безпеки і забезпечення конфіденційності у всіх областях, включаючи ідентифікацію пристроїв і користувачів, аутентифікацію і авторизацію, захист даних на пристроях та під час передачі, а також стратегії атестації даних.

– Масштабованість розгортання. Пропонована архітектура повинна

підтримувати мільйони підключених пристроїв. Це повинно підтримати пілотні проекти, які починаються з невеликого числа пристроїв, а потім повинні бути масштабовані до суттєвих розмірів.

– Гнучкість. Неоднорідні потреби ринку IoT вимагають підтримання відкритого складу різних послуг і компонент. Еталонна модель побудована на композитному принципі, що дозволяє використовувати кілька точок розширення. Це дозволяє використовувати різні типи технологій сторонніх виробників для окремих концептуальних компонентів. Висококласна, керована подіями архітектура з брокерськими зв'язками є основою для слабо-зв’язаного складу послуг і модулів обробки.

2.3 Концепції даних

Розуміння концепцій даних є найважливішим першим кроком до створення системи, що забезпечує збір, аналіз та управління даними, орієнтованими на пристрої. Моделі часто описують схему для описових метаданих про пристрій, таких як його модель і серійний номер, схеми для даних, що формуються пристроєм, і схеми для параметрів конфігурації, які керують поведінкою пристроїв, а також операцій і параметрів для процесів управління, які може виконати пристрій, або подій, які він може спостерігати [3]. У різних галузях існує багато різних моделей пристроїв, і еталонна архітектура Інтернету речей займає нейтральну позицію, щоб підтримати більшість моделей та схемотехнічних рішень [16,17]. Архітектура використовує фундаментальну абстракцію потоків даних, де пристрої та моделі даних не прив‘язані до потоків, маршрутів або інформації про зберігання в компонентах базової платформи. На рівні рішення структуровані дані забезпечені моделями і схемами даних на той момент часу, коли вони формуються або споживаються компонентами.

Еталонна архітектура IoT використовує основне поняття потоків даних, які

складаються із записів даних та представляють потік даних через систему. Потоки даних не мають встановленого формату для вмісту записів тому що будь-яка структура буде залежати від типу даних, які транспортуються. Еталонна модель нейтральна щодо структури корисного навантаження і семантики даних. Вона не вказує і не приймає ніякої залежності від найменування, значення або типу будь-якого елементу або структури даних, які не потрібні для базової функції платформи. На рис.2.2 показаний логічний потік даних від пристрою IoT через точку проходу хмарного шлюзу і потік подій процесор в сховище [18,19].

Дані датчика пристрою складаються з цифрових сигналів-послідовностей спостережень «точка-час», виконаних за певний період, який вважається відкритим для сигналу. Цим сигналам може знадобитися деякий рівень попередньої обробки на пристрої, і в цьому випадку результат обробки відноситься до системи, а не до окремих точок вимірювання. Також варто враховувати, що попередньо оброблені цифрові сигнали можуть кодуватися і передаватися в багатьох різних форматах кодування. Звукові і світлові сигнали є прикладами даних датчиків, які вимагають аналізу на місці у багатьох сценаріях.

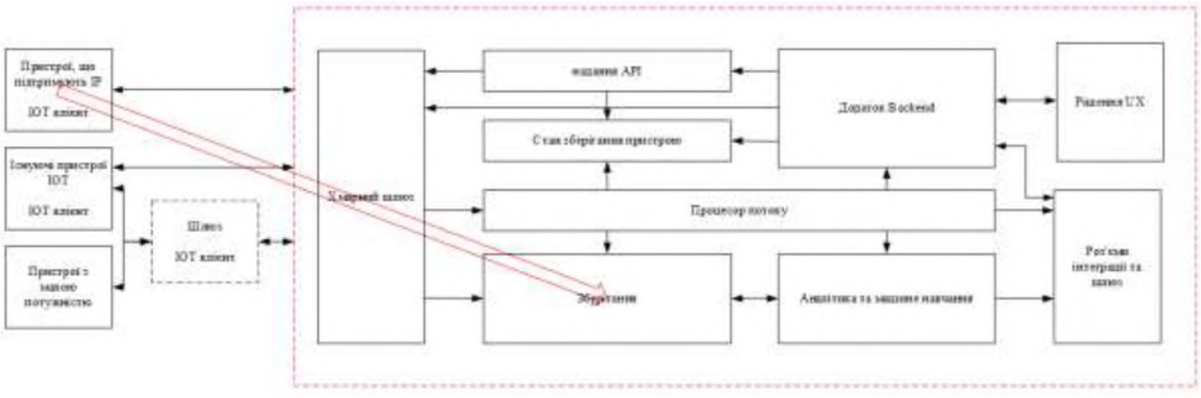


Рисунок 2.2 – Потік даних в еталонній архітектурі IoT

Еталонна архітектура не надає конкретних типів записів для станів попередження або дій в потоці даних. «Стан» параметра стає «останнім відомим

значенням», представленим останнім записом, що переносить цей конкретний параметр. Пристрої можуть відправляти записи з усіма визначеними параметрами для конкретного запису або в багатьох випадках тільки значення, які були змінені з моменту відправки останнього повідомлення. «Попередження» - це правило обробки, що ініціюється записом події, що відповідає визначеним умовам. «Дія» - це операція, ініційована у відповідь на отримання запису про подію з конкретними критеріями. Дії можуть бути визначені різними способами в залежності від сценарію. Потоки даних, що складаються із записів, доставляються на хмарний сервіс Azure з пристроїв Інтернету речей. Ці записи або повідомляють про стан точки даних (стан пристрою), або діють як попередження, на які необхідно вжити заходів. Будь-яка структура буде залежати від виду даних, які транспортуються, тому немає встановленого формату для змісту записів. Оскільки немає єдиної угоди про час або часові мітки, конкретне рішення може вибрати задана модель для обліку часових подій.

Так як дані потрапляють в хмарний сервіс Azure, важливо зрозуміти, як може змінюватися маршрут обробки даних. Обробник проміжного потоку буде приймати одну форму потоку даних, але може перетворювати її і формувати інший тип потоку вихідних даних. Крім того, потоки даних можуть припинятися і з'являтися в різних частинах системи в залежності від типу обробки даних. Ця проміжна обробка може виконуватися «на льоту», як частина потокових процесорів або передача даних за залишковими даними. Якщо декільком потокам даних необхідно протікати одночасно, наприклад, представляючи телеметрію для різних підкомпонентів пристрою, рекомендується відокремити їх, використовуючи дискримінатори в базовому протоколі додатку для кадрів корисного навантаження. Прикладом може служити використання поля «subject» в AMQP3 або, альтернативно, властивість додатку повідомлення (наприклад, «stream-id"). Потоки також можуть бути розділені з використанням поля конвенції навколо певного

корисного навантаження при використанні єдиного структурованого кодування даних. До еталонної архітектури можливо щоб додати додаткове поле, наприклад «_subject», до кожного запису корисного навантаження. Для кожного потоку даних висувається припущення, що всі записи мають сумісну структуру і семантику. Полю даних з даним «ім'я» в одному записі, має відповідати будь-якому іншому полю кореляції в запису потоку даних, як по типу, так і по семантиці. Повнота і сумісність залишаються у відповідальності розробників рішень і пристроїв. Іншим прикладом є ідентифікатор версії, коли записи версій дозволяють декільком паралельним потокам бути ізольованими за версією. Наприклад, використання угоди, в якій значення «_version» є тегом версії, може підтримувати загальний потік, але дозволяє його диференціювати. При наявності моделі управління версіями розробники рішень можуть відповідним чином вирішити потенційні конфлікти полів записів в термінах семантики або типу.

2.4 Взаємодія пристроїв

Еталонна модель застосовує принципіальні підходи, засновані на послузі Assisted Communication, для створення надійних зв’язків із пристроями, які потенційно розгортаються в ненадійному фізичному просторі. Це наступні принципи:

– Пристрої не приймають мережеві підключення. Всі з'єднання і маршрути встановлюються в одному шлюзі.

– Пристрої зазвичай підключаються або встановлюють маршрути тільки до відомих сервісних шлюзів, з яких вони проглядаються. В разі необхідності вони повинні передавати інформацію або отримувати команди зі служб, пристрої проглядаються за допомогою шлюзу, який піклується про маршрутизацію інформації у поток і гарантує, що команди приймаються тільки від авторизованих сторін до направлення їх на пристрій.

– Шлях зв'язку між пристроєм і сервісом або пристроєм і шлюзом забезпечується при транспортуванні на рівні протоколу додатків. Пристрої додатків не довіряють мережі канального рівня.

– Авторизація та автентифікація на системному рівні повинні ґрунтуватися на ідентифікаторах для кожного пристрою і отримувати доступ до облікових даних, дозволи повинні бути майже миттєво скасовані у разі зловживання пристроєм.

– Двонаправлений зв'язок для пристроїв, підключення може бути полегшено шляхом проведення команд і повідомлень на пристроях до тих пір, поки вони не підключяться до шлюзу.

– Дані корисного навантаження додатків можуть бути захищені окремо для захищеного транзиту через шлюзи до певного сервісу.

Пристрої можуть бути підключені прямо або через польовий шлюз. І пристрої, і польові шлюзи можуть реалізувати логіку інтелекту і аналітики. Це дозволяє зробити дві речі: агрегацію і скорочення вихідних даних телеметрії до перенесення на Backend і можливість прийняття рішень на місцевому рівні з правилами, які виконуються або на пристрої, або на Backend. На рис.2.3 представлено концептуальне уявлення різних варіантів підключення пристроїв для рішень IoT. Цифри на рисунку 2.3 відповідають чотирьом ключовим моделям підключення, які визначаються наступним чином:

1. Безпосереднє під'єднання пристрою до шлюзу хмари: Для пристроїв з підтримкою IP, які можуть встановлювати безпечні з'єднання через Інтернет.

2. Підключення через польовий шлюз: Для пристроїв, що використовують галузеві протоколи (наприклад, CoAP5, OPC), комунікаційні технології ближньої дії (такі як Bluetooth, ZigBee), а також для пристроїв з обмеженими ресурсами, які не здатні розміщувати стек TLS/SSL, або пристрої, не підключені до Інтернету. Ця опція також корисна при виконанні агрегації потоків і даних на польовому шлюзі перед переходом на хмару.

3. Підключення через призначений для користувача шлюз: пристрої, для яких потрібно перетворення протоколу або будь-яка форма обробки користувача, до досягнення хмари кінцева точка зв'язку шлюзу.

4. Можливість підключення через польовий шлюз і призначений для користувача шлюз: сценарії шлюзу можуть вимагати деякої адаптації або настройки протоколу на стороні хмари і, отже, можуть вибрати підключення до призначеного для користувача шлюзу, що працює в хмарі. Деякі сценарії вимагають інтеграції польових і хмарних шлюзів з використанням ізольованих мережевих тунелів, використовуючи технологію VPN або використовуючи сервіс ретрансляції на рівні додатку.

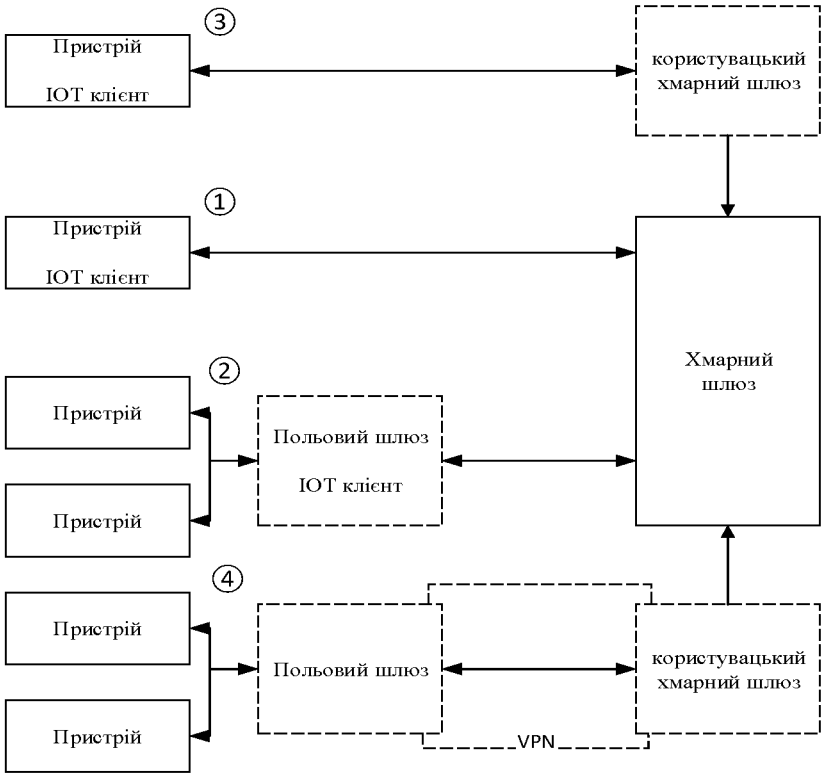


Рисунок 2.3 – Концептуальне уявлення підключенням пристроїв IoT

Прямий зв'язок між пристроєм і хмарою дозволяє здійснювати управління локальною мережею і потоками інформації або операцій, коли кілька пристроїв виконують якусь скоординовану дію. Чисто локальні взаємодії знаходяться за

межами масштабу цієї архітектури та охоплюються галузевими стандартами, такими як AllJoyn, UPnP / DLNA та ін. Важливо розуміти термінологію і ключові компоненти, що використовуються для опису підключення пристроїв Azure IoT еталонної архітектури.

Мета побудови еталонної архітектури полягає в тому, щоб забезпечити безпечний, ефективний і надійний зв'язок між хмарою та практично будь-якими пристроями і шлюзами. Це можна зробити як безпосередньо, так і через шлюзи, з тим щоб реалізувати практичні хмарні або хмарні комерційні рішення. Цільові пристрої - це бізнес-активи, від простих датчиків температури до складних заводських виробничих ліній з сотнями компонентів та датчиками всередині них.

Польовий шлюз являє собою спеціалізований пристрій або ПК загального призначення із програмним забезпеченням, який діє як засіб комунікації і, потенційно, як локальна система управління пристроями і концентратор обробки даних пристроїв. Польовий шлюз може виконувати локальну обробку і функції управління по відношенню до пристроїв; з іншого боку, він може фільтрувати або збирати телеметрію пристроїв і, таким чином, зменшувати обсяг даних, переданих на Backend хмари. Область дії польового шлюзу включає в себе сам шлюз і всі підключені до нього пристрої. Як випливає з назви, польові шлюзи діють поза спеціалізованими засобами обробки даних і зазвичай розміщуються разом з пристроями. Це об'єкт, пов'язаний з додатком, і мережеве з'єднання або сеансовий термінал. Наприклад, шлюз в цьому контексті може допомогти в налаштуванні пристроїв, фільтрації даних, дозуванні і агрегації, буферизації даних, перетворення протоколу і обробці правил подій. Пристрої NAT або брандмауери, навпаки, не кваліфікуються як польові шлюзи, оскільки вони не є явними з'єднаннями або сеансами зв'язку, а скоріше маршрутами (або заборонами) з'єднань або сеансів, виконаних через них.

Хмарний шлюз є частиною хмарної архітектури і забезпечує віддалений

зв'язок із пристроями. Деякі із цих пристроїв потенційно можуть перебувати на кількох різних сайтах. Хмарний шлюз доступний через мережу Інтернет або як віртуальна мережа (VPN), або як приватне мережеве підключення до центрів обробки даних Azure. Це необхідно щоб ізолювати і шлюз, і всі підключені до нього пристрої або польові шлюзи від іншого мережевого трафіку. Він, як правило, керує всіма аспектами комунікації, включаючи управління з'єднаннями на рівні транспортного протоколу, захист маршрутів даних, автентифікація та авторизація пристроїв по відношенню до системи. Він забезпечує підключення і обмеження пропускної здатності, а також збирає дані, які використовуються для виставлення рахунків, діагностики та інших завдань моніторингу. Потік даних з хмарного шлюзу використовує один або декілька протоколів обміну повідомленнями рівня додатків.

Щоб підтримувати архітектури, засновані на подіях, і загальні шаблони зв'язку, шлюз пропонує модель посередницького зв'язку. Телеметрія та інші повідомлення з пристроїв передаються до хмари, обмін повідомленнями здійснюється посередником. Трафік з датчиків пристрою до мережі зазвичай формується за допомогою «вхідного» шаблону. Навіть коли пристрій в знаходиться автономному режимі, повідомлення, відправлені на нього, будуть зберігатися в сховищі або черзі і доставляються, як тільки пристрій буде підключено до мережі. Через можливість періодичної генерації подій, надання суттєвого значення часу життя пакету (TTL) важливо, особливо для чутливих до часу команд, таких як «відкрити машину», «відкрити двері будинку», «завести автомобіль». Вхідний шаблон збереже повідомлення у надійному сховищі для заданої тривалості TTL, навіть якщо закінчиться термін дії повідомлень. Взаємодія через описані шаблони дозволяє розділити повноваження з хмарними компонентами щодо залежностей у часі виконання, швидкості обробки і контрактів на поведінку. Це також узгоджує провайдерів і споживачів при створенні ефективних, широкомасштабних,

керованих подіями рішень. В хмарній технології Azure роль хмарних шлюзів відіграє додаток Azure IoT Hub, який забезпечує безпечний двосторонній зв'язок з різними пристроями. Azure IoT Hub поєднує мільйони пристроїв і підтримує високорівневу телеметричну обробку в хмарному середовищі, а також управління і контроль трафіком пристрою. Azure IoT Hub забезпечує підтримку AMQP 1.0 з додатковою підтримкою WebSocket, MQTT, і власний HTTP 1.1 за протоколом TLS. Azure Event Hubs - це високопродуктивний сервіс для збору даних телеметрії від паралельних джерел з дуже високою пропускною здатністю. Крім IoT Hub, в сценаріях IoT можуть використовуватися концентратори подій для вторинних потоків телеметрії, або збору даних з інших джерел системи (наприклад, каналів погоди або потоків соціальних мереж). Концентратори подій не пропонують індивідуальну ідентифікацію або функції управління, тому можуть підходити тільки для додаткових потоків даних, які можуть бути співвіднесені з технологіями телеметрії пристроїв, але не як первинні шлюзи для підключення пристроїв. Azure Event Hubs забезпечує підтримку AMQP 1.0 з додатковою підтримкою WebSocket і власні HTTPS-протоколи. Підтримка додаткових протоколів AMQP, MQTT і HTTP може бути реалізована з використанням шлюзу протоколу адаптаційної моделі.

Призначений для користувача шлюз хмари Azure забезпечує адаптацію протоколів та деяку форму обробки даних користувача перед завантаженням їх до хмарного сховища. Він може включати відповідну реалізацію протоколу, необхідну пристроям і шлюзам переадресацію повідомлень до хмари для подальшої обробки і передачі команд і управління від проміжного шлюзу назад до пристроїв. Крім того, така обробка, як перетворення повідомлень або стиснення/декомпресія також може бути реалізована як частина користувацького шлюзу. Шлюзи користувача допомагають підключати різні пристрої з загальними або різними вимогами і нормалізувати трафік на вході до хмари.

Протокол шлюзу Azure IoT представляє собою платформу з відкритим вихідним кодом для шлюзів користувача. Шлюз протоколу Azure IoT полегшує широкомасштабний двосторонній зв'язок між пристроями і концентратором Azure IoT. Він включає в себе адаптер протоколу для MQTT, який демонструє методи для реалізації користувальницьких протоколів і дозволяє налаштовувати поведінку протоколу MQTT, якщо це необхідно. Протокол шлюзу також дозволяє додаткову обробку, таку як автентифікація, перетворення повідомлень, стиснення/декомпресія або шифрування/дешифрування. Клієнт хмарного IoT для зв'язку з пристроями або польовими шлюзами повинен використовувати захищені канали до кінцевих точок.

У IoT-системах є три ключових шаблони для підключення клієнтів:

– Пряме підключення до пристрою програмного рівня;

– Можливість підключення через агентів;

– Використання клієнтських компонентів, інтегрованих в прикладному / програмному рівні пристрою або шлюзу.

У другому випадку зв'язок з кінцевою точкою хмарного шлюзу спочатку ініціюється у пристрої (або польовому шлюзі), використовуючи необхідні протоколи. Агент - це програмний компонент, встановлений на пристрої (або польовому шлюзі), який виконує дії від його імені за допомогою іншої програми або керуючого компонента. Як правило, агенти є закритими системами, обмеженими можливостями для набору підтримуваних платформ. переносимістю інших платформ або налаштуваннями і розширеннями. Компоненти клієнта надають набір можливостей, які можуть бути інтегровані в програмний код, що виконується на пристрої, щоб спростити підключення до мережі. Вони зазвичай надаються у вигляді бібліотек або SDK, які можуть бути пов'язані або скомпільовані в програмний рівень пристрою. У порівнянні з агентами клієнтські компоненти вимагають інтеграції в програмне забезпечення пристрою, але вони

забезпечують максимальну гнучкість для розширюваності. SDK-пристрої Azure IoT представляють собою набір клієнтських компонентів, які можуть використовуватися на пристроях або шлюзах для спрощення підключення до Azure IoT Hub. SDK пристрою можна використовувати для реалізації клієнта IoT, який полегшує підключення до хмари. Бібліотеки дозволяють підключати гетерогенний діапазон пристроїв і польових шлюзів до Azure-based IoT. Вони спрощують спільні завдання підключення, абстрагують деталі основних протоколів і шаблони обробки повідомлень. Бібліотеки можуть використовуватися безпосередньо в додатку пристрою або для створення окремого агента, що працює на пристрої, який встановлює зв'язок з хмарним шлюзом і полегшує обмін даними між пристроєм і хмарою IoT.

Сховище ідентифікаторів пристроїв є важливим компонентом і зберігає всі ідентифікаційні дані пристрою. У ньому також зберігаються криптографічні ключі, що дозволяє виконати перевірку автентифікації клієнтського пристрою. Сховище ідентифікаційних даних не надає ніяких засобів індексування або пошуку, крім прямого пошуку ідентифікаторів пристроєм; ця функціональна роль покладається на реєстр. Сховища для ідентифікації та реєстри розділені з міркувань безпеки; пошук в реєстрі не повинен дозволяти розкривати криптографічний ключ. Крім того, обмеження сховища ідентифікаційних даних на мінімальний набір системних атрибутів допомагає забезпечити швидкість і гнучкість, з іншого боку, використання реєстру визначається вимогами рішення. Локальний шлюз використовує інформацію зі сховища ідентифікаторів для автентифікації пристроїв і управління ними. Сховище ідентифікаторів може міститися і в хмарному шлюзі, або, альтернативно, хмарний шлюз може використовувати окремі ідентифікатори пристроїв. При підключенні чи відключенні пристроїв сховище ідентифікаторів використовується для створення нових ідентифікаторів або для видалення пристроїв із системи. Пристрої також можуть бути включені або відключені. Коли

вони відключені, у них немає доступу до системи, але всі правила доступу, ключі та метадані залишаються на місці (рис. 2.4).

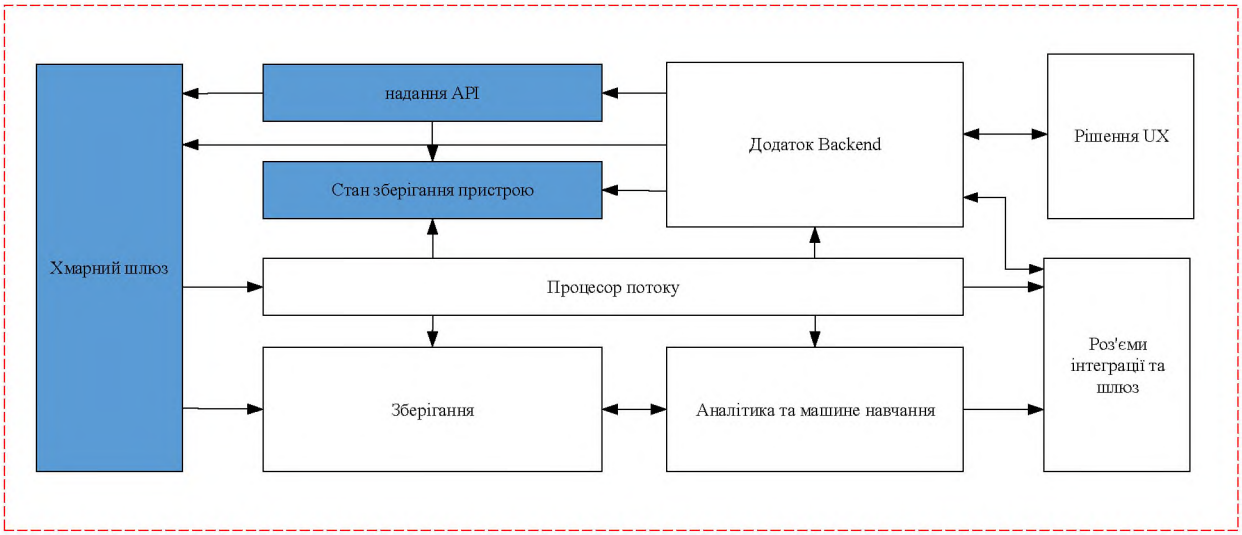


Рисунок 2.4 – Взаємодія в Azure IoT при наданні пристроям ідентифікаторів

Azure IoT Hub включає вбудоване сховище ідентифікаторів пристроїв, яке зберігає дані автентифікації для зареєстрованих пристроїв і надає для кожного пристрою облікові дані безпеки. Коли використовується користувальницький шлюз для хмарних обчислень, він також може використовувати сховище ідентифікаторів концентратора IoT і задійювати його механізми автентифікації і авторизації. У разі, якщо існують конкретні вимоги до рішення, які потребують спеціальної реалізації сховища ідентифікаторів, то його проектують як окремий компонент, який буде в першу чергу забезпечувати унікальність ідентифікатора, зберігати всю необхідну безпеку ключів для пристрою, і потенційно може містити статус «включено/вимкнено». Сховище ідентифікаторів має дозволяти доступ тільки до привілейованих частин системи в міру необхідності; призначені для користувача шлюзи будуть шукати необхідний матеріал автентифікації з цього сховища. Якщо не використовувати концентратор Azure IoT, зовнішні пристрої можуть бути через доповнення Azure DocumentDB, Azure Tables, Azure SQL

Database або сторонні рішення.

В Azure DocumentDB кожен пристрій представлено документом. На системному рівні ідентифікатору пристрою безпосередньо відповідає «id» документа.

В Azure Tables сховище ідентифікаторів замінюється таблицею. Ідентифікатор пристрою на системному рівні асоціюється з комбінацією PartitionKey і RowKey, які разом забезпечують унікальність. Всі інші властивості зберігаються в стовпцях; складні дані можуть зберігатися як JSON, якщо це необхідно.

В базі даних SQL сховище ідентифікаторів також асоціюється з таблицею, і кожен пристрій представлено записом у ній. Ідентифікатор пристрою на системному рівні асоціюється із стовпчиком первинного ключа кластерного індексу.

Реєстр пристроїв - це база даних «індексів», що використовується разом зі сховищем ідентифікаторів. Реєстр містить записи та довідкові дані, що відносяться до підключених пристроїв. Хоча сховище ідентифікаторів містить тільки атрибути системного управління і криптографічні ключі, в реєстрі будуть зберігатися інші пов'язані з пристроєм метадані та інформація для управління ним. Реєстр не нав'язує якусь конкретну схему або структуру схеми для метаданих пристрою, але можна визначити або вибрати деяку вертикальну стандартну схему для метаданих пристрою. Під час підготовки кожен пристрій зареєстровують в записах метаданих, які можуть містити структуровані метадані і можуть включати посилання на зовнішні експлуатаційні дані. Реєстр пристроїв є індексом, а сховище ідентифікаторів є списком ідентифікаторів пристроїв. Запис в сховищі ідентифікаторів визначає, чи є активним пристрій в системі. З позицій безпеки, в реєстрі пристроїв не повинно зберігатися ніяких ключів або іншої криптографічного інформації, що відноситься до пристрою. Різниця між

метаданими, що описують сам пристрій, і робочими даними, що відображають стан пристрою важлива, оскільки вона безпосередньо впливає на те, як дані реєстру можуть використовуватися, кешуватися і поширюватися у всій системі. Метадані - це зазвичай дані, що повільно змінюються, а оперативні дані навпаки, будуть швидко змінюватися. Зміни в реєстрі пристроїв повинні проводитися через API Provisioning.

Реєстр пристроїв, який зберігає описову інформацію про пристрій, повинен забезпечувати можливості індексу з метою забезпечення швидкого пошуку. Сховище реєстру може бути реалізовано поверх однієї з наступних технологій:

– Azure DocumentDB;

– База даних SQL;

– сторонні служби даних, доступні через Azure.

Надання ідентифікатору являє собою етап життєвого циклу пристрою, який повинен бути відомий системі. Provisioning API - це загальний зовнішній інтерфейс для того, щоб внести зміни в сховище ідентифікаторів пристроїв і реєстр пристроїв. Це абстрактний інтерфейс із звичайними подіями, і є реалізація цього абстрактного інтерфейсу для сховищ даних і реєстру. Робочі процеси більш високого рівня можуть реалізовувати один і той же або схожий інтерфейс і делегувати Provisioning API, якщо це необхідно в реалізації робочого процесу.

У процесі створення резервної копії рішення виконується обробка індивідуальних і масових запитів для реєстрації нових пристроїв та оновлення або видалення існуючих пристроїв. При такій обробці також оброблятимуть активацію, і, можливо, тимчасове призупинення чи відновлення доступу. Операції можуть включати взаємодію із зовнішніми системами, такими як M2M API мобільного оператора, для включення або відключення мережевих SIM-карт або бізнес-систем. Прикладом бізнес-систем можуть бути підтримка рішень, управління взаємовідносинами з клієнтами, варіанти технологій чи сценаріїв.

В якості альтернативи традиційним методам програмування Azure API Apps може використовуватися для реалізації Provisioning API. API Apps надає платформу для створення, розміщення і поширення API-інтерфейсів в хмарі і на місцях. Додатки API легко інтегруються з додатками Azure Logic, які можуть бути використані для реалізації та забезпечення робочого процесу через рішення IoT і зовнішні бізнес-системи. Інтерфейс ініціалізації є простим набором методів для управління життєвим циклом пристрою. Інтерфейс ініціалізації (API) повинен бути реалізований в якості основного API поверх сховищ даних і реєстру і, можливо, іншого внутрішнього рішення компонентів, якщо це необхідно. Він використовується не тільки з інтерфейсу рішення користувача (наприклад, порту адміністрування пристрою), але також служить інтерфейсом для робочого процесу вищого рівня, який також може взаємодіяти з зовнішніми об'єктами, такими як оператор мобільного оператора M2M API для управління SIM-картами або Backend бізнес-системою для активації платіжного облікового запису, пов'язаного з пристроєм. Ключі безпеки можуть генеруватися поза API і передаватися як параметри або можуть бути створені і призначені службами як частина виклику API ініціалізації. Створення маркера безпеки може бути виконано в Provisioning API з використанням необхідного ключа підпису. Токен, виданий пристрою буде обмежено за охопленням конкретної кінцевої точки. Дані, які повертаються операцією Register і ResetCredentials, містять необхідний захист і повинні бути передані на пристрій. В якості альтернативи, на пристрої можуть бути створені і передаватися назовні маркери безпеки. Для шлюзів користувача необхідні облікові дані можуть генеруватися ззовні і передаватися в API для зберігання, або API може бути розширений для створення ключів. Операційні дані, пов'язані з пристроями, зберігаються в сховищі станів пристрою. Сховище станів пристроїв відокремлене від реєстру пристроїв. Запис реєстру будь-якого пристрою буде зазвичай вказувати на сховище стану пристрою. Хоча зберігання вихідної інформації з пристрою в

сховище станів є необов'язковим архітектурним елементом. Проста реалізація може включати в себе запис потоку даних пристрою в сховище за замовчуванням, але сам потік даних буде видалений або змінений. Форма за замовчуванням для сховища станів пристрою полягає в тому, що воно зберігає два види інформації. Одна частина - це сирий потік вхідних подій з пристрою. Інша - це запис «останніх відомих значень», яка є проекцією останніх значень, отриманих з пристрою. Для мінімальної реалізації вхідні необроблені дані або прогнози повідомлень можуть бути збережені в Azure Blobs або Tables, що допомагає оптимізувати витрати. Вхідні повідомлення можуть бути відправлені або можуть бути змінені. Дані можуть бути використані безпосередньо зі сховища, або додатково переформуватися з використанням Azure Data Factory, що дозволить проводити вторинні перетворення, агрегації і переміщення даних за необхідності. На додаток до необроблених даних останні відомі значення від пристроїв зберігаються окремо як записи, і постійно перезаписуються. Ці записи можуть являти собою окремий об'єкт Azure blob або table, і можуть бути додані як окремий документ до реєстру пристроїв для швидких запитів. До цих або до окремого запису можуть бути додані агреговані або обчислені значення. У деяких випадках операційні дані будуть розділені в різних сховищах на основі шаблонів доступу. Наприклад, дані для аудиту (такі як зміни стану, історія операцій та команди), будуть доступні і оброблені незалежно інших експлуатаційних даних. Ще один приклад - архівні дані, переміщені в окреме сховище після певного періоду зберігання. Логічний поділ даних можливий як для груп пристроїв (наприклад, географічних), так і для для підкомпонентів складних пристроїв. Конструкція розбиття буде відповідати потребам доступу і операційним вимогам до даних. Вибір реалізації повинен проводитися для кожного, індивідуально керованого шаблонами робочого навантаження:

– Azure Data Lake - це розподілене сховище даних, що дозволяє зберігати

величезну кількість реляційних і нереляційних даних без перетворення або визначення схеми. Воно може обробляти великі обсяги невеликих записів, оптимізоване для пропускної здатності і добре підходить для потоків подій в сценаріях IoT.

– Сховище Azure Blob може використовуватися для зберігання необроблених даних пристрою. У ньому використовуються контейнери та імена blob. Воно являє собою певну структуру простору для зберігання, яка буде розроблена на основі вимог до рішення.

– Таблиці Azure: записи пристроїв і агреговані або обчислені значення можуть бути збережені в таблиці Azure. Вони підходять для ведення журналу, інформації аудиту або інших типів даних пристрою, до яких звертатимуться однорівневі сутності, без використання вторинних індексів. Стратегія розбиття і використання PartitionKey і RowKey закріплена за конкретними рішеннями і повинна бути розроблена виробником обладнання.

– Azure DocumentDB: набори даних, які можуть використовувати гнучкі можливості індексування схем і діагностики; багатий SQL інтерфейс. DocumentDB поєднує в собі управління без схем та no-SQL документи зі складними SQL-запитами.

– База даних SQL: для наборів даних, для яких потрібні можливості реляційного зберігання і запитів. База даних SQL також надає розширені функції для управління даними, захисту та безпеки і безперервності бізнесу.

– Azure Search: можна використовувати в сценаріях, які потребують повнотекстового пошуку, на основі вмісту даних і розширену поведінку пошуку.

– Azure Cache: додатково до розглянутих варіантів зберігання, стан пристрою також може зберігатися в Azure Cache для швидкого пошуку робочих даних пристрою.

– Сторонні рішення. Прикладами сторонніх служб є MongoDB, Cassandra,

Elastic Search, частина OpenTSDB і InfluxDB. Якщо використовується актор-модель, стан пристрою буде ґрунтуватися на параметрах які надає інфраструктура Actor.

Як було описано у попередньому розділі, кілька потоків даних можуть протікати одночасно. Крім того, шлюз забезпечує зв'язок і підтримує декількох споживачів, одні і ті ж дані можуть бути використані різними потоковими процесорами для різних цілей. Наприклад, потоковий процесор може прослуховувати тільки спеціальні типи подій, в той час як інший паралельно може виконувати складну обробку подій. Ці процесори можуть визначати маршрут даних або виконувати складні завдання з обробки подій, такі як агрегація даних, обчислення кореляції з довідковими даними, а також аналітичні завдання, такі як виявлення граничних меж або аномалій і генерації попереджень. На рис.2.5 показано місце потокових в архітектурі IoT.

Рисунок 2.5 – Потокові процесори в архітектурі IoT

До обробки даних від пристроїв IoT, процесор потоку може продовжувати оновлювати «останні відомі значення» для цих пристроїв. Конкретні агреговані або попередньо прораховані значення також можуть бути збережені в сховище станів пристроїв для зручності доступу, якщо це вимагає логіка рішення. У деяких

випадках пристрої можуть відправляти повідомлення, що вказують на зміни атрибутів їх метаданих. Як правило, вони відокремлені від загального потоку телеметрії. Процесор подій може «прослуховувати» ці повідомлення і оновлювати при необхідності записи в реєстрі пристроїв. Прикладом такої події може служити зміна конфігурації пристрою.

До потоку телеметрії пристрою може формуватися потік даних діагностичної інформації. Цей потік описує умови експлуатації пристрою, і швидше за все буде оброблятися окремо від загальної телеметрії. Складний механізм обробки подій може аналізувати пропущені події в режимі реального часу, порівнюючи кілька потоків або дані одного потоку зі збереженими значеннями і моделями. Це дозволяє виявляти аномалії, проводити розпізнавання по швидкісним вікнам і дає можливість ініціювати оповіщення при виникненні конкретної помилки або додати повідомлення до потоку. Створенні попередження відправляються в додатки, які обробляються відповідно до бізнес-правил або можуть безпосередньо ініціювати інтеграційний робочий процес з лінійними бізнес-системами.

Еталонна архітектура передбачає використання декількох процесорів подій, призначених для обробки одного або декількох потоків основних даних. Пристрої зазвичай передають трафік в декількох потоках даних, використовуючи дискримінатори в заголовках протоколу додатку (наприклад, функції обміну повідомленнями, такі як «stream-id" або "subject»), які дозволять маршрутизацію і обробку відповідним потоковим процесором. Пристрої або шлюзи можуть також виконувати деякі команди наприклад, класифікувати і розбивати дані на категорії для обробки холодного або гарячого маршруту. У хмарі Microsoft Azure запущена служба Stream Analytics. Компонент Apache Storm реалізований в Azure HDInsight. Процесори обробки подій користувача можуть поділити потік даних від точки входу концентратора Azure IoT до концентратора подій. Вони можуть виконувати попередню обробку потоку даних: агрегацію, виявлення кореляції з зовнішніми

даними, виявлення граничних меж і створення попереджень та іншу аналітику. Результати обробки можуть зберігатися в Azure Blobs, Azure Tables, Azure SQL, базах даних HDInsight HBase. Результати також можуть бути перенаправлені на концентратори подій або до черг шини обслуговування Azure для поширення в підсистемах, де далі виконується подальша обробка. В даний час Azure Stream Analytics дозволяє пересилати дані в Azure Blobs та Tables, SQL бази даних, DocumentDB, концентратори подій, черги і теми службової шини, а також Power BI. Azure Stream Analytics і Apache Storm дозволяють побудувати архітектуру Lambda в Azure, використовуючи такі засоби як Apache Kafka, Apache Cassandra і Apache Spark для подальшої обробки. Вхідний потік даних буде направлятися на різні рівні обробки через Apache Kafka і зберігатися в Cassandra, щоб з цими даними можна було виконувати подальшу обробку. Якщо дані знаходяться в «стані спокою», їх можна збирати, перетворювати і зберігати, використовуючи постійні або періодично доступні ресурси Azure Data Factory. Data Factory можуть виконувати довільні перетворення і перенесення даних спокою між блоками Azure, таблицями Azure і базами даних SQL в Azure PaaS, IaaS або на локальні сервери. Azure Stream Analytics використовує мову SQL для швидкої розробки, в той час як HDInsight Storm і процесори подій використовують програмні рішення, які вимагають розгортання коду в середовищі виконання. Через простоту використання Stream Analytics і прозорість синтаксису SQL забезпечуються повна гнучкість при обробці даних.

Stream Analytics безпосередньо підключається до Azure IoT Hub в якості потоку джерел і використовується як накопичувач даних і механізмів правил, що виконує такі задачі опрацювання:

– Вихідна телеметрія: завдання Stream Analytics, що використовує запит «SELECT * FROM <iot-hub input>», охоплює вхідні необроблені дані. Якщо виключити спеціальні типи подій, за допомогою WHERE можна відфільтрувати ці

повідомлення.

– В сховищі Blob додаткові перетворення або переміщення даних можуть бути реалізовані з використанням Azure IoT.

– Телеметричні перетворення. Крім того, додаткові завдання Stream Analytics можуть бути визначені для створення прогнозів через ряд операторів від простих фільтрів до складних кореляцій і пошуків. Це призводить до появи нових потоків, що можуть бути розглянуті як частина моделі, специфічної для конкретного додатка.

– Правила і попередження. Stream Analytics може використовуватися для виконання правил виявлення порогових значень і обмеженнь. Тим не менш, одного піку в вимірах може бути недостатньо, щоб сформувати попередження. Середнє значення порушення заданого порога виміру протягом певного періоду часу може бути більш підходящим способом формування попередження. Створене попередження буде виведено в чергу службової шини або на концентратор подій для обробки у пристрої, що зазвичай запускає відкат попередньої дії.

– Телеметрія для кожного пристрою. Процесор подій користувача приймаючи дані від концентратора Azure IoT, може використовуватися для виділення вхідного потоку даних в сховище станів для кожного пристрою. Використовуючи цей метод, події потоку даних для кожного пристрою записуються в окреме сховище станів пристроїв на основі сховища Azure Blob.

Альтернативний підхід полягає в зміні завдання Stream Analytics для необробленої телеметрії і налаштування його виходу на концентратор подій всередині потоку. Потім процесор подій для концентратора використовується для направлення та обробки вхідного потоку даних в сховище станів кожного пристрою. Перевага цього методу полягає в тому, що додатково до необроблених даних пристроїв, деякі значення можна отримати як прогнози з Stream Analytics, як середні значення, або інші відповідні атрибути пристрою. Загалом, навіть для

початкової телеметрії, що проходить через Stream Analytics, є переваги, які дозволяють легко модифікувати запит завдання для створення проекції потоку даних (наприклад, включати фільтрацію або кореляцію з довідковими даними). У дуже спрощеній формі реалізації, коли немає механізму правил і додаткових шляхів потоку даних, процесор подій може бути використаний для обробки вхідного потоку даних безпосередньо з приймального концентратора IoT. Використання Azure Stream Analytics або HDInsight в якості механізму управління даними і правилами дозволяють гнучко додавати нові вихідні прогнози і потоки даних за необхідності.

Інтерфейсне рішення (UX) зазвичай включає в себе веб-сайт, але також може включати веб-служби та API-інтерфейси з графічним інтерфейсом користувача у вигляді мобільного або комп'ютерного додатків. Рішення UX є частиною архітектури IoT, що реалізує доступ і візуалізацію даних пристрою, результати аналізу, виявлення пристроїв через реєстр, можливості управління, а також робочі процеси виділення ресурсів. У багатьох випадках кінцеві користувачі будуть повідомлені про попередження, умови тривоги або необхідні дії за допомогою push-повідомлень. Рішення UX може також забезпечувати або інтегруватися з інтерактивними панелями моніторингу, які є придатною формою візуалізації для сценаріїв IoT з великою сукупністю пристроїв. Рішення IoT часто включають в себе служби геолокації і геоінформації, і інтерфейс користувача повинен буде забезпечити відповідні можливості. Безпека є ключовою і рішення UX, яке забезпечує контроль над системою і пристроями повинні бути з належним контролем доступу, диференційованим за ролями користувачів в залежності від авторизації. Технологія Azure App Service - це керована платформа з потужними можливостями для створення веб-додатків і мобільних додатків для багатьох платформ і мобільних пристроїв. Веб-додатки та мобільні додатки дозволяють розробникам створювати, використовуючи такі мови, як .NET, Java, NodeJS, PHP

або Python. Крім того, API-інтерфейс Azure дозволяє легко переглядати і управляти API-інтерфейсами, що надає доступ до мобільних або веб-клієнтів.

Azure Notification Hubs дозволяє відправляти push-повідомлення на особисті мобільні пристрої (тобто смартфони і планшети). Воно підтримує платформи iOS, Android, Windows, абстрагуючи деталі іншої платформи системи повідомлень (PNS). За допомогою єдиного виклику API, повідомлення може налаштовуватися як для окремого користувача, так і для сегмента аудиторії з великою кількістю користувачів. Power BI - це хмарний сервіс, який забезпечує простий спосіб створювати багаті інтерактивні інформаційні панелі для візуалізації та аналізу. Power BI також пропонує інтерактивні панелі моніторингу, які дозволяють користувачам моніторити зміни даних і показників. Power BI включає власні додатки для настільних і мобільних пристроїв. Іншою підходящою технологією візуалізації IoT є Bing Maps. API-інтерфейси Bing Maps включають елементи управління і служби та карти, які можна використовувати для включення Bing Maps в додатки і веб-сайти. Крім інтерактивних і статичних карт, API забезпечують доступ до геопросторових функцій, таких як геокодування, дані маршруту і трафіку, а також джерела просторових даних, які можна використовувати для зберігання і запитів до даних з просторовим компонентом, таких як розташування пристроїв. Веб-додатки та мобільні додатки можуть бути інтегровані з Azure Active Directory (AAD) для автентифікації і авторизації. Додатки будуть використовувати функції управління ідентифікаторами користувачів в AAD і можуть легко забезпечити управління доступом на основі ролей для функціональності програми. У багатьох випадках встановлюються логічні зв'язки між пристроями IoT і користувачами (або між групами пристроїв і групами користувачів). Дозволи та управління доступом на основі ролей можуть управлятися як частина матриці асоціацій між ідентифікаторами пристроїв (що підтримується в сховищі ідентифікаторів пристроїв) і ідентифікаторів користувачів, керованих AAD.

Важливими частинами Backend додатків є керуюча логіка рішення, виявлення і візуалізація пристроїв, управління станом пристроїв і виконання команд, а також частина функцій управління пристроєм, яка управляє життєвим циклом, дозволяє розподілити конфігурацію і оновлення програмного забезпечення і дистанційне керування.

На відміну від традиційних бізнес-систем, бізнес-логіка рішення IoT може поширюватися на різні компоненти системи. Частина управління пристроєм рішення зазвичай буде використовувати обчислювальні вузли, тоді як частина аналітики рішення буде в значній мірі реалізована безпосередньо всередині відповідних можливостей аналітики. У деяких випадках прості рішення можуть не мати незалежно розгорнутого керованого додатку «бізнес-логіка», але основна логіка може існувати як вираження правил, розміщені всередині потоку процесори, деякі можливості аналітики або як частина бізнес-процесів і компонентів з'єднувача.

Рішення IoT зв'язується з існуючими бізнес-додатками і стандартними програмними рішеннями через бізнес роз'єми або можливості шлюзу. Кінцеві користувачі в сценаріях будуть взаємодіяти з такими пристроями та через нього можна використовувати спеціальні пристрої IoT. У багатьох випадках кінцеві користувачі будуть використовувати персональні мобільні пристрої для доступу до функціональності. Ці персональні мобільні пристрої концептуально відрізняються від пристроїв IoT, хоча в деяких випадках можуть асоціаціюватися з ними або зі шлюзом між мобільним пристроєм кінцевого користувача і пристроями IoT. Наприклад, в сценарії домашньої автоматизації, мобільний смартфон може діяти як польовий шлюз, підключатися до пристроїв IoT і полегшувати зв'язок із хмарним сервісом. З точки зору авторизації асоціація між кінцевими користувачами, персональними мобільними пристроями і пристроями IoT буде управлятися в Backend IoT.

Висновки до розділу 2

У розділі розглянуто особливості моделей хмарних сервісів. На сьогоднішній день отримали розвиток приватні та публічні хмарні системи, але незабаром більшого поширення набудуть хмарні системи комбінованого типу. Переваги підходу хмарних технологій - доступність і можливість гнучкого масштабування, клієнтам не потрібно створювати і підтримувати власну обчислювальну інфраструктуру. Використання хмарних технологій в багатьох випадках дозволяє скоротити витрати.

Розглянуто особливості еталонної архітектури Azure IoT. Azure є першим великим постачальником хмарних служб, який зобов'язався за контрактом дотримуватися вимог та загального регламенту щодо захисту даних (GDPR). Крім того, ця платформа визнана найбільш надійною хмарою для державних організацій США і отримала рівень аккредітаціі FedRAMP High (високий) для 18 служб Azure. Azure IoT охоплює всі сфери, включаючи розробку додатків, забезпечення безпеки і управління, управління ідентифікаторами пристроїв, а також роботу з платформою даних. Вона дозволяє мінімізувати ризики і знизити вартість гібридного хмарного середовища. При роботі з такими сценаріями Інтернету речей як віддалений моніторинг та прогнозне обслуговування, застосовуються попередньо налаштовані рішення в Azure IoT Suite. Майже половина сертифікованих в Azure пристроїв Інтернету речей працюють під управлінням Linux, Android і інших технологій с відкритим кодом.

РОЗДІЛ 3
МОДЕЛЬ АРХІТЕКТУРИ СЕРВІСУ IAAS ДЛЯ ПІДТРИМКИ СИСТЕМИ INTERNET OF THINGS

3.1 Модель IaaS інфраструктури

Оскільки завдання організації надання ресурсів як послуг в хмарному середовищі, побудови їх моделей, розрахунок показників якості обслуговування є досить місткими, пропонується розглянути їх послідовно за принципом накопичення знань від простого до складного.

Згідно [20], базовими послугами, що надаються хмарними провайдерами, є IaaS, PaaS і SaaS, інші типи послуг є похідними (табл. 3.1).

Таблиця 3.1 – Базові і похідні типи хмарних послуг

Категорія хмарних послуг	Тип хмарного ресурсу, що надається		
	Інфраструктура	Платформа	Додатки
Infrastructure as a Service	X		
Network as a Service	X	X	X
Platform as a Service		X	
Software as a Service			X

На рис.3.1 наведена хмарна IoT архітектура з послугою IaaS [8].

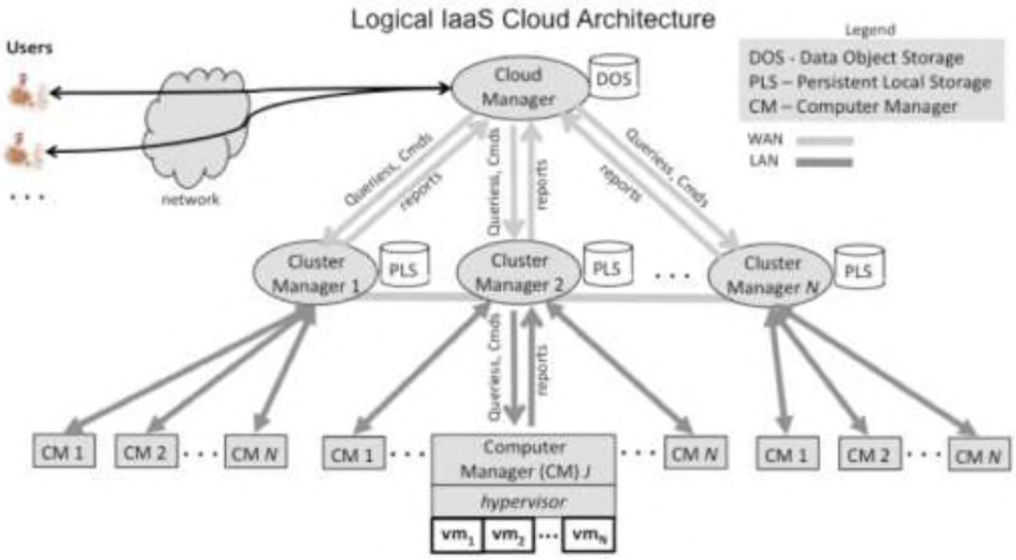


Рисунок 3.1. IaaS хмарна IoT архітектура

Після надходження заявки на обслуговування, вона потрапляє у RPDE -чергу (Resource provisioning decision engine). В RPDE можливі два види відмов в обслуговуванні заявки - при переповнюванні черги вхідних заявок системи-вирішувача і при недостатності фізичних, віртуальних і буферних ресурсів безпосередньої ланки обслуговування заявки.

3.2 Вибір показників якості обслуговування, доступності і енергоефективності функціонування хмарної інфраструктури

Згідно [26,29], в процесі надання хмарних послуг беруть участь сторони споживача і провайдера послуги (а також ряд посередників) між якими полягає угода про якість послуг (SLA), що надаються (рис.3.2).

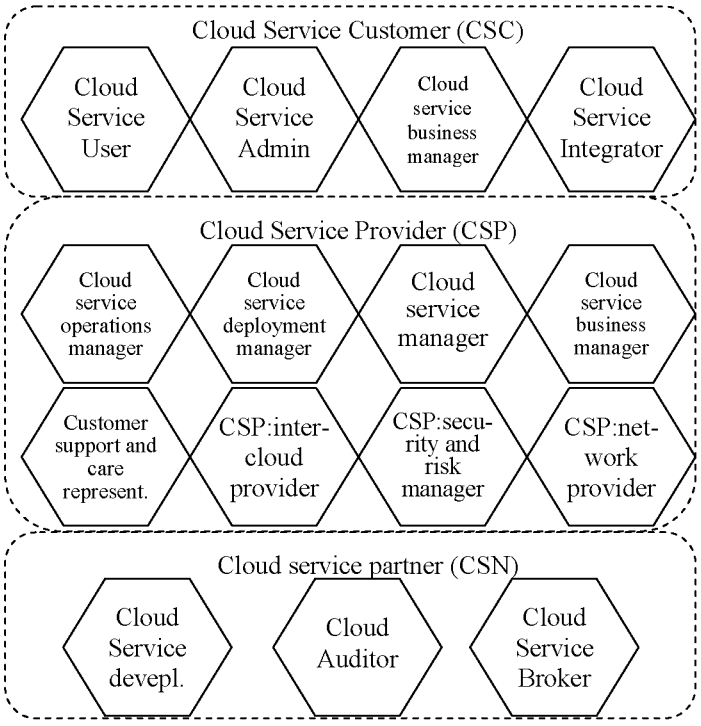


Рисунок 3.2 – Ролі учасників процесу надання хмарних послуг

У угоді SLA регламентуються параметри якості надання хмарних послуг з боку провайдера за умови, що споживач не вийде за межі вхідних параметрів інтенсивності заявок на надання послуг. Параметрами якості, що регламентуються

в SLA виступають :

– затримка до відгуку хмарного сервера;

– доступність хмарного сервера.

У свою чергу, доступність є комплексним показником, обумовленим зайнятістю ресурсів їх надійністю і безпекою. Як правило, провайдер не завжди розкриває причини простоїв і недоступності устаткування; а користувач оцінює загальний час доступності серверів не вдаючись до подробиць їх функціонування. Але це не означає, що при оцінюванні параметрів доступності можна ігнорувати ті або інші причини простоїв (якщо це не прописано окремими пунктами угоди SLA).

Окрім виконання пунктів угоди SLA, провайдери хмарних послуг використовують додаткові критерії оцінювання ефективності функціонування хмарної інфраструктури. Основними з них є:

– мінімізація енергоспоживання устаткуванням хмарної інфраструктури;

– оптимізація вартості витрат на обслуговування хмарної інфраструктури, на енергетику, на забезпечення високої доступності і безпеки; а також тарифного плану хмарних послуг з метою максимізації прибутку.

3.3 Побудова моделі системи-вирішувача

Ця модель побудована з використанням апарату моделювання безперервних марковських ланцюгів (Continuous Time Markov Chain - CTMC) і представлена на рис.3.3.

Ця модель є модифікованою системою масового обслуговування (СМО) типу M/M/1/N, де:

M - марковський (з експоненціальним розподілом і інтенсивністю λ) потік вхідних заявок;

M - марковські (з експоненціальним розподілом і інтенсивностями обслуговування δ_h, δ_w, δ_c) потоки обслуговування заявок - пошуку ресурсів в гарячому/теплому/холодному пулах відповідно;

1 – кількість обслуговуючих приладів (один вирішувач RPDE);

N - довжина черги (буфера) заявок.

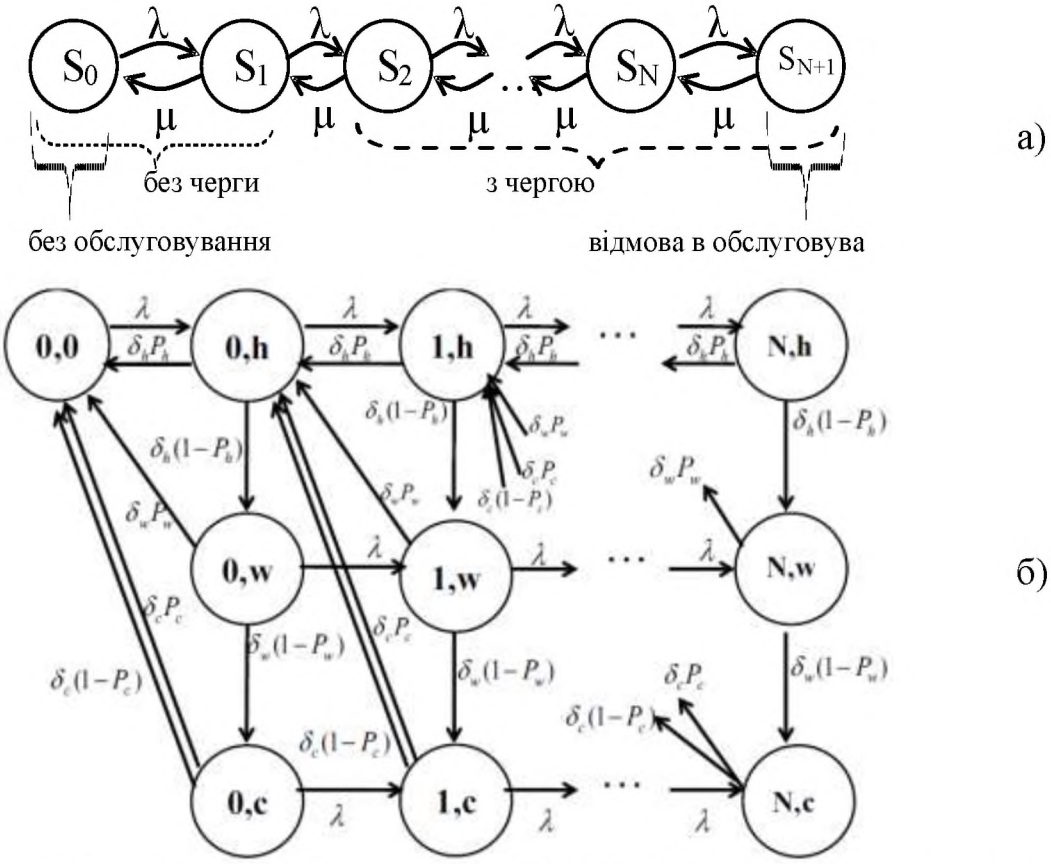


Рисунок 3.3 – Класична модель СМО M/M/1/N (а) і подмодель функціонування системи-вирішувача RPDE (б), для стану $S_{i,j}$: i - кількість заявок в черзі, j - тип пулу, [h, w, c]

Система-вирішувач функціонує таким чином. Після надходження першої заявки система переходить в стан пошуку вільного ресурсу в гарячому пулі серверів ($S_{0,h}$). З урахуванням вірогідності обслуговування заявки в гарячому пулі P_h, вирішувач або знайде необхідний ресурс і повернеться в початковий стан $S_{0,0}$ з інтенсивністю $P_h\delta_h$, або перейде з інтенсивністю $(1 - P_h)\delta_h$ в стан пошуку ресурсу в теплому пулі ($S_{0,w}$). Аналогічно, з інтенсивністю $P_w\delta_w$, вирішувач може знайти вільний ресурс в теплому пулі і повернутися в початковий стан, а може і не знайти його і з інтенсивністю $(1 - P_w)\delta_w$ перейти в стан пошуку ресурсу в холодному пулі ($S_{0,c}$). Із стану S_0, с вирішувач повертається в початковий стан і при знаходженні вільного ресурсу в холодному пулі (з інтенсивністю $P_c\delta_c$) так і за відсутності

вільного ресурсу (з інтенсивністю $(1 - P_c)\delta_c$), але в останньому випадку заявка буде не обслужена через нестачу ресурсів в хмарі.

Як і в класичній моделі СМО, коли вирішувач здійснює пошук вільних ресурсів (стани $S_{0,\,h}$, $S_{0,\,w}$ і $S_{0,\,c}$), на його вхід може поступити нова заявка з інтенсивністю λ. Ця заявка буде поміщена в чергу (буфер) і система відповідно перейде в стани $S_{1,\,h}$, $S_{1,\,w}$ і $S_{1,\,c}$. Процес обслуговування (пошуку ресурсу) в цих станах нічим не відрізняється від процесу, описаного вище.

Система вирішувач має обмежену довжину черги заявок, тому, досягнувши станів $S_{N,\,h}$, $S_{N,\,w}$ і $S_{N,\,c}$ наступні заявки отримають відмову в обслуговуванні, обумовлену обмеженою довжиною черги/буфера RPDE.

Вхідними параметрами моделі є:

λ - інтенсивність вхідного потоку заявок на обслуговування IaaS -услуги;

$1/\delta_h$, $1/\delta_w$, $1/\delta_c$ - середній час пошуку вільних ресурсів в гарячому/теплом/холодному пулах серверів відповідно;

P_h, P_w, P_c - вірогідність успішного обслуговування заявки в гарячому/теплому/холодному пулах серверів відповідно.

Кількісні значення вхідних параметрів представлені в табл. 3.2.

Таблиця 3.2 – Значення вхідних параметрів моделі RPDE

Вхідний параметр	**Клас інтенсивності заявки**	**Значення**
λ	Низька	10 – 500 заявок в годину
	Середня	500 – 1500 заявок в годину
	Висока	Більше 1500 заявок в годину
$1/\delta_h$, $1/\delta_w$, $1/\delta_c$		1 – 5 секунд
Ph, Pw, Pc		0 – 1
N_{RPDE}		0 – 1000

Після побудови розміченого графа моделі, згідно з правилом Колмогорова-Чепмена складається система диференціальних рівнянь (СДУ).

Для ряду завдань (дослідження надійності і готовності, стійкості системи до зміни вхідних параметрів) знадобиться рішення СДУ, як правило, чисельним методом. Інші завдання (у їх числі і визначення показників якості обслуговування) вимагають знаходження значень стаціонарної вірогідності Pi. Для таких завдань

СДУ перетворять в систему лінійних рівнянь (СЛУ) з урахуванням початкових станів об'єкту моделювання. Перехід від СДУ до СЛУ рекомендується виконувати шляхом прирівнювання похідних dPi/dt до нуля, а вірогідності Pi вважаються константами (не залежать від часу t). Рішення отриманої таким чином СЛУ знаходиться з використанням умови нормування, інакше, система матиме нескінченну безліч рішень.

Після визначення вірогідності Pi знаходження системи в кожному із станів, можна визначити ряд вихідних показників моделі. До них відносяться:

P_{block} - вірогідність відмови в обслуговуванні із-за переповнювання черги/буфера RPDE;

P_{drop} - вірогідність відмови в обслуговуванні із-за нестачі ресурсів в пулах фізичних серверів;

P_{reject} - сумарна вірогідність відмови в обслуговуванні;

$E[N_{RPDE}]$ - середня кількість заявок, які знаходяться в черзі вирішувача;

$E[T_{q\_dec}]$ - середній час знаходження прийнятої заявки в черзі вирішувача;

$E[T_{decision}]$ - середній час пошуку вільного ресурсу у вирішувачі.

Розрахункові формули для визначення вихідних показників зведені в табл. 3.3.

Таблиця 3.3 – Вихідні показники моделі RPDE

Показник	**Прив'язка до вірогідності** Pi	**Розрахункова формула**
P_{block}	Крайні праві стани графа стану графа $P_{N,h}$ $P_{N,w}$ $P_{N,c}$	$\sum_{j\in h,w,c} P_{N,j}$
P_{drop}	Нижні стани графа $P_{0,c}$ $P_{1,c}$. $P_{N,c}$ з урахуванням інтенсивності відмови $(1 - P_c)\delta_c$	$\frac{\delta_c(1-P_c)}{\lambda} \times \sum_{i=0}^{N} P_{i,c}$
P_{reject}	Pblock + Pdrop	
$E[N_{RPDE}]$	Сума творів вірогідності станів $P_{i,j}$ на i - кількість заявок в черзі, j={h, w, c}	$\sum_{i=1}^{N} i \cdot \left(P_{i,h} + P_{i,w} + P_{i,c}\right)$
$E[T_{q\_dec}]$	$E[N_{RPDE}] \Big/ \lambda\left(1-P_{reject}\right)$	
$E[T_{decision}]$	$\frac{E[N_{decision}]}{\lambda\left(1-P_{reject}\right)} = \frac{1-P_{0,0}}{\lambda\left(1-P_{reject}\right)}$	

Модель системи-вирішувача реалізована у вигляді функціонального блоку середовища Matlab, код якого представлено у додатку А. Результати моделювання представлені у графічному вигляді у додатку Б.

3.4 Оцінювання економічної ефективності функціонування елемента хмарної IoT інфраструктури

У загальному випадку величина економічного ефекту E визначається як різниця між доходами від використання системи та витратами $G_П$, пов'язаними з її функціонуванням [51]. Таким чином, для формула для визначення економічної ефективності системи вирішувача наступна:

$$E = p_{об} \cdot \lambda \cdot c \cdot T - G_П,$$

де c – середній економічний ефект, отриманий під час обслуговування однієї заявки;

T – інтервал часу, на якому досліджується функціонування системи;

$G_П$ – величина втрат у системі.

У системі обслуговування з відмовами втрати включають:

– витрати на експлуатацію системи,

– збитки через наявність необслужених заявок в системі,

– збитки через вимушені простої компонент хмарної архітектури.

Таким чином, для системи з відмовами величина втрат визначається за формулою [52]:

$$G_П = \left(q_K \cdot N + q_B \cdot p_B \cdot \lambda + q_{Пр} \cdot N_B\right) \cdot T,$$

де q_K – вартість експлуатації однієї компоненти хмарної архітектури за одиницю часу;

q_B – вартість збитків внаслідок необслуговування заявок за одиницю часу;

$q_{Пр}$ – вартість одиниці часу простою компоненти хмарної архітектури;

N_B – середня кількість вільних комопнент.

У системі обслуговування з обмеженою чергою заявок втрати пов'язані з:

– витратами на експлуатацію;

– збитками від ненадходження заявок;

– збитками від простою приладів;

– збитками від втрати заявок.

Таким чином, величина $G_П$ втрат у системі обслуговування з обмеженою чергою буде обчислюватися за формулою [53]:

$$G_П = \left(q_K \cdot N + q_O \cdot K_O + q_B \cdot p_B \cdot \lambda + q_{Пр} \cdot N_B\right) \cdot T ,$$

де q_o – вартість збитків, пов'язаних з простоєм черги за одиницю часу.

Для системи обслуговування з очікуванням у необмеженій черзі втрати пов'язані з:

– збитками від простою черги;

– збитками від простою приладів.

Таким чином, величина $G_П$ втрат у системи обслуговування з очікуванням у необмеженій черзі буде обчислюватися за формулою:

$$G_П = \left(q_K \cdot N + q_O \cdot K_O + q_{Пр} \cdot N_B\right) \cdot T .$$

Можна зазначити, що як критерій економічної ефективності цільової функції можуть бути обрані і інші показники: товарообіг, дохід, прибуток. Тоді оптимальне значення параметрів системи обслуговування знаходиться вже при максимізації зазначених показників.

Зокрема, для значень вхідних даних [53], а саме: інтенсивність обслуговування заявок RPDE 25,3 заявки/год. Інтенсивність заявок в хмарний сервіс в годину пік 500 заявок/ год, в інший час = 85 заявок/ год. Кількість місць у черзі було взято рівним 2. У ході комп'ютерного експерименту було отримано такі результати.

Для випадку, коли інтенсивність заявок в хмарний сервіс 500 заявок /год, оптимальна кількість віртуальних машин, розрахована за моделлю, дорівнює 29. Імовірність відмови в обслуговуванні становитиме 0,5%, середня довжина черги – 0,02 заявки, середнє завантаження системи вирішувача– 70%. Для випадку, коли

інтенсивність заявок в хмарний сервіс 85 заявок/год, оптимальна кількість віртуальних машин, розрахована за моделлю, дорівнює 10. Імовірність відмови в обслуговуванні становитиме 1,5%, середня довжина черги – 0,05 заявок, середнє завантаження системи вирішувача – 66%.

Аналіз результатів показує, що система в повному обсязі обслуговує всі заявки , що до неї надходять. Проте, попри мінімальні витрати, вона обох випадках завантажена в неповному обсязі.

Спроба визначити статистичні показники при повному завантаженні системи вирішувача показала наступні результати. Повне завантаження системи при інтенсивності вхідних заявок 500 заявок/год можливе у разі використання 19 віртуальних машин. Але в цьому випадку ймовірність відмови в обслуговуванні зросте до 14%, що викличе підвищення витрат, пов'язаних із перебуванням заявок у черзі та з відходом необслуговуваних клієнтів. Аналогічна картина спостерігається і за інтенсивності вхідних заявок 85 заявок/час, коли повне завантаження системи відбувається для n=6. Тут ймовірність відмови становитиме вже 22%.

Висновки до розділу 3

У третьому розділі було розглянуто моделі розгортання інформаційних ресурсів IoT на хмарних платформах IaaS. Виділено елементи IaaS хмарної інфраструктури: систему-вирішувач, пули гарячих, теплих та холодних серверів та безпосередні виконавці – віртуальні машини. Як один з показників якості обслуговування розглянуто середній час до безпосереднього обслуговування заявки в хмарі. Розглянуто ролі учасників процесу надання хмарних послуг та особливості домовленостей між ними.

Розроблено та досліджено модель системи-вирішувача RPDE як системи масового обслуговування з відмовами та обмеженою чергою. Підвищено точність математичної моделі шляхом розв‘язку системи диференційних рівнянь

Колмогорова-Чепмена за допомогою вирішувача ode15s до рівня 10^{-9}. Аналіз отриманих результатів моделювання компоненти хмарного сервісу показав, після зростання кількості заявок більше 1000 за годину починається рівень ймовірності відмови збільшується до 0,1; а середній час пошуку вільного ресурсу зменшується.

Визначено порядок розрахунку економічної ефективності функціонування системи вирішувача за показником втрат від необслуговування заявок та простоїв черги. Повне завантаження системи при інтенсивності вхідних заявок 500 заявок/год можливе у разі використання 19 віртуальних машин. Але в цьому випадку ймовірність відмови в обслуговуванні зросте до 14%, що викличе підвищення витрат, пов'язаних із перебуванням заявок у черзі та з відходом необслуговуваних клієнтів.

ВИСНОВКИ

У роботі була поставлена і вирішена актуальна наукова задача розробки та дослідження моделі компоненти архітектури IaaS хмарного сервісу (системи вирішувача RPDE) для систем Internet of Things.

У ході виконання кваліфікаційної роботи було проведено дослідження стану розвитку галузі в сфері Internet of Things, та перспективи їх розвитку в сфері IoT, розглянуто особливості рівнів архітектури та протоколів Інтернету речей. На основі роботи можна сформулювати наступні висновки.

1. Для оцінювання ризиків стандартної архітектури IoT потрібно виділити чотири зони безпеки: пристроїв, польових і хмарних шлюзів, служб. Поділ на зони являє собою універсальний спосіб сегментування рішення. Кожна з зон часто містить власні дані, а також пред'являє окремі вимоги до перевірки автентичності та авторизації.

2. Для реалізації Інтернету речей можливі варіанти побудови через однорангову систему та централізований сервіс.

3. На сьогодні поширене використання приватних та публічних хмарних систем, але незабаром більшого поширення набудуть хмарні системи комбінованого типу. Переваги підходу хмарних технологій - доступність і можливість гнучкого масштабування, клієнтам не потрібно створювати і підтримувати власну обчислювальну інфраструктуру. Використання хмарних технологій в багатьох випадках дозволяє скоротити витрати.

4. Архітектура Azure IoT забезпечує вимоги та загального регламенту щодо захисту даних (GDPR). Крім того, ця платформа визнана найбільш надійною хмарою для державних організацій США і отримала рівень аккредітаціі FedRAMP High (високий) для 18 служб Azure. Azure IoT охоплює всі сфери, включаючи розробку додатків, забезпечення безпеки і управління, управління ідентифікаторами пристроїв, а також роботу з платформою даних. Вона дозволяє мінімізувати ризики і знизити вартість гібридного хмарного середовища

5. Елементи IaaS хмарної інфраструктури, параметри яких доцільно

моделювати за допомогою систем масового обслуговування: система-вирішувач, пули гарячих, теплих та холодних серверів та безпосередні виконавці – віртуальні машини. Як один з показників якості обслуговування розглянуто середній час до безпосереднього обслуговування заявки в хмарі.

6. За рахунок розв‘язку системи диференційних рівнянь Колмогорова-Чепмена вирішувачем ode15s у розробленій моделі системи-вирішувача RPDE підвищено точність до 10^{-9}. Аналіз отриманих результатів моделювання компоненти хмарного сервісу показав, після зростання кількості заявок більше 1000 за годину починається рівень ймовірності відмови збільшується до 0,1; а середній час пошуку вільного ресурсу зменшується.

7. Визначено порядок розрахунку економічної ефективності функціонування системи вирішувача за показником втрат від необслуговування заявок та простоїв черги. Повне завантаження системи при інтенсивності вхідних заявок 500 заявок/год можливе у разі використання 19 віртуальних машин. Але в цьому випадку ймовірність відмови в обслуговуванні зросте до 14%, що викличе підвищення витрат, пов'язаних із перебуванням заявок у черзі та з відходом необслуговуваних клієнтів.

Таким чином, поставлені задачі розв‘язано у повному обсязі. Напрямком подальших досліджень є розробка та дослідження імітаційних моделей хмарної архітектури при непуасонівських потоках впливів на систему.