Міністерство освіти і науки України
Державний університет «Житомирська політехніка»

І. В. Пулеко, А. А. Єфіменко

АРХІТЕКТУРА ТА ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ

Навчальний посібник
для бакалаврів**,** які навчаються за спеціальністю
123 - Комп’ютерна інженерія

Рекомендовано Вченою радою
Державного університету «Житомирська політехніка»

Житомир
2022

УДК 004.738.5: 621.391
П88

Рекомендовано до друку вченою радою
Державного університету «Житомирська політехніка»
(протокол № 15 від 19 грудня 2022 року)

Рецензенти:

Фриз С. П. – професор кафедри телекомунікацій та радіотехніки Житомирського військового інституту імені С. П. Корольова, д.т.н., проф.
Гнатюк С.О. – заступник декана факультету кібезбезпеки, комп'ютерної та програмної інженерії Національного авіаційного університету, д.т.н., проф.
Воротніков В. В. – професор кафедри комп'ютерної інженерії та кібербезпеки Державного університету «Житомирська політехніка», д.т.н., доц.
.

Пулеко І. В. Єфіменко А. А.
П88 Архітектура та технології Інтернету речей: навч. посіб. / І.В. Пулеко, А.А. Єфіменко. – Електронні дані. – Житомир: Державний університет «Житомирська політехніка», 2022. – 234 с.

У навчальному посібнику викладено загальні відомості про Інтернет речей (IoT), розглянуті засоби ідентифікації, вимірювань, виконавчі пристрої та живлення в IoT. Викладені основні технології передачі даних, хмарні, туманні технології та аналіз даних в IoT. У загальному вигляді розглянуті проблеми безпеки IoT.

Навчальний посібник призначено для бакалаврів, які навчаються за спеціальністю 123 Комп'ютерна інженерія.

ISBN 978-966-683-616-1 **УДК 004.738.5: 621.391**

Навчальне видання

ПУЛЕКО Ігор Васильович

ЄФІМЕНКО Андрій Анатолійович

Навчальний посібник

Електронне видання

Комп'ютерний дизайн та верстка: Пулеко І.В., Єфіменко А.А.

Державний університет «Житомирська політехніка»

вул. Чуднівська, 103, м. Житомир, 10005

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

IoT – Інтернет речей (Internet of Things);

LAN – локальна мережа (Local Area Network) – мережа, що зазвичай покриває відносно невелику територію або невелику групу будівель (будинок, офіс, фірму);

M2M – Технологія між машинної взаємодії (Machine to machine);

MAC – адреса управління доступом до середовища (Media Access Control);

MAN – міська мережа (Metropolitan Area Network) – об'єднує окремих користувачів та локальні мережі в межах міста, є мережею за розмірами більшою, ніж LAN, але меншою, ніж WAN;

MEMS – Технологія виготовлення мікроелектромеханічних систем (Micro-Electro-Mechanical Systems);

PAN – персональна мережа (Personal Area Network) - це мережа, побудована "навколо" людини;

WAN – глобальна мережа (Wide Area Network) – пов'язує користувачів та мережі, розосереджені на відстані сотень та тисяч кілометрів;

WPAN – бездротова персональна мережа (Wireless Personal Area Network);

WSN – бездротова сенсорна мережа (Wireless Sensor Network);

АЦП – аналого-цифровий перетворювач;

ЕРС – електрорушійна сила;

МСЕ-Т – Міжнародний союз електрозв'язку;

ПНВЕС – перетворювач неелектричної величини в електричний сигнал.

ВСТУП

Досить важко знайти в наш час людину, яка нічого не чула про Internet of Things (IoT, Інтернет речей). У засобах масової інформації говорять про майбутню технологічну революцію, яка торкнеться всіх і змінить наше життя в тій же мірі, як поява телебачення, персональних комп'ютерів або мережі Інтернет. На відповідні запити в Інтернеті висвітиться велика кількість посилань на інформаційні джерела. Однак більшість з цих інформаційних джерел є філософського, концептуального, або науково популярного направлення і з погляду фахівців у технічній галузі є малоінформативними. У той же час, визнаючи перспективи розвитку «Інтернету речей» у багатьох світових вищих навчальних закладах, і в Україні також, відкрилися освітні програми чи спеціалізації пов'язані з IoT.

В основу навчального посібника покладено роботу [1] та курси лекцій, що читаються авторами у Державному університеті «Житомирська політехніка» навчальні дисципліни «Архітектура та технології IoT».

При формуванні змісту навчального посібника враховано, що ряд питань що стосуються 3 та 4 розділу студенти проходять на інших дисциплінах, таких як: «Комп'ютерні мережі», «Бездротові мережі» та «Хмарні технології», «Системи штучного інтелекту».

Видання є навчальним і автори не претендують на наукову новизну викладених у навчальному посібнику положень. Усі запозичені елементи мають відповідні посилання у тексті. Однак авторами була проведена кропітка робота по систематизації та адаптації навчального матеріалу та викладено ряд особистих думок, які можуть бути предметом наукової дискусії. Усі заперечення, виявлені недоліки та пропозиції щодо змісту підручника, прохання направляти на електрону пошту: pulekoigor@gmail.com.

1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНТЕРНЕТ РЕЧЕЙ

1.1. Концепція та поняття Інтернету речей (IoT)

КОНЦЕ́ПЦІЯ (від лат. conceptio – осягати, сприймати) – система поглядів, понять про ті чи інші явища або процеси, спосіб їхнього розуміння, тлумачення; основна ідея будь-якої теорії, головний задум; ідея чи план нового, оригінального розуміння; конструктивний принцип художньої, технічної та інших видів діяльності [5, 6].

У широкому сенсі, згідно з найбільш поширеним формулюванням, **Інтернет речей (англ. Internet of Things, IoT)** - концепція обчислювальної мережі фізичних предметів («речей»), оснащених вбудованими технологіями для взаємодії один з одним або з зовнішнім середовищем [7], яка розглядає організацію таких мереж як явище, здатне перебудувати економічні та суспільні процеси, що виключає з частини дій і операцій необхідність участі людини [8].

Ця концепція була сформульована в 1999 році як осмислення перспектив широкого застосування засобів радіочастотної ідентифікації для взаємодії фізичних предметів між собою і з зовнішнім оточенням. Наповнення концепції «Інтернету речей» різноманітним технологічним змістом і впровадження практичних рішень для її реалізації починаючи з 2010-х років вважається стійкою тенденцією в інформаційних технологіях [8], перш за все, завдяки повсюдного поширення бездротових мереж, появи хмарних обчислень, розвитку технологій міжмашинної взаємодії (Machine to machine (M2M)), початку активного переходу на IPv6 і освоєння програмно-конфігуруємих мереж.

Також, концепція передбачає, що Інтернет речей здатний серйозно вплинути на розвиток сучасного суспільства, оскільки дозволить багатьом процесам відбуватися без участі людини.

Для практичної реалізації концепції всі навколишні предмети і пристрої (домашні прилади і посуд, одяг, продукти, автомобілі, промислове обладнання та ін.) повинні бути забезпечені мініатюрними ідентифікаційними і сенсорними (датчиками) пристроями. Тоді при наявності необхідних каналів зв'язку з ними можна не тільки відслідковувати ці об'єкти і їх параметри в просторі і в часі, але і керувати ними, а також впроваджувати інформацію про них в загальну «розумну планету».

У загальному вигляді сутність Інтернету речей може бути зведена у вигляді символічної формули:

Фізичні об'єкти + ідентифікатор, сенсори, контролери, виконавчі пристрої + Інтернет = IoT

Тобто, будь-яка система IoT складається з набору фізичних об'єктів, кожен з яких:

містить ідентифікатор, що однозначно ідентифікує фізичний об’єкт чи його сенсори;

містить сенсор (датчик), що вимірює будь-який фізичний параметр, і/або виконавчий механізм, що спрацьовує від будь-якого фізичного параметра;

містить мікроконтролер, що забезпечує збір даних з сенсора їх адаптацію для передачі по мережі та/або інтелектуальність;

має можливість комунікації через Інтернет або будь-якої іншої мережі.

Унікальним аспектом IoT, в порівнянні з іншими мережевими системами, очевидно є наявність множини фізичних речей і пристроїв, відмінних від обчислювальних пристроїв і пристроїв обробки даних.

Тобто, фактично Інтернет речей - це глобальна мережа сенсорів (датчиків), комп'ютерів (обчислювачів) і виконавчих пристроїв (актуаторів), що зв'язуються між собою з використанням інтернет протоколу IP (Internet Protocol) та здатна надавати послуги людству.

Вважається, що Інтернет речей ґрунтується на *трьох базових принципах* [2]. По-перше, повсюдно поширену комунікаційну інфраструктуру, по-друге, глобальну ідентифікацію кожного об'єкта і, по-третє, можливість кожного об'єкта відправляти і отримувати дані за допомогою персональної мережі або мережі Інтернет, до якої він підключений.

Фундаментальними характеристиками Інтернету речей є:

– *Взаємозв’язаність*. Всі пристрої взаємодіють через глобальну або локальну інфраструктуру інформаційного обміну.

– *Сервіси, орієнтовані на пристрої*. Інтернет речей здатний забезпечити семантичну узгодженість між фізичними об’єктами реального світу і їх інформаційним поданням у віртуальному просторі і об’єднати фізичні пристрої з урахуванням правил і обмежень.

– *Гетерогенність*. Пристрої в IoT неоднорідні за визначенням і можуть належати різним мережам і апаратним платформам, що не є перешкодою до взаємодії.

– *Динамічність*. Стан пристроїв змінюється постійно: включення і виключення, контекстна і технологічна інформація, включаючи місце розташування і швидкість. Кількість підключених пристроїв також може динамічно змінюватися.

– *Масштабність*. Кількість пристроїв, які будуть «спілкуватися» і отримувати керуючий вплив в десятки разів перевищить кількість вузлів в поточній мережі Інтернет. Очевидно, що кількість комунікацій, які можуть бути ініційовані пристроями, радикально перевищить можливе число з'єднань, ініціаторами яких виступають люди. Тому на перший план виходять питання інтерпретації даних, з метою їх подальшого застосування.

Найбільш важливими відмінностями Інтернету речей від існуючого Інтернету людей є:

- фокус на речах, а не на людину;
- істотно більше число підключених об'єктів;
- істотно менші розміри об'єктів і, як правило, невисокі швидкості передачі даних;
- фокус на зчитуванні і обробці інформації, а не на комунікаціях;
- необхідність створення нової інфраструктури і альтернативних стандартів.

Концепція мереж наступного покоління NGN передбачала можливість комунікацій людей (безпосередньо або через комп'ютери) в будь-який час і в будь-якій точці простору. Концепція Інтернету речей включає ще один напрямок – комунікація будь-яких пристроїв або речей (рис. 1.1).

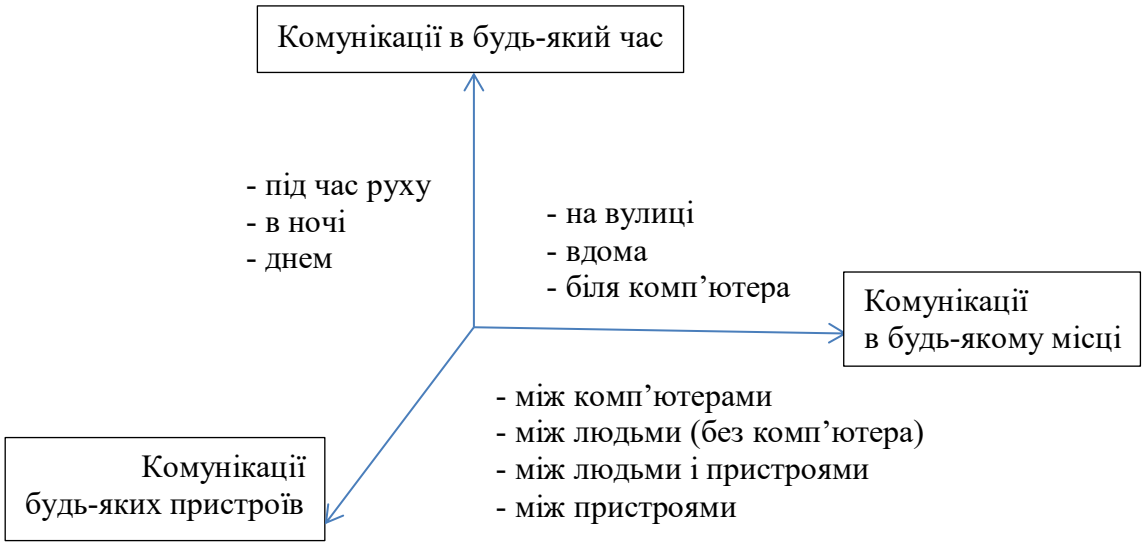


Рис. 1.1. Новий напрямок комунікацій, що реалізовується Інтернетом речей (джерело: MCE-T Y.2060)

1.2. Історія розвитку IoT

Концепція «Інтернету речей» є досить новою і її історія розвивалась фактично на протязі останніх трьох десятиліть.

Вважається, що першу в світі Інтернет-річ створив один з розробників протоколу TCP/IP Джон Ромки в 1990 році, коли він підключив до мережі свій тостер. Але тільки в 21 столітті в зв'язку з бурхливим розвитком інформаційно-комунікаційних технологій сформувалася концепція IoT і отримала своє практичне втілення.

Безпосередньо концепція Internet of Things і термін для неї вперше були сформульовані засновником дослідницької групи Auto-ID (англ.) при Массачусетському технологічному інституті Кевіном Ештоном (англ. Kevin Ashton) [1] в 1999 році на презентації для керівництва компанії Procter & Gamble. У презентації розповідалося про те, як всеосяжне впровадження засобів радіочастотної ідентифікації (RFID) зможе видозмінити систему управління корпоративними системами поставок та дозволить порахувати і відстежити товари без людського втручання.

Далі у 2000 році компанія LG презентувала перший у світі холодильник LG Internet Digital DIOS з доступом до Інтернету [2]. У цьому ж році з’явилися і перші прояви розробленої компанією HP концепції всепроникної комп'ютеризації (Cooltown): HP Labs - система обчислювальних і комунікаційних технологій, які в поєднанні один з одним створюють підключення до Інтернету для людей, місць і об'єктів [3].

У 2004 році в науково-популярному журналі Scientific American була опублікована велика стаття [1], присвячена «Інтернету речей», що наочно показала можливості концепції в побутовому застосуванні. У статті наведена ілюстрація, що показує як побутові прилади (будильник, кондиціонер), домашні системи (система садового поливу, охоронна система, система освітлення), датчики (теплові, датчики освітленості і руху) і «речі» (наприклад, лікарські препарати, що забезпечені ідентифікаційної міткою) взаємодіють один з одним за допомогою комунікаційних мереж (інфрачервоних, бездротових, силових і слабкострумових мереж) і забезпечують повністю автоматичне виконання процесів (включають кавоварку, змінюють освітленість, нагадують про прийом ліків, підтримують температуру, забезпечують полив саду, дозволяють зберігати енергію і керувати нею споживанням). Подані варіанти домашньої автоматизації були не новими, але акцент в публікації робився на об'єднанні пристроїв і «речей» в єдину обчислювальну мережу, яка обслуговується Інтернет-протоколами.

У 2005 році спеціалізована установа ООН «Міжнародний союз електрозв'язку» випустив звіт, в якому вперше були сформульовані прогнози розвитку Інтернету речей [1].

Період з 2008 по 2009 рік вважається «справжнім народженням "Інтернету речей"», так як, за оцінками аналітиків корпорації Cisco (рис. 1.2), саме в цьому проміжку кількість пристроїв, підключених до глобальної мережі, перевищила чисельність населення Землі [9], тим самим «Інтернет людей» став« Інтернетом речей ».

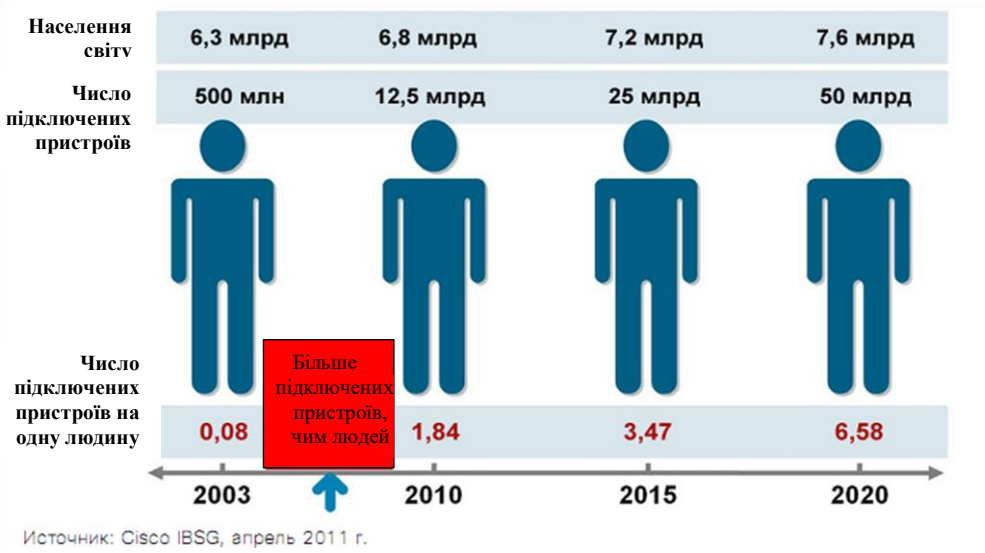


Рис. 1.2. Оцінки аналітиків корпорації Cisco

У цьому ж 2008 році у звіті Національної розвідувальної ради США (англ. National Intelligence Council) «Інтернет речей» вже фігурує як одна з шести потенційно небезпечних технологій. У звіті вказується, що повсюдне і непомітне для споживачів перетворення в інтернет-пристрої таких поширених речей, як: товарна упаковка, меблі, паперові документи, може завдати шкоди національній інформаційній безпеці [8].

Починаючи з 2009 року за підтримки Єврокомісії в Брюсселі щорічно проводиться конференція «Internet of Things» [10], на якій представляють доповіді єврокомісари і депутати Європарламенту, урядові чиновники з європейських країн, керівники таких компаній як SAP, SAS Institute, Telefónica, провідні вчені великих університетів і дослідницьких лабораторій.

З початку 2010-х років «Інтернет речей» стає рушійною силою парадигми «туманних обчислень» (англ. Fog computing), що розповсюджує принципи хмарних обчислень від центрів обробки даних до величезної кількості взаємодіючих географічно розподілених пристроїв, яка розглядається як платформа «Інтернету речей» [12].

Починаючи з 2011 року Gartner переміщує «Інтернет речей» в загальний цикл зрілості нових технологій на етап «технологічного тригера» із зазначенням терміну становлення більше 10 років, а в 2012

році випущений спеціальний цикл зрілості для технологій «Інтернету речей» [14]. З тих часів технології «Інтернету речей» стрімко розвиваються.

Декому може здаватися, що світ Інтернету речей - не більше ніж фантастика або дуже далека перспектива. Але це не так. У доповіді Fortune Business Insights вказується, що світовий ринок Інтернету речей, вартість якого в 2018 році оцінювалася в 190 мільярдів доларів, досягне до 2026 року 1,11 трильйона доларів, продемонструвавши сукупний темп зростання 24,7% в рік (рис. 1.3).

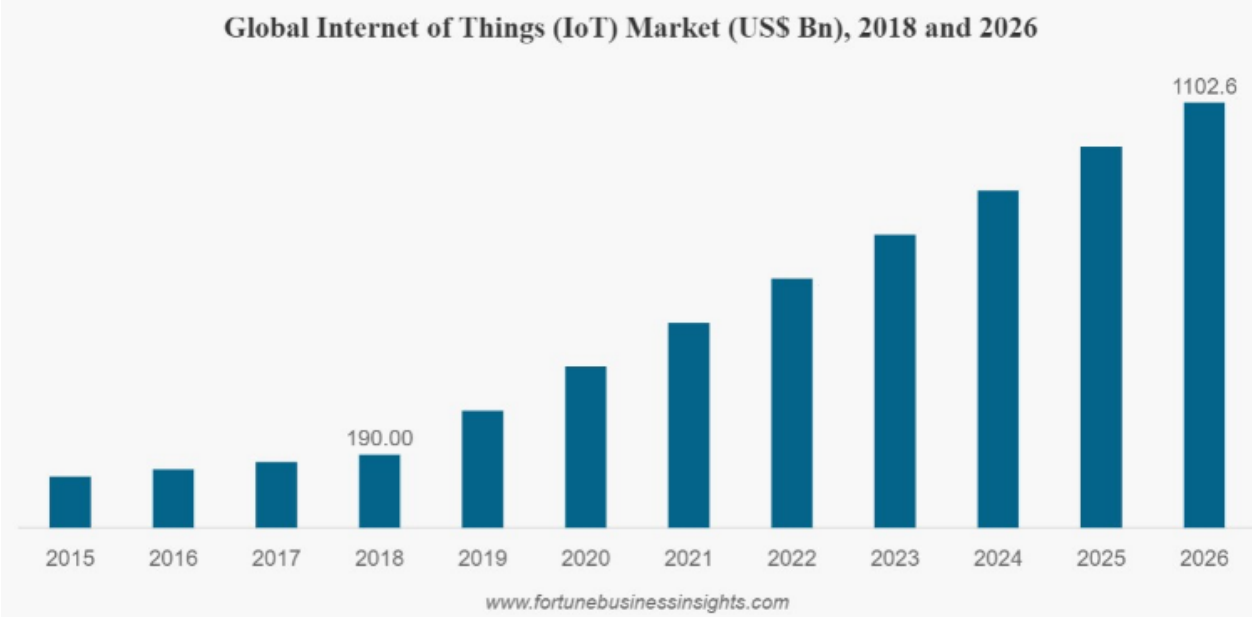


Рис. 1.3. Прогноз обсягу ринку IoT в 2018-2026.
(Джерело: Fortune Business Insights)

Очікується, що найбільшим сегментом ринку буде банківський сектор і сектор фінансових послуг.

Аналітики Gartner повідомляють, що в 2019 році кількість пристроїв IoT досягло 14,2 млрд. Компанія також прогнозує, що до 2025 року кількість підключених пристроїв досягне рівня в 25 мільярдів.

IDC дають ще більш оптимістичний прогноз: до 2025 року до мережі Інтернету речей буде щохвилини підключатися 152 200 пристроїв. Помноживши 152 200 на 525 600 (кількість хвилин в році), отримаємо, що в 2025 році Інтернет речей буде містити близько 80 мільярдів пристроїв.

За даними дослідження IoT - The Internet of Transformation 2018, опублікованого Juniper Research, ключовими ринками IoT залишаються Північна Америка, Західна Європа, Далекий Схід і Китай. Саме ці

регіони забезпечать більш 60% всіх доходів, пов'язаних з Інтернетом речей.

1.3. Принцип побудови та архітектура IoT

В цілому для Інтернету речей, як нового напряму розвитку інформаційних комунікацій, в даний час визначені лише найзагальніші концептуальні та архітектурні рішення. Найближчим часом основною проблемою буде гармонізація різних стандартів з метою формування єдиної і несуперечливої нормативної бази для практичної реалізації Інтернету речей.

1.3.1. Принцип побудови IoT

В рамках діяльності сектора стандартизації телекомунікацій Міжнародного союзу електрозв'язку (МСЕ-Т) проводиться ініціатива присвячена стандартизації Інтернету речей - IoT-GSI (Global Standards Initiative on Internet of Things).

IoT-GSI будує свою роботу на основі зусиль МСЕ-Т в таких областях, як мережеві аспекти ідентифікаційних систем (Network Identificator, NID), всепроникні сенсорні мережі (Ubiquitous Sensor Networks, USN), міжмашинного зв'язку (M2M), WEB речей (WoT) і т.п. В рамках серії МСЕ-Т Y.2xxx, присвяченій мережам наступного покоління, вже затверджені перші рекомендації, присвячені спеціально Інтернету речей: Y.2060 «Огляд Інтернету речей», Y.2063 «Основа WEB речей» і Y.2069 «Терміни та визначення Інтернету речей».

Офіційне визначення Інтернету речей наведено в Рекомендації МСЕ-Т Y.2060, згідно з яким IoT - глобальна інфраструктура інформаційного суспільства, що забезпечує передові послуги за рахунок організації зв'язку між речами (фізичними або віртуальними) на основі існуючих і розвиваються сумісних інформаційних і комунікаційних технологій.

Нижче наведено список визначень ключових термінів з Рекомендації Y.2060:

Мережа зв'язку (Communication Network): інфраструктурна мережа, що з'єднує пристрої та додатки, така як мережа на основі стека протоколів IP або Інтернет.

Річ (Thing): предмет фізичного світу (фізичні речі) або інформаційного світу (віртуальні речі), який може бути ідентифікований та інтегрований в мережі зв'язку.

Пристрій (Device): елемент обладнання, який володіє обов'язковими можливостями зв'язку та додатковими можливостями вимірювання, спрацьовування, а також введення, зберігання і обробки даних.

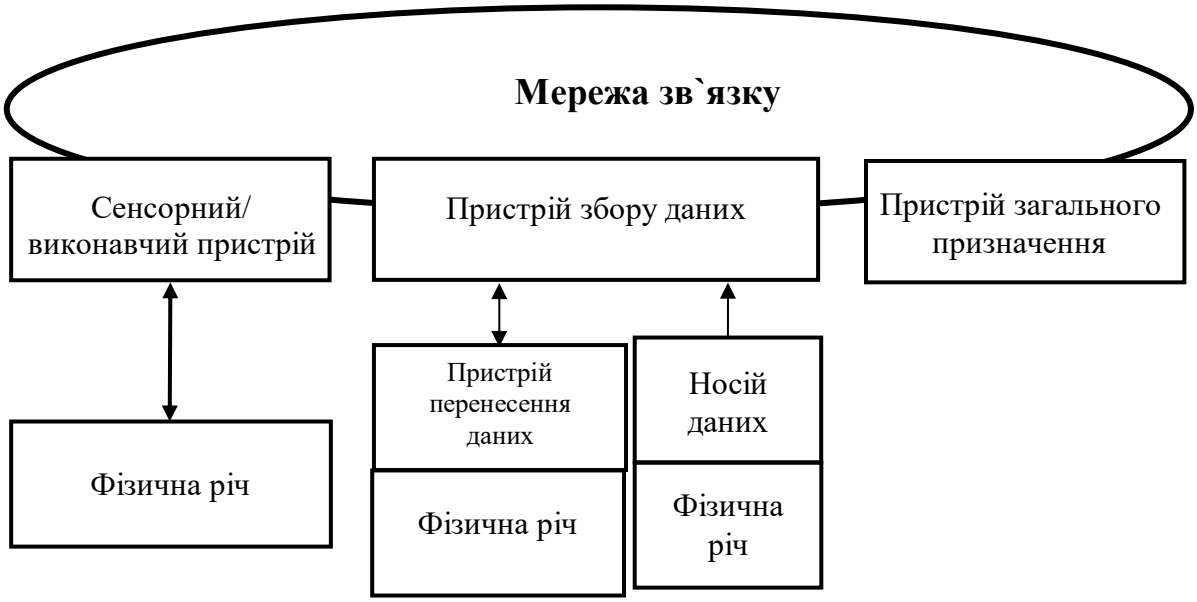


Рис. 1.4. Типи пристроїв IoT і їх взаємодія.
(Джерело: ITU-T Y.4000/Y2060)

Пристрій переносу даних (Data-carrying Device): пристрій переносу даних підключається до фізичної речі і непрямим чином з'єднує цю фізичну річ з мережами зв'язку.

Пристрій збору даних (Data-capturing Device): під пристроєм збору даних розуміється зчитуючий/записуючий пристрій, що має можливість взаємодії з фізичними речами. Взаємодія може здійснюватися непрямим чином за допомогою пристроїв перенесення даних або безпосередньо за допомогою носіїв даних, підключених до фізичних речей.

Носій даних (Data Carrier): безбатарейний об'єкт перенесення даних, що підключений до фізичної речі і має можливість надавати інформацію приданому для цього пристрою збору даних.

Сенсорний пристрій (Sensing Device): пристрій, який може виявляти або вимірювати інформацію, що відноситься до навколишнього середовища, і перетворювати її в цифрові електричні сигнали.

Виконавчий пристрій (Actuating Device): пристрій, який може перетворювати цифрові електричні сигнали, що надходять від інформаційних мереж, в дії.

Пристрій загального призначення (General Device): пристрій загального призначення володіє вбудованими можливостями обробки і зв'язку і може обмінюватися даними з мережами зв'язку з використанням дротових або бездротових технологій. Пристрої загального призначення включають обладнання та прилади, які стосуються різних галузей застосування IoT, наприклад, верстати, побутові електроприлади і смартфони.

Шлюз (Gateway): елемент IoT, що з'єднує пристрої з мережами зв'язку. Він виконує необхідну трансляцію між протоколами, що використовуються в мережах зв'язку і в пристроях.

Аналіз цих визначень показує, що МСЕ-Т в більшій мірі приділяє увагу аспектам комунікацій і міжмережевих з'єднань, ніж додаткам і особливостям IoT.

Схема відображення фізичних і віртуальних речей представлена на рис. 1.5.

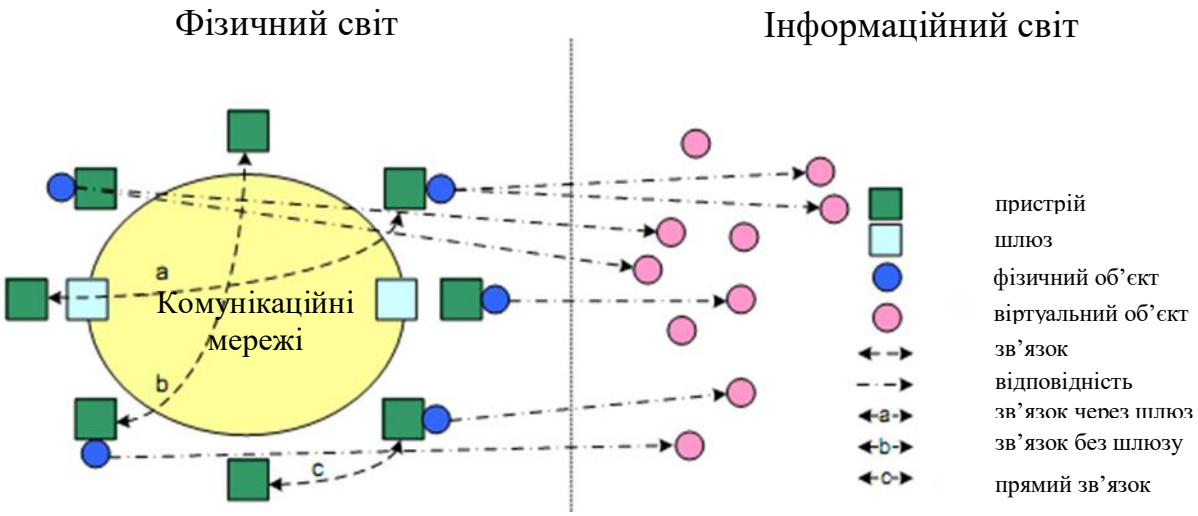


Рис. 1.5. Схема відображення фізичних і віртуальних речей (джерело: МСЕ- Y.2060)

З рисунка слідує, що віртуальні речі можуть існувати без їх фізичних втілень, в той час як фізичним об'єктам/речам обов'язково відповідає мінімум один віртуальний об'єкт. При цьому провідну роль грають саме пристрої, які можуть збирати різну інформацію і поширювати її по комунікаційних мережах різними способами: через шлюзи і через мережу; без шлюзів, але через мережу; безпосередньо між собою. Рекомендація Y.2060 описує різне поєднання перерахованих способів з'єднань. Це вказує на те, що МСЕ-Т передбачає використання для IoT безлічі мережевих технологій - глобальних мереж, локальних

мереж, бездротових систем, що самоорганізуються (ad-hoc) і чарункових (mesh) мереж. Зазначені мережі зв'язку переносять дані, зібрані пристроями, до відповідних програмних додатків, а також передають команди від програмних додатків до пристроїв.

В Рекомендації Y.2060 приведена еталонна модель IoT, яка включає чотири базових горизонтальних рівня (рис. 1.6): рівень додатків IoT, рівень підтримки додатків і послуг, мережевий рівень і рівень пристроїв.

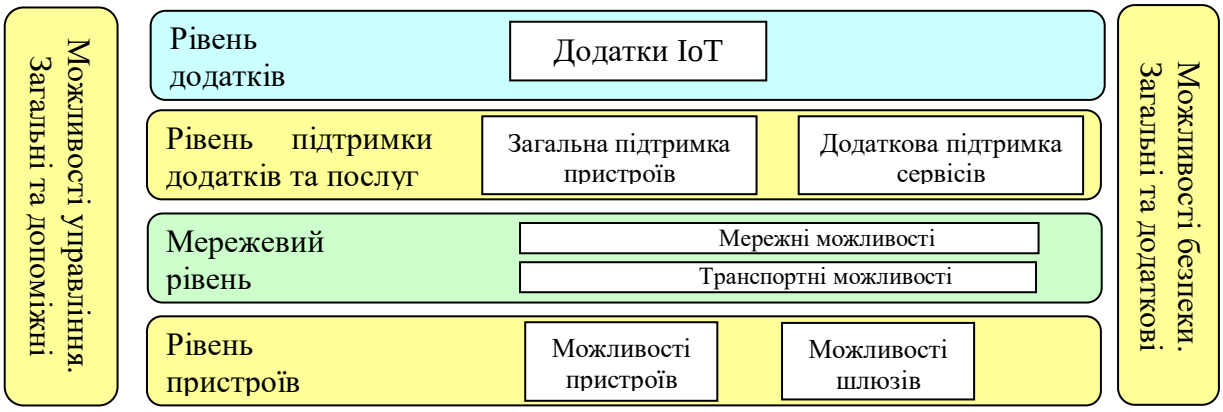


Рис. 1.6. Еталонна модель IoT згідно МСЕ-Т Y.2060

Рівень додатків IoT в Рекомендації Y.2060 детально не розглядається.

Рівень підтримки додатків і послуг включає загальні можливості для різних об'єктів IoT з обробки та зберігання даних, а також можливості, необхідні для деяких програм IoT або груп таких додатків.

Мережевий рівень включає мережеві можливості (функція управління ресурсами мережі доступу та транспортної мережі, управління мобільністю, функції авторизації, аутентифікації і розрахунків) і транспортні можливості (забезпечення зв'язності мережі для передачі інформації додатків і послуг IoT).

Рівень пристроїв включає можливості пристрою і можливості шлюзу. Можливості пристрою припускають прямий обмін з мережею зв'язку, обмін через шлюз, обмін через бездротову динамічну ad-hoc мережу, а також тимчасову зупинку і відновлення роботи пристрою для енергозбереження. Можливості шлюзу припускають підтримку множини інтерфейсів для пристроїв (шина CAN, ZigBee, Bluetooth, WiFi і ін.) і для мереж доступу/транспортних мереж (2G/3G, LTE, DSL і ін.). Додатковою можливістю шлюзу є підтримка конверсії протоколів, в разі, якщо протоколи інтерфейсів пристроїв і мереж відрізняються один від одного.

Існує також два вертикальних рівня - рівень управління і рівень безпеки, що охоплюють всі чотири горизонтальних рівня. Можливості вертикального рівня експлуатаційного управління передбачають управління наслідками відмов, можливостями мережі, конфігурацією, безпекою та даними для білінгу. Основними об'єктами управління є пристрої, локальні мережі та їх топологія, трафік і перевантаження на мережах. Можливості вертикального рівня безпеки залежать від горизонтального рівня. Для рівня підтримки програм та послуг визначено функції доступу, антивірусний захист, тести цілісності даних. Для мережевого рівня - можливості авторизації, аутентифікації, захисту інформації, протоколів сигналізації. На рівні пристроїв - можливості авторизації, аутентифікації, контроль доступу і конфіденційність даних.

Основною метою іншого проекту - Європейського інтеграційного проекту IoT-A (Internet of Things - Architecture), учасниками якого є різні компанії, є розробка еталонної архітектурної моделі Інтернету речей з описом основних складових компонентів, яка б дозволила інтегрувати різнорідні технології IoT в єдину взаємопов'язану архітектуру.

Функціональна модель IoT-A подана рис. 1.7. Вона дещо відрізняється від попередньої моделі МСЕ. Модель складається вже з семи горизонтальних рівнів, що доповнюються двома вертикальними (управління і безпека), які беруть участь у всіх процесах.

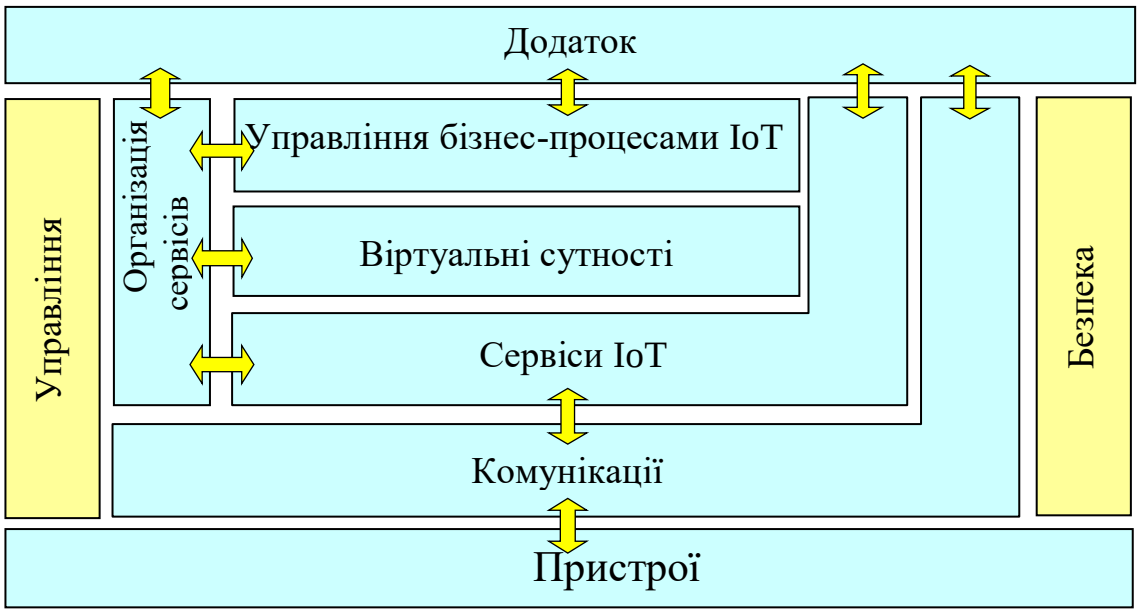


Рис. 1.7. Функціональна модель архітектури IoT-A

Якщо порівнювати між собою моделі передачі даних в Інтернет та модель передачі даних в Інтернеті речей IoT-A (рис. 1.8), то можна сказати, що друга буде дещо відрізнятися від першої. У моделі

архітектури IoT-A з’являються два важливих поняття: мережа з обмеженнями та мережа без обмежень. Мережа з обмеженнями характеризується відносно низькими швидкостями передачі - менше 1 Мбіт/с (наприклад, стандарт IEEE 802.15.4) і досить високими затримками. Мережа без обмежень відповідно характеризується високими швидкостями передачі даних (десятки Мбіт/с і більше) і схожа на існуючу зараз мережу Інтернет.

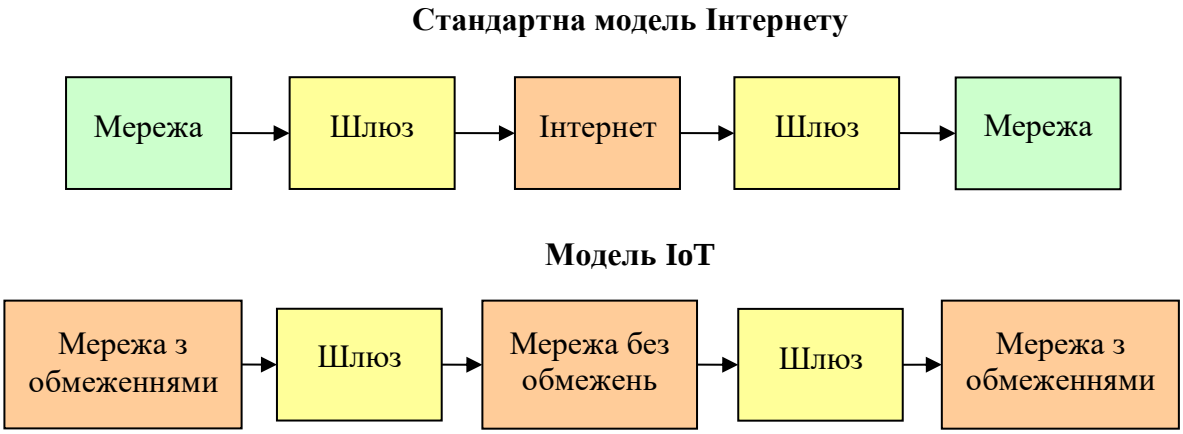


Рис. 1.8. Порівняння моделей передачі даних в Інтернеті і в IoT

1.3.2. Архітектура IoT

Інтернет речей концептуально належить до мереж наступного покоління, тому його архітектура багато в чому схожа з відомою чотиришаровій архітектурою NGN. IoT складається з набору різних інфокомунікаційних технологій, що забезпечують функціонування Інтернету речей, і його архітектура показує, як ці технології пов'язані один з одним. Архітектура IoT включає чотири функціональних рівня (рис. 1.9), описаних нижче.

1. Рівень сенсорів і сенсорних мереж.

Найнижчий рівень архітектури IoT складається з «розумних» (smart) об'єктів, інтегрованих з сенсорами (датчиками). Сенсори реалізують з'єднання фізичного і віртуального (цифрового) світів, забезпечуючи збір та обробку інформації в реальному масштабі часу. Мініатюризація, яка призвела до скорочення фізичних розмірів апаратних сенсорів, дозволила інтегрувати їх безпосередньо в об'єкти фізичного світу. Існують різні типи сенсорів для відповідних цілей, наприклад, для вимірювання температури, тиску, швидкості руху, місця розташування та ін. Сенсори можуть мати невелику пам'ять, даючи можливість записувати кілька результатів вимірювань. Сенсор може вимірювати фізичні параметри контрольованого об'єкта/явища і

перетворити їх в сигнал, який може бути прийнятий відповідним пристроєм. Сенсори класифікуються відповідно до їх призначення, наприклад, сенсори навколишнього середовища, сенсори для тіла, сенсори для побутової техніки, сенсори для транспортних засобів і т.д.

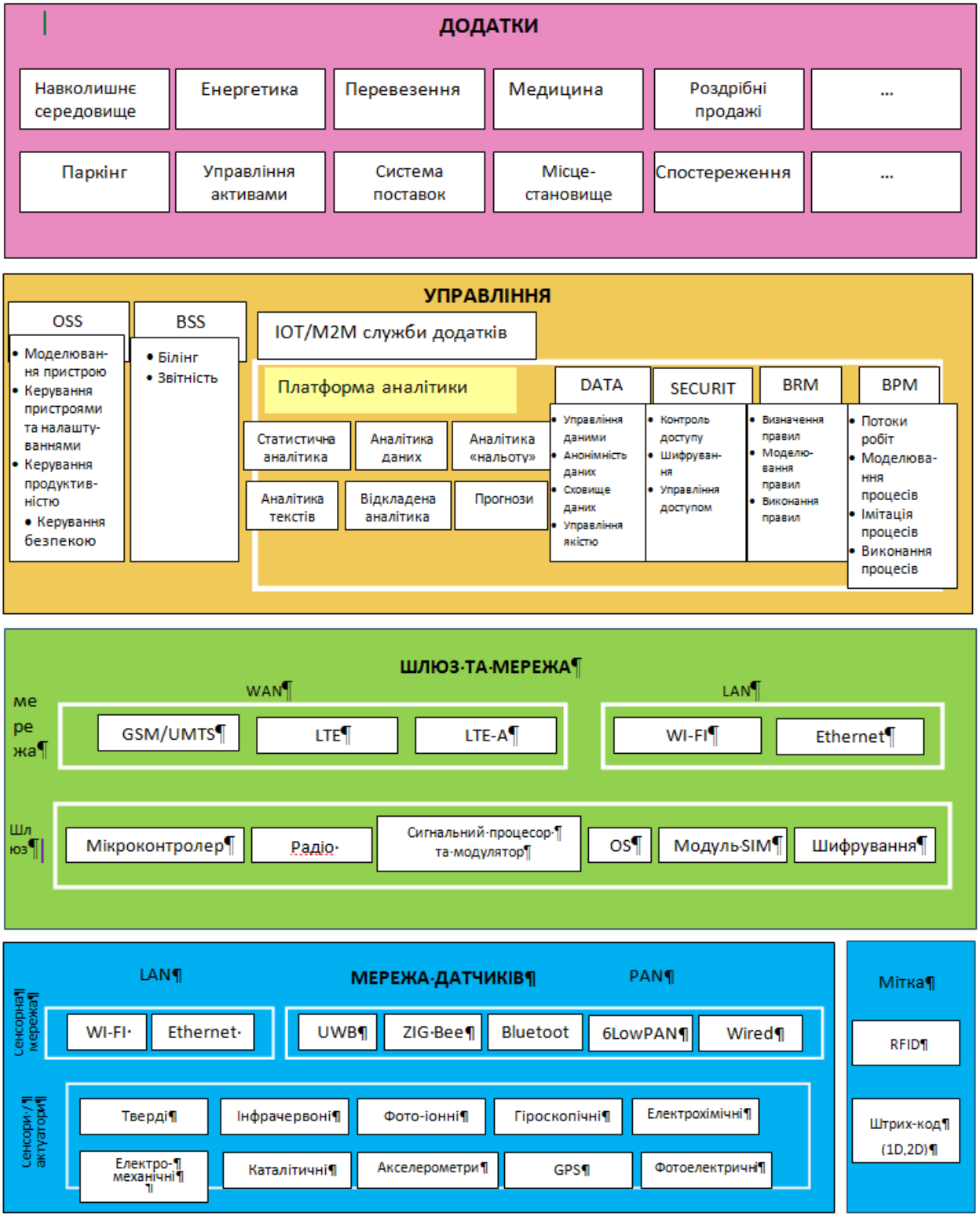


Рис. 1.9. Архітектура IoT

Більшість сенсорів вимагає з'єднання з агрегатором сенсорів (шлюзом), які можуть реалізуватися бути реалізовані з використанням локальної обчислювальної мережі (LAN, Local Area Network), таких як Ethernet і Wi-Fi або персональної мережі (PAN, Personal Apea Network), таких як ZigBee , Bluetooth і ультраширокосмугового бездротового зв'язку на малих відстанях (UWB, Ultra-Wide Band). Для сенсорів, які не вимагають підключення до агрегатора, їх зв'язок з серверами/додатками може надаватися з використанням глобальних бездротових мереж WAN, таких як GSM, GPRS і LTE. Сенсори, які характеризуються низьким енергоспоживанням і низькою швидкістю передачі даних, утворюють широко відомі бездротові сенсорні мережі (WSN, Wireless Сенсор Network). WSN набирають все більшої популярності, оскільки вони можуть містити набагато більше сенсорів з підтримкою роботи від батарей і охоплювали великі площі.

2. Рівень шлюзів і мереж.

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

3. Сервісний рівень

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

4. Рівень додатків

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

1.4. Класифікація систем IoT

Для здійснення класифікації систем IoT можна використовувати різні критичні ознаки.

За сферою застосуванням системи IoT можна розділити на побутові і промислові. Як правило, до промислових систем IoT висувається ряд додаткових вимог пов'язаних із їх взаємодією з існуючими промисловими системами та стандартами до них. У свою чергу промислові системи IoT можна поділити за областю застосування: транспорт, сільське господарство, медицина, військові і т.д.

За важливістю наслідків застосування системи IoT можна розділити на звичайні та критичні до наслідків роботи. Збої чи несправності у роботі критичних до наслідків роботи IoT можуть привести до складних наслідків, що пов'язані із життям та здоров'ям людини, станом навколишнього середовища. До критичних за наслідками систем IoT можна віднести наприклад медичні, авіаційні, IoT на атомних станціях, хімічних виробництвах і т.п. Як правило, критичні до наслідків роботи IoT потребують особливої уваги до питань безпеки роботи та додаткового резервування інформаційних каналів.

За ступенем захищеності системи IoT можна розділити на сильно-захищені чи стандартно-захищені. Питанням безпеки в системах IoT приділяється особлива увага. Застосування додаткових засобів захисту у системах IoT пов'язане з додатковими витратами, що суперечить принципу здешевлення пристроїв IoT. Тому підвищені засоби захисту будуть характерні наприклад, для критичних за наслідками систем IoT. Рішення щодо додаткових заходів і засобів захисту приймається для кожного типу систем IoT окремо кожним розробником.

За можливостями щодо руху (переміщення у просторі) системи IoT можна поділити на статичні та динамічні. Статичні системи IoT мають елементи (в основному це сенсори), що розміщуються в одному місці і на протязі всього свого існування їх місцеположення не

змінюється. Для динамічних систем IoT характерно переміщення у просторі і інформація про їх місцезнаходження також має важливе значення для роботи системи. Наприклад, автопілот автомобіля, безпілотний літальний апарат з передачею даних по інтернет-каналах.

За вимогами до часу проходження сигналу в системі розрізняють IoT реального часу (близькі до реального часу) та мало критичні до часу. Говорити про системи реального часу, що використовують інтернет-канали достатньо складно, адже точно розрахувати затримки проходження у сигналу фактично неможливо. У цьому випадку оцінюється максимальний чи середній час та оцінюється критичність цього фактору для системи IoT. Наприкдад, затримки сигналу управління в 1 сек. при включенні IoT-чайника не призведе до якихось складнощів, однак це велика затримка при управлінні швидкісним дроном і нехтувати нею не можна.

Контрольні питання до розділу 1

1. Що входить в поняття Інтернету речей?
2. Коли виник Інтернет речей і чому?
3. Вкажіть базові принципи IoT.
4. Як співвідносяться фізичні та віртуальні речі?
5. Хто займається стандартизацією Інтернету речей?
6. Поясніть призначення функціональних рівнів базової архітектури Інтернету речей.

2. ЗАСОБИ ІДЕНТИФІКАЦІЇ, ВИМІРЮВАНЬ, ВИКОНАВЧІ ПРИСТРОЇ ТА ЖИВЛЕННЯ В ІНТЕРНЕТІ РЕЧЕЙ

2.1. Засоби ідентифікації в IoT

Залучення в «Інтернеті речей» предметів фізичного світу, навіть не обов'язково оснащених засобами підключення до мереж передачі даних, вимагає застосування технологій ідентифікації цих предметів («речей»). Хоча поштовхом для появи концепції стала технологія RFID, але в якості таких технологій можуть використовуватися всі засоби, що застосовуються для автоматичної ідентифікації: оптично розпізнавані ідентифікатори (штрих-коди, Data Matrix, QR-коди), засоби визначення місцезнаходження в режимі реального часу. При всеосяжному поширенні «Інтернету речей» принципово необхідно забезпечити унікальність ідентифікаторів об'єктів, що, в свою чергу, вимагає стандартизації.

Для об'єктів, безпосередньо підключених до мережі Інтернет, традиційний ідентифікатор - MAC-адреса мережевого адаптера, що дозволяє ідентифікувати пристрій на канальному рівні, при цьому діапазон доступних адрес практично невичерпний (248 адрес в просторі MAC-48). Але використання ідентифікатора канального рівня не дуже зручно для додатків. Ширші можливості по ідентифікації для таких пристроїв дає протокол IPv6, що забезпечує унікальними адресами мережевого рівня не менше 300 млн. пристроїв на одного жителя Землі.

Загальний термін для автоматичної ідентифікації та збору даних (AIDC, від англ. Automatic Identification and Data Capture) - методи автоматичної ідентифікації об'єктів, збору даних про них і обробку даних автоматичними і автоматизованими системами. Зазвичай до AIDC відносять такі технології [14]:

Контактні методи: магнітна карта та чіп-карта.

Безконтактні методи: оптичні (штрих код, Data Matrix, OCR) та радіочастотні (RFID, RTLS).

2.1.1. MAC-адреси

MAC-адреса (від англ. Media Access Control - управління доступом до середовища, також Hardware Address) - унікальний ідентифікатор, який присвоюється кожній одиниці активного обладнання або деяким їх інтерфейсам в комп'ютерних мережах Ethernet.

При проектуванні стандарту Ethernet було передбачено, що кожна мережева карта (так само як і вбудований мережевий інтерфейс) повинна мати унікальний шестибайтний номер (MAC-адресу), «прошитий» в ній при виготовленні. Цей номер використовується для ідентифікації відправника і одержувача фрейму і передбачається, що при появі в мережі нового комп'ютера (або іншого пристрою, здатного працювати в мережі) адміністратору не доведеться налаштовувати MAC-адресу цього комп'ютера вручну [15].

Унікальність MAC-адрес досягається тим, що кожен виробник отримує в координаційному комітеті IEEE Registration Authority діапазон з 16777216 (2^{24}) адрес і, в міру вичерпання виділених адрес, може запросити новий діапазон. Тому за трьома старшими байтам MAC-адреси можна визначити виробника. Існують таблиці, що дозволяють визначити виробника по MAC-адресі; зокрема, вони включені в програми типу arpalert.

У широкомовних мережах (таких, як мережі на основі Ethernet) MAC-адреса дозволяє унікально ідентифікувати кожен вузол мережі і доставляти дані тільки цьому вузлу. Таким чином, MAC-адреси формують основу мереж на канальному рівні моделі OSI, яку використовують протоколи більш високого (мережевого) рівня. Для перетворення MAC-адрес в адреси мережевого рівня і назад застосовуються спеціальні протоколи (наприклад, ARP і RARP в мережах IPv4, і NDP в мережах на основі IPv6).

Більшість мережевих протоколів канального рівня використовують 1 з 3 просторів MAC-адрес, керованих IEEE (або MAC-48, або EUI-48, або EUI-64). Адреси в кожному з цих просторів, теоретично, повинні бути глобально унікальними. Але не всі протоколи використовують MAC-адреси; і не всі протоколи, що використовують MAC-адреси, потребують подібної унікальності цих адрес.

Адреси на кшталт MAC-48 найбільш поширені; вони використовуються в таких технологіях, як Ethernet, Token ring, FDDI, WiMAX і інших. Вони складаються з 48 біт; таким чином, адресний простір MAC-48 налічує 2^{48} (або 281 474 976 710 656) адрес. Згідно з підрахунками IEEE, цього запасу адрес вистачить щонайменше до 2100 року.

EUI-48 від MAC-48 відрізняється лише семантично: в той час як MAC-48 використовується для мережевого обладнання - EUI-48 застосовується для інших типів апаратного і програмного забезпечення.

Ідентифікатори EUI-64 складаються з 64 біт і використовуються в FireWire, а також в IPv6 (в якості молодших 64 біт мережевої адреси вузла).

Структура MAC-адреси подана на рис. 2.1

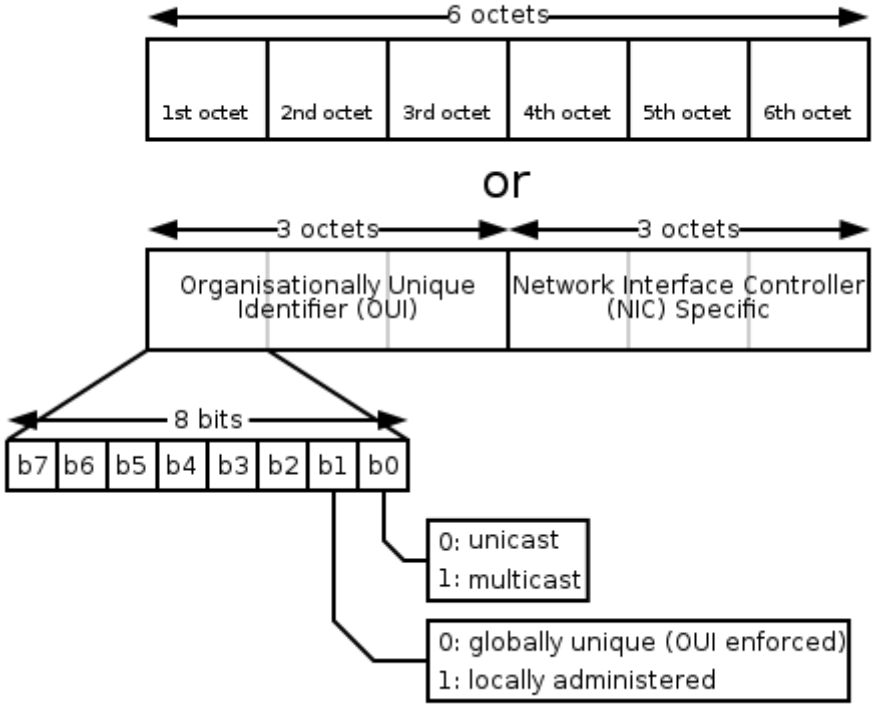


Рис. 2.1. Структура MAC-адреси

Стандарти IEEE визначають 48-розрядний (6 октетів) MAC-адресу, яка розділена на чотири частини.

Перші 3 октети (в порядку їх передачі по мережі; старші 3 октети, якщо розглядати їх у традиційній біт-реверсному шістнадцятирічному запису MAC-адрес) містять 24-бітний унікальний ідентифікатор організації (OUI) [1], або код MFG (Manufacturing, виробника), який виробник отримує в IEEE. При цьому, в найпершому октеті використовуються тільки 6 старших розрядів, а два молодших мають спеціальне призначення:

- Нульовий біт - вказує: для одиночного (0) або групового (1) адресата призначений кадр;

- Перший біт - вказує, чи є MAC-адреса глобально (0) або локально (1) адміністрованою.

Наступні три октети - вибираються виробником для кожного екземпляра пристрою (за винятком мереж системної мережевої архітектури SNA).

Таким чином, глобально адмініструєма MAC-адреса пристрою є глобально унікальною і зазвичай «зашита» в апаратуру.

Існує поширена думка, що MAC-адресу «жорстко вшито» в мережеву карту і змінити її не можна (або тільки за допомогою програматора) - але насправді MAC-адреса легко змінюється програмним шляхом, так як значення, вказане через драйвер, має більш високий пріоритет, ніж «зашите» в плату. Однак все ж існує обладнання, в якому зміну MAC-адреси зробити неможливо без вибору програм (зазвичай це телекомунікаційне обладнання; наприклад, приставки для IP-TV (STB)).

У деяких пристроях, оснащених веб-інтерфейсом управління, можлива зміна MAC-адреси під час налаштування: більшість маршрутизаторів дозволяють дублювати MAC-адресу мережевої плати, через яку він підключений до комп'ютера.

В ОС «Windows» зміну MAC-адреси можна здійснити вбудованими засобами ОС: у властивостях мережевої плати, у вкладці «Додатково», для редагування є властивість «Мережевий адреса» (англ. «Network Address», у деяких виробників мережевих плат це властивість називається «Locally Administered Address») - дозволяє примусово привласнити потрібний MAC-адресу.

Як бачимо використання MAC-адрес для кінцевих IoT пристроїв досить ускладнено.

2.1.2. Оптичні ідентифікатори

Штриховий код (штрихкод [1]) - графічна інформація, що наноситься на поверхню, маркування або упаковку виробів, що надає можливість зчитування її технічними засобами - послідовність чорних і білих смуг, або інших геометричних фігур.

Виділяють такі способи оптичного кодування інформації: лінійні та двовимірні.

Лінійними (смуговими кодами) називаються штрих-коди, що читаються в одному напрямку (по горизонталі). Найбільш поширені лінійні символіки: EAN (EAN-8 складається з 8 цифр, EAN-13 - використовуються 13 цифр); UPC (UPC-A, UPC-E); Code56; Code128 (UPC/EAN-128); Codabar; «Interleaved 2 of 5». Лінійні символіки дозволяють кодувати невеликий об'єм інформації.

Двомірні символіки були розроблені для кодування великого обсягу інформації. Розшифровка такого коду проводиться в двох вимірах (по горизонталі і по вертикалі).

Двомірні коди поділяються на багаторівневі (stacked) і матричні (matrix). Багаторівневі штрих-коди з'явилися історично раніше, і являють собою поставлені один на одного кілька звичайних лінійних

кодів. Матричні ж коди більш щільно упаковують інформаційні елементи по вертикалі.

Рис. 2.2. Приклад штрихового коду

а) б)

Рис. 2.3. Приклади коду Data Matrix, що кодує текст: а -«Wikipedia, the free encyclopedia»; б - двовимірний штрих-код на медичному рецепті

В даний час розроблено безліч двовимірних штрих-кодів, що застосовуються з тією чи іншою частотою.

DataMatrix - двовимірний матричний штрихкод, який представляє собою чорно-білі елементи або елементи декількох різних ступенів яскравості, зазвичай у формі квадрата, розміщені в квадратній або прямокутній групі. Матричний штрихкод призначений для кодування тексту або даних інших типів. Найчастіше в промисловості і торгівлі застосовуються бітові матриці, що кодують від декількох байт до 2 кілобайт даних. При бажанні можна роздрукувати на принтері матриці ємністю в сотні кілобайт і потім зчитувати їх з досить високою точністю за допомогою фотоапаратів, матриці яких містять мільйони пікселів. Прообразом штрихкодів у вигляді матриць є перфокарти [13].

Один з варіантів бітових матриць, «Data Matrix», був розроблений компанією RVSI/Acuity CiMatrix (нині частина концерну Siemens AG). Код застосовується для маркування в електроніці, автомобілебудуванні,

харчовій промисловості, авіакосмічної та оборонної промисловості, енергетичному машинобудуванні. [1]

Також дані коди застосовуються в рекламної та розважальної сферах. За допомогою DataMatrix можна закодувати як текст, так і інші типи даних - веб-посилання, адреси електронної пошти, номери телефонів і SMS.

Рис. 2.4. Приклад маркування коду на твердих поверхнях

Рис. 2.5. Приклад універсального промислового зчитувача кодів

Технічна специфікація

Символи DataMatrix утворені з модулів, розташованих в межах шаблону пошуку. Ними можна зашифрувати до 3116 кодів таблиці ASCII (включаючи надлишкову інформацію). Символ складається з областей даних, які містять модулі у вигляді періодичного масиву. Кожна область даних обмежена шаблоном пошуку і оточена з усіх чотирьох сторін межами вільної зони. (Зауваження: модулі можуть бути круглими або квадратними, конкретна форма стандартом не закріплена).

Data Matrix ECC 200 - це новітня версія DataMatrix, що використовує коди Ріда-Соломона для запобігання помилок і відновлення стертої інформації. ECC 200 уможливлює відновлення всієї послідовності закодованої інформації, коли символ містить 30% пошкоджень, припускаючи, що матриця все ще розташована в точності правильно. DataMatrix має частоту появи помилок менше, ніж 1 на 10 мільй відсканованих символів.

Символи мають парну кількість рядів і парна кількість стовпців. Більшість символів квадратні розмірами від 10x10 до 144x144 модулів. Однак деякі символи прямокутні і мають розміри від 8x18 до 16x48

модулів (тільки парні значення). Всі символи, що підтримують виправлення помилок ECC 200, можуть бути пізнані по верхньому правому кутовому модулю, має один колір з фоновим.

Додаткові можливості, що відрізняють ECC 200 символи від більш ранніх стандартів:

зворотний порядок читання символів (світле зображення на темному тлі);

специфікація набору символів;

прямокутні символи;

структурне приєднання (з'єднання до 16 символів для кодування більшої кількості інформації) [2].

Основною відмінністю Data Matrix від інших матричних штрих-кодів є можливість вибору форми зображення коду, яка може бути квадратної або прямокутної форми

QR-код

QR-код (англ. Quick Response Code - код швидкого реагування; скор. QR code) - товарний знак для типу матричних штрих-кодів (або двовимірних штрих-кодів), спочатку розроблених для автомобільної промисловості Японії. Штрихкод - прочитується машиною, це оптична мітка, що містить інформацію про об'єкт, до якого вона прив'язана. QR-код використовує чотири стандартизованих режиму кодування (числовий, буквено-цифровий, двійковий і кандзі) для ефективного зберігання даних; можуть також використовуватися розширення [1].

Рис. 2.6. Приклад художнього QR-коду. Незважаючи на додаткову інформацію, цей код залишається читаним

Система QR-кодів стала популярною за межами автомобільної промисловості завдяки можливості швидкого зчитування і більшої місткості в порівнянні зі штрих-кодами стандарту UPC. Розширення включають відстеження продукції, ідентифікацію предметів, відстеження часу, управління документами і загальний маркетинг [2].

QR-код складається з чорних квадратів, розташованих у квадратній сітці на білому тлі, які можуть зчитуватися за допомогою

пристроїв обробки зображень, таких як камера, і оброблятися з використанням кодів Ріда-Соломона до тих пір, поки зображення не буде належним чином розпізнано. Потім необхідні дані витягуються з шаблонів, які присутні в горизонтальних і вертикальних компонентах зображення.

QR-код розроблений і представлений японською компанією Denso-Wave [3] в 1994 році. Величезна популярність штрихкодів в Японії призвела до того, що обсяг інформації, зашифрованої в них, незабаром перестав влаштовувати промисловість. Японці почали експериментувати з новими сучасними способами кодування невеликих обсягів інформації в графічній картинці.

На відміну від старого штрихкоду, який сканують тонким променем, QR-код визначається датчиком або камерою як двовимірне зображення. Три квадрата в кутах зображення і менші синхронізуючі квадратики по всьому коду дозволяють нормалізувати розмір зображення і його орієнтацію, а також кут, під яким датчик розташований до поверхні зображення. Точки переводяться в двійкові числа з перевіркою по контрольній сумі.

Основна перевага QR-коду - це легке розпізнавання скануючим обладнанням, що дає можливість використання в торгівлі, виробництві, логістиці.

Хоча позначення «QR code» є зареєстрованим товарним знаком «DENSO Corporation», використання кодів не обкладається жодними ліцензійними відрахуваннями, а самі вони описані і опубліковані в якості стандартів ISO.

Специфікація QR-коду не описує формат даних. Найбільш популярні програми перегляду QR-кодів підтримують такі формати даних: URL, закладка в браузер, Email (з поміткою), SMS на номер (с темою), MeCard, vCard, географічні координати. Також деякі програми можуть розпізнавати файли GIF, JPG, PNG або MID менше 4 КБ і зашифрований текст, але ці формати не отримали популярності.

Найменший QR-код (версія 1) має розмір 21 × 21 піксель (без урахування полів), найбільший (версія 40) - 177 × 177 пікселів.

Існує чотири основних види кодування QR-кодів:

Цифрове: 10 бітів на три цифри, до 7089 цифр.

Алфавітно-цифрове: підтримуються 10 цифр, літери від A до Z і кілька спецсимволів. 11 бітів на два символи, до 4296 символів

Байтове: дані в будь-якому зручному кодуванні (за замовчуванням ISO 8859-1), до 2953 байт.

Кандзі: 13 бітів на ієрогліф, до 1817 ієрогліфів.

Також існують «псевдокодіровкі»: завдання способу кодування в даних, розбиття довгого повідомлення на кілька кодів і т. д.

Для виправлення помилок застосовується код Ріда-Соломона з 8-бітовим кодовим словом. Є чотири рівня надмірності: 7, 15, 25 і 30%. Завдяки виправлення помилок вдається нанести на QR-код малюнок і все одно залишити його читаним.

Щоб в коді не було елементів, здатних заплутати сканер, область даних складається по модулю 2 зі спеціальною маскою. Для коректної роботи декодер повинен перепробувати всі варіанти масок, порахувати штрафні очки для кожної з особливими правилами і вибрати найбільш вдалу.

2.1.3. Радіочастотна ідентифікація (RFID)

RFID (Radio Frequency Identification) — це спосіб забезпечення зберігання та передачі інформації зі зручного носія-мітки в потрібне місце, за допомогою спеціальних пристроїв. Такі мітки-ідентифікатори дозволяють полегшити розпізнавання різних об'єктів: товарів в магазині, рухомих засобів при транспортуванні, допомагають визначати їх місце розташування, можуть ідентифікувати людей і тварин, не кажучи вже про широкі можливості ідентифікації документів і майна (рис.2.7).

Рис. 2.7. Приклад позначення технології RFID

RFID-система складається з трьох основних елементів (рис.2.8): RFID-зчитувача («рідера»), RFID-мітки (інші назви: ідентифікатор, транспондер чи "тег", від англ. tag) та комп'ютерної системи обробки даних [11].

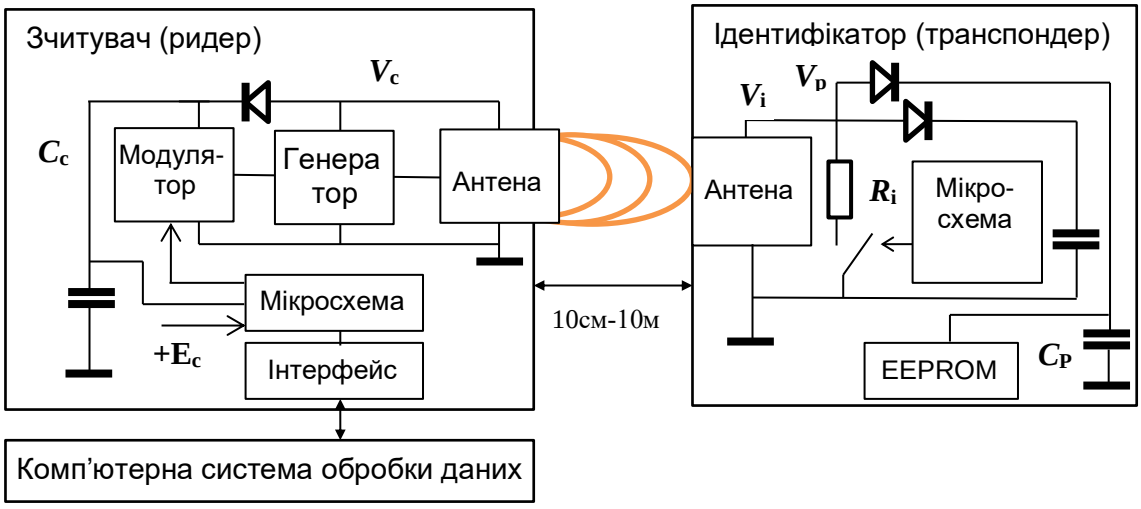


Рис. 2.8. Структура RFID-системи

RFID-зчитувач (або «рідер») має приймально-передавальний пристрій - він посилає сигнал до мітки та приймає відповідний сигнал від мітки. Мікропроцесор (пасивної мітки), який віддає сигнал живиться за рахунок індукційного струму з котушки або антени, в якій виробляється струм, коли мітка проходить через електромагнітне поле. Мітки можуть мати більш складну реалізацію з мікропроцесором, що перевіряє і декодує дані, а не просто відповідає ідентифікатором, а також пам'ять, що зберігає дані для наступної передачі, якщо це необхідно.

Зчитувачі можуть бути постійно підключеними до системи обробки, або працювати автономно. Залежно від частотного діапазону мітки, дистанція стійкого зчитування і запису даних може бути різна. Розрізняють стаціонарні та мобільні зчитувачі.

Стаціонарні зчитувачі кріпляться нерухомо на стінах, дверях, рухомих складських пристроях (штабеляторах, навантажувачах). Вони можуть бути виконані у вигляді замка, вмонтовані в стіл або закріплені поряд з конвеєром на шляху проходження виробів. В порівнянні з мобільними, зчитувачі такого типу зазвичай мають більшу зону читання та потужність, і здатні одночасно обробляти значний потік інформації. Стаціонарні зчитувачі на виробництві інтегруються в інформаційну систему що дозволяє поетапно фіксувати переміщення маркованих об'єктів в реальному часі, або ідентифікувати положення мічених предметів в просторі.

Мобільні зчитувачі мають порівняно меншу дальність дії і часто не мають постійного зв'язку з системою обробки даних. Вони мають внутрішню пам'ять, в яку записуються дані з прочитаних міток (потім цю інформацію можна синхронізувати з системою обліку) і, як і

стаціонарні зчитувачі, здатні записувати дані в мітку (наприклад, інформацію про проведений контроль).

Основними частинами RFID-мітки (див. рис. 2.8) є: приймач, передавач, антена і блок пам'яті. Все окрім антени розміщується в корпусі маленької мікросхеми — чіпа, тому з вигляду може здатися що мітка складається лише з багатовиткової антени і чіпа. В активних мітках є ще одна частина - джерело живлення, наприклад літієва батарейка. Енергію мітка отримує з радіосигналу антени зчитувача або від власного джерела живлення, після отримання зовнішнього сигналу, мітка відповідає власним сигналом, у якому міститься певна ідентифікаційна інформація.

RFID-мітка приймає від антени електромагнітну хвилю. Хвиля активізує її, і стають можливими як запис даних на мітку, так і зчитування даних з мітки. Антена служить таким чином багатофункціональним каналом зв'язку між прийомопередавачем і міткою, повністю забезпечує процеси передачі і отримання даних.

Антени різних форм і розмірів можуть вбудовуватися в сканери, ворота, турнікети, - різні засоби для роботи з RFID-мітками, з метою забезпечення доступу до інформації, що зберігається в мітках товарів, предметів, людей, транспорту і т. д. - все, що переміщується через зону дії антени сканера, і має на собі RFID-мітки (рис. 2.9).

Антена може безперервно працювати і постійно зчитувати мітки у великій кількості, весь час опитуючи їх, або може включатися на деякий час за сигналом від оператора. Антена з прийомопередавачем і декодером часто знаходяться в одному загальному корпусі, щоб сигнал від антени відразу б демодулювався, розшифровувався і передавався б через стандартний інтерфейс на ПК для подальшої обробки отриманих даних.

Немає ніякої потреби в контакті або прямій видимості між зчитувачем і міткою, оскільки радіосигнал легко проникає через неметалеві матеріали. Таким чином, мітки навіть можуть бути сховані усередині тих об'єктів, що підлягають ідентифікації.

Існує декілька способів класифікації RFID-міток і систем: за робочою частотою; за джерелом живлення; за типом пам'яті.

За робочою частотою виділяють високочастотні, середньо частотні та низькочастотні системи.

Рис. 2.9. Приклади антен RFID-міток

Високочастотні (850–950 Mhz і 2.4-5 Ghz), що використовуються там, де потрібна велика відстань і висока швидкість зчитування, наприклад контроль залізничних вагонів, автомобілів, системи збору відходів. Наприклад, рідери встановлюють на шлагбаумах, а транспондер закріплюється на вітровому або бічному склі автомобіля. Велика дальність дії дає можливість безпечної установки рідерів поза межами досяжності людей.

Середньочастотні системи (10 MHz—15 MHz) застосовують там, де повинні бути передані великі обсяги даних.

Низькочастотні (100–500 Khz). Використовуються там, де припустима невелика відстань між об'єктом і рідером. Звичайна відстань зчитування становить 0,5 метра. Для міток, умонтованих у маленькі «кнопочки» (наприклад домофон), дальність читання, як правило, ще

менше — близько 0,1 метра. Велика антена рідера може якоюсь мірою компенсувати таку дальність дії невеличкої мітки, але випромінювання високовольтних ліній, моторів, комп'ютерів, ламп і т.п., заважає її роботі. Більшість систем керування доступом, безконтактні картки, керування складами і виробництвом використовує низьку частоту.

За типом джерела живлення RFID-мітки діляться на: пасивні, активні, напівпасивні.

Пасивні RFID-мітки не мають вбудованого джерела енергії. Електричний струм, що індукується в антені електромагнітним сигналом від зчитувача, забезпечує достатню потужність для функціонування кремнієвого CMOS-чипа, розміщеного в мітці, і передачі у відповідь сигналу.

Пасивні мітки УВЧ (ультрависокочастотні дециметрові хвилі) і НВЧ (надвисокочастотні сантиметрові і міліметрові хвилі) діапазонів (860–960 МГц і 2,4-2,5 ГГц) передають сигнал методом модуляції відбитого сигналу частоти, що несе (англ. Backscattering Modulation модуляція зворотного розсіяння). Антена зчитувача випромінює сигнал частоти, що несе, і приймає відбитий від мітки модульований сигнал. Пасивні мітки ВЧ діапазону передають сигнал методом модуляції навантаження сигналу частоти, що несе (англ. Load Modulation модуляція навантаження). Кожна мітка має ідентифікаційний номер. Пасивні мітки можуть містити перезаписувану незалежну пам'ять EEPROM-типу. Дальність дії міток становить 1—200 см (ВЧ-МІТКИ) і 1-10 метрів (УВЧ і НВЧ-мітки).

Активні RFID-мітки володіють власним джерелом живлення і не залежать від енергії зчитувача, унаслідок чого вони читаються на дальній відстані, мають великі розміри і можуть бути оснащені додатковою електронікою. Проте, такі мітки найдорожчі, а у батарей обмежений час роботи. Активні мітки в більшості випадків надійніші (наприклад, здійснюють меншу кількість помилок), ніж пасивні, завдяки особливій сесії зв'язку між міткою і пристроєм зчитування. Активні мітки, володіючи власним джерелом живлення, також можуть генерувати вихідний сигнал більшого рівня, ніж пасивні, дозволяючи застосовувати їх в агресивніших для радіочастотного сигналу середовищах: воді (включаючи людей і тварин, які в основному складаються з води), металах (корабельні контейнери, автомобілі), для великих відстаней на повітрі. Більшість активних міток дозволяють передати сигнал на відстані в сотні метрів при тривалості життя батареї живлення до 10 років. Деякі RFID-мітки мають вбудовані сенсори, наприклад, для моніторингу температури товарів, які швидко псуються. Інші типи сенсорів в сукупності з активними мітками можуть

застосовуватися для вимірювання вологості, реєстрації поштовхів/вібрації, світла, радіації, температури і газів в атмосфері (наприклад, етилену).

Активні мітки зазвичай мають набагато більший радіус зчитування (до 300 м) і обсяг пам'яті, ніж пасивні, і здатні зберігати більший обсяг інформації для відправки приймачем. В даний час, активні мітки роблять розмірами не більше звичайної пілюлі і продають за ціною в декілька доларів.

Напівпасивні RFID-мітки, також їх називають напівактивними, дуже схожі на пасивні мітки, але оснащені батареєю, яка забезпечує чип енергоживленням. При цьому дальність дії цих міток залежить тільки від чутливості приймача зчитувача і вони можуть функціонувати на більшій відстані і з кращими характеристиками.

За типом використовуваної пам'яті RFID-мітки діляться на:

- RO (англ. Read Only) дані записуються тільки один раз, відразу при виготовленні. Такі мітки придатні тільки для ідентифікації. Ніяку нову інформацію в них записати не можна, і їх практично неможливо підробляти.
- WORM (англ. Write Once Read Many) окрім унікального ідентифікатора такі мітки містять блок одноразово записуваної пам'яті, яку надалі можна багато разів читати.
- RW (англ. Read and Write) такі мітки містять ідентифікатор і блок пам'яті для читання/запису інформації. Дані в них можуть бути перезаписані багаторазово.

Основні переваги RFID-технології:

- для RFID не потрібний контакт або пряма видимість;
- RFID-мітки читаються швидко і точно (наближаючись до 100%-ої ідентифікації);
- RFID може використовуватися навіть в агресивних середовищах, а RFID-мітки можуть читатися через бруд, фарбу, пар, воду, пластмасу, деревину;
- пасивні RFID-мітки мають фактично необмежений термін експлуатації;
- RFID-мітки несуть велику кількість інформації і можуть бути інтелектуальними;
- RFID-мітки можуть бути не тільки для читання, але і з записом інформації;

Недоліки RFID-технології

- В деяких випадках мітки не деактивуються повністю, є можливість повторного спрацьовування.

- Мітку можна виявити на товарі і в багатьох випадках пошкодити або відірвати.
- З огляду на легку можливість маскування, можуть використовуватись для шпигунства без згоди власника товару.
- У випадку оплати товару карткою банку технічно залишається можливість асоціювання імені власника з товаром.
- В індивідуальних випадках категорична відмова від імплантації з огляду на страх втрати приватності індивідом.
- Технічно можливо збирати приватну інформацію - національність та інші дані з паспортів, куплену літературу і т.п.
- RFID-мітку відносно легко ввести в оману. Для цього необхідно щоб мітка пройшла в радіусі дії несанкційованого зчитувача, часто мобільного, який збереже інформацію з мітки у себе. Після цього достатньо прийти до точки доступу і протранслювати зчитану інформацію системі за допомогою спеціального транслятора. Ускладнює ситуацію й те, що у виготовленні транслятора не виникає ніяких складностей і його схеми є широко розповсюджені в мережі Інтернет.

2.1.4. Система позиціонування в режимі реального часу RTLS

RTLS (скор. від англ. Real-time Locating Systems - система позиціонування в режимі реального часу) - автоматизована система, що забезпечує ідентифікацію, визначення координат, відображення на плані місцезнаходження контрольованих об'єктів в межах території, охопленій необхідною інфраструктурою. RTLS накопичує, обробляє і зберігає інформацію про місцезнаходження і переміщеннях людей, предметів, мобільних механізмів і транспортних засобів з метою моніторингу технологічних і бізнес-процесів, сигналізації про відхилення від регламентів, а також з метою ретроспективного аналізу тих чи інших процесів і ситуацій.

До основних характеристик RTLS можна віднести:

Точність позиціонування - точність визначення координат об'єкту, що контролюється. Для різних технологій RTLS характерна точність позиціонування становить від декількох десятків метрів (для WiFi) до декількох сантиметрів (для ультразвукових).

Достовірність позиціонування - в реальних умовах точність позиціонування в значній мірі залежить від впливу перешкод і багатопроменевого загасання (відбитих сигналів), тому говорячи про точність позиціонування RTLS зазвичай вказують і вірогідну характеристику достовірності. Наприклад, «точність позиціонування 1

метр з достовірністю 90%», тобто точність буде забезпечуватися в 90% вимірювань.

Періодичність опитування - для забезпечення позиціонування в режимі реального часу проміжок часу між вимірами повинен бути таким, щоб об'єкт, рухаючись з характерною для нього швидкістю, встигав проходити відстань не більш подвоєною точності позиціонування. Наприклад, щоб забезпечити позиціонування в реальному часі з точністю один метр людини, що має характерну швидкість пересування 1,5 метра в секунду (5,4 км/год), виміри треба проводити з періодичністю не менше одного разу кожні 1,3 секунди. Це дозволяє будувати досить точні для практичних цілей траєкторії руху об'єкта навіть при різких змінах швидкості та напрямку руху.

Важливе значення мають також:

надійність і живучість (здатність самовідновлюватися при виході з ладу будь-якого вузла);

малі габарити і вага, а також низьке енергоспоживання міток (з метою економії заряду акумуляторів).

До складу більшості типів RTLS зазвичай входять:

Активна мітка RTLS - радіоелектронний пристрій, які прикріплюються до контрольованих об'єктів і взаємодіють зі зчитувачами RTLS. Зчитувачі отримують сигнал від активних міток і вирішуючи тріангуляційну задачу визначають координати об'єкта.

Інфраструктура RTLS - базові станції обладнання забезпечує реперні точки з фіксованими координатами, об'єднаних мережею передачі даних і в деяких типах RTLS мережею синхронізації. Базова станція (БС) - пристрій, який взаємодіє з мітками в процесі визначення координат останніх. Базові станції мають фіксовані координати, щодо яких визначаються координати міток. Базові станції розташовуються так, щоб в будь-якій точці контрольованої території мітка могла «бачити» мінімум три базові станції.

Серверне програмне забезпечення - програмне забезпечення, що забезпечує управління процесом вимірювань, розрахунок координат об'єктів, обробку та накопичення даних.

Мітки в RTLS позиціонуються щодо базових станцій з відомими координатами. Координати обчислюються за допомогою:

трилатерації - обчислення координат за результатами вимірювання відстані від мітки до трьох базових станцій (БС);

мультилатерації (також відомої як гіперболічне позиціонування) - обчислення координат за результатами вимірювання відстаней від мітки до трьох або більше БС;

тріангуляції - обчислення координат шляхом вимірювання кутів напрямку від мітки до трьох БС.

Для підвищення точності і достовірності позиціонування використовуються складні алгоритми, що враховують наявність перешкод, обмежувачів руху (стін, бар'єрів), аттракторов (зручних, надають найменший опір шляхів), також в мітки може бути інтегрована інерціальна система навігації.

Суть процесу ідентифікації полягає в такому. Контрольовані системою об'єкти - люди, обладнання, транспортні засоби, рухомі механізми, інструменти, вантажі, цінні і небезпечні предмети та ін., забезпечуються мітками RTLS. Контрольована системою територія обладнується інфраструктурою RTLS. В процесі роботи мітки обмінюються з вхідними в інфраструктуру БС пакетами даних і в ході обміну вимірюють відстані до них (або кути напрямку на БС).

Серверне програмне забезпечення:

обчислює координати міток;

накопичує отримані дані;

сигналізує про знаходження об'єктів в заданих або заборонених зонах, рух об'єктів по заданих маршрутах або відхилення від них, порушеннях швидкісного режиму;

візуально відображає на екранах операторів місцезнаходження обраних об'єктів і траєкторії їх руху за заданий відрізок часу

2.2. Засоби вимірювань (датчики) в IoT

Особливу роль в «Інтернеті Речей» відіграють засоби вимірювання, що забезпечують перетворення відомостей про зовнішнє середовище в дані зрозумілі для ЕОМ, і тим самим здатні наповнити обчислювальне середовище цінною інформацією. Зараз використовується широкий клас засобів вимірювання, від елементарних датчиків (наприклад, температури, тиску, освітленості), приладів обліку споживання (таких, як інтелектуальні лічильники) до складних інтегрованих вимірювальних систем. В рамках концепції «Інтернету Речей» принциповим є об'єднання засобів вимірювання в мережі (такі, як бездротові сенсорні мережі, вимірювальні комплекси), за рахунок чого можлива побудова систем міжмашинної взаємодії [16].

Як особлива практична проблема впровадження «Інтернету Речей» наголошується на необхідності забезпечення максимальної автономності засобів вимірювання, перш за все, проблема енергопостачання датчиків. Знаходження ефективних рішень, що

забезпечують автономне живлення сенсорів (використання фотоелементів, перетворення енергії вібрації, повітряних потоків, використання бездротової передачі електрики), дозволяє масштабувати сенсорні мережі без підвищення витрат на обслуговування (у вигляді зміни батарейок або підзарядки акумуляторів датчиків).

Слід сказати, що на сьогоднішній день парадигма «Інтернету Речей» ще тільки набирає обертів, однак досить давно існує область науки, що займається здійсненням вимірів на відстані, яка носить назву *телеметрія* [17]. *Телеметрія* пов'язана з вимірюванням і передачею на відстань результатів вимірювань різних фізичних величин, більшість з яких є неелектричними. Вимірювання цих параметрів в основному здійснюється електричними методами. Переважне використання електричних методів вимірювання обумовлено:

зручністю перетворення електричних сигналів;

високою точністю і чутливістю перетворювачів фізичних величин в електричні сигнали;

малою інерційністю елементів електричного перетворювача;

надійністю роботи схем;

безперервністю вимірювання в часі;

зручністю запису і накопичення інформації;

широкими межами вимірювань і ін.

Як у телеметрії так і в «Інтернеті Речей» первинним джерелом інформації є датчик. Техніка конструювання і застосування датчиків (сенсорика), за останні роки розвилася у самостійну галузь вимірювальної техніки.

2.2.1. Загальні відомості про датчики

Датчиком називається інформаційний пристрій, що перетворює контрольований фізичний параметр в сигнал, зручний для подальшої обробки в каналі вимірювальної системи. Досить часто зустрічаються і інші назви датчиків: вимірювальний перетворювач, давач, детектор, вимірювач, чутливий елемент, зонд, сенсор, рецептор.

Датчик (рис. 2.10) включає в себе чутливий елемент і перетворювач неелектричних величин в електричні сигнали (ПНВЕС). Часто між чутливим елементом і перетворювачем включається передавально-розмножувальний механізм, необхідний для підвищення чутливості датчика. Деякі датчики забезпечуються підсилювальними схемами, що грають таку ж роль.

Рис. 2.10. Будова найпростішого датчика

У відповідності з ГОСТ 16263-70, “Сенсор” це пристрій, що називається первинним вимірювальним перетворювачем (primary measuring transducer), його частина на яку безпосередньо діє вимірювана величина, – чутливим елементом (detector), а всі наступні складові вимірювального ланцюга – вимірювальним перетворюванням (measuring transducer).

Взагалі розрізняють три класи датчиків:

аналогові датчики, тобто датчики, які виробляють аналоговий сигнал;

цифрові датчики, що генерують на виході двійковий код вимірюємої величини;

сигнальні (бінарні, двійкові) датчики, які виробляють сигнал тільки двох рівнів: (0/1).

Однак для «Інтернету Речей» можуть використовуватись лише цифрові та сигнальні датчики. При цьому будова датчика (рис. 2.11) дещо ускладнюється і він набуває певних «інтелектуальних» властивостей.

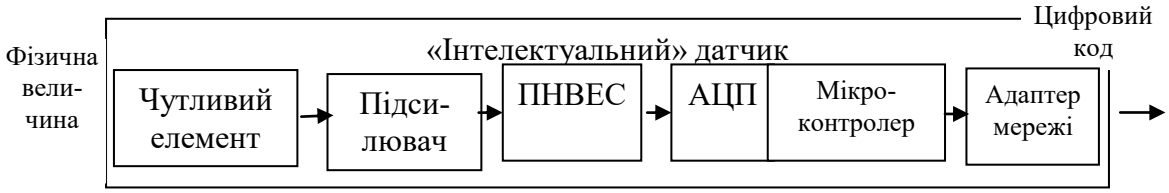


Рис. 2.11. Будова цифрового «інтелектуального» датчика

Електричний сигнал за допомогою АЦП перетворюється в цифровий код який може оброблятися та коректуватися мікроконтролером (часто АЦП входить до складу мікроконтролера) та за допомогою адаптера мережі пристосовуватись для передачі по Інтернету.

Данні сигнального датчика можуть одразу ж передаватись через адаптер мережі.

Важливим фактором для подальшої інтерпретації результатів вимірів є їх прив'язка до часу. Тобто доцільно разом з кодом вимірів фізичних процесів передавати і час їх виміру.

Інколи можливо чи доцільно організовувати вимірювальні системи (рис. 2.12) коли один мікроконтролер обслуговує декілька датчиків. У цьому випадку передбачають адресне розділення кожного вимірювального елемента.

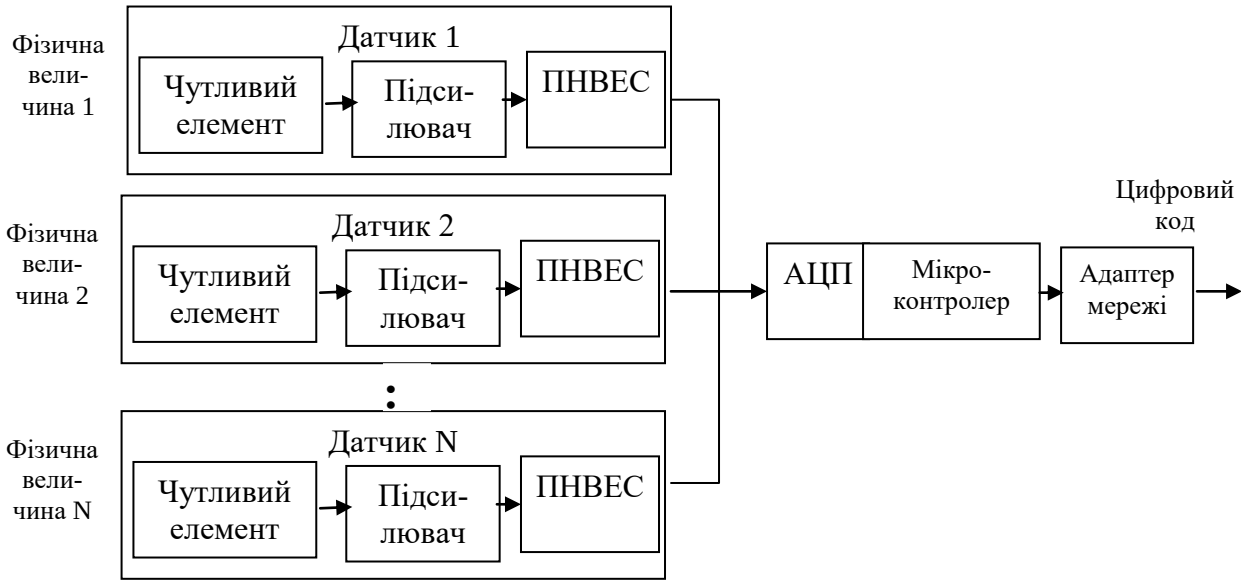


Рис. 2.12. Побудова вимірювальних схем

2.2.2. Основні характеристики (параметри) датчиків

Перейдемо до розгляду деяких характеристик датчиків. Вище зазначалося, що датчик можна вважати перетворювачем вимірюваної (фізичної) величини λ(t) у вихідний (електричний) сигнал s (t).

У динамічному режимі λ(t) і s(t) безупинно змінюються, і зв'язок між ними визначається диференціальним рівнянням, отриманим на основі фізичного принципу і схеми датчика [17]:

$$f_1\left[s^{(n)}, s^{(n-1)}, \ldots, s\right] = f_2\left[\lambda^{(m)}, \lambda^{(m-1)}, \ldots, \lambda\right] \quad (1.1)$$

У сталому (статичному) режимі вимірювання всі похідні λ і S обертаються в нуль і диференціальне рівняння переходить в алгебраїчне рівняння, що визначає статичну характеристику датчика,

$$f_1(s) = f_2(\lambda) \quad \text{або} \quad s = f(\lambda) \tag{1.2}$$

Тому прийнято виділяти статичні і динамічні характеристики датчиків.

2.2.2.1. Статичні характеристики датчиків

Статичні характеристики датчиків показують, наскільки коректно вихід датчика S відображає вимірювану величину λ через деякий час після її зміни, коли вихідний сигнал встановлюється у нове значення. Важливими статистичними параметрами є: чутливість, роздільна здатність, лінійність, дрейф, робочий діапазон, повторюваність і відтворюваність результату [17].

Статичну характеристику датчика прийнято називати *таріровочною (тарувальною) характеристикою*. Вона може бути лінійною і нелінійною. Дня зручності обробки інформації бажано мати лінійну характеристику s = *f*(λ). Лінеаризація проводиться зазвичай шляхом введення спеціальної корекції в різні ланки датчика.

Чутливість (sensitivity) датчика визначається як відношення величини зміни вихідного сигналу до одиничної зміни вхідної величини

$$S = \lim_{\Delta\lambda \to 0} \left(\frac{\Delta s}{\Delta\lambda}\right) = \frac{ds}{d\lambda} \tag{1.3}$$

Роздільна здатність (resolution) – це найменша зміна вимірюваної величини, котра може бути зафіксована і точно показана датчиком.

Точність (accuracy) визначає різницю між вимірюваною і дійсною величиною; вона може стосуватися датчика в цілому або конкретного його показника.

В процесі перетворення параметра λ в сигнал *s* виникають різні похибки, які складаються з методичних, динамічних та інструментальних похибок. У приладобудуванні точність датчиків оцінюють за допомогою приведеної відносної похибки, яка дорівнює відношенню абсолютної похибки до абсолютної величини діапазону вимірювання [18]:

$$\varepsilon = \frac{\Delta\lambda}{L_\lambda} \quad \text{або} \quad \frac{\Delta s}{L_s}.$$

Лінійність (linearity) не описується аналітично, а визначається виходячи з градаційної кривої датчика. Статична градаційна крива

показує залежність вихідного сигналу від вхідного при стаціонарних умовах. Наближення цієї кривої до прямої лінії і визначає ступінь лінійності.

Статичне підсилення (static gain) чи підсилення по постійному струму (d.c. gain) – це коефіцієнт підсилення датчика на дуже низьких частотах.

Дрейф (drift) визначається як відхилення показників датчика, коли вимірювана величина залишається постійною на протязі довготривалого моменту часу. Величина дрейфу може визначатися при нульовому, максимальному чи деякому проміжному значенні вхідного сигналу.

Деякі датчики через вплив сил тертя, люфтів і інших причин мають *зону нечутливості* (зону нульової чутливості). Для таких датчиків вводиться поняття порога чутливості, тобто мінімального збільшення параметра λ при якому сигнал s починає змінюватися.

2.2.2.2. Динамічні характеристики датчиків

Динамічні властивості датчика характеризуються цілим рядом параметрів, які, однак, доволі рідко наводяться в технічних описах виробників. З метою уніфікації методів динамічних випробувань і зіставлення характеристик різних датчиків прийнято реальний зовнішній вплив λ(t) замінювати *типовими вхідними діями*, основними з яких є *сходинкова*, *імпульсна* і *гармонійна*. Реакцію на типові впливи зазвичай визначають методом перетворення Лапласа при відомій передавальній функції датчика [19].

Динамічну характеристику датчика можна експериментально отримати з реакції на стрибок вимірюваної вхідної величини. Параметри, що описують реакцію датчика, дають уяву про його швидкодію (наприклад час нарощення, запізнювання, час досягнення першого максимуму), інерційних властивостей (відносне перерегулювання, час встановлення) і точності (зміщення).

Потрібно мінімізувати наступні параметри.

Час проходження зони нечутливості (dead time) – час між початком вимірювання фізичної величини і моментом реакції датчика, тобто моментом початку зміни вихідного сигналу.

Запізнювання (delay time) – час, через який показання датчика перший раз досягають 50 % значення, що встановилося. В літературі зустрічаються і інші визначення запізнення .

Час нарощення (rise time) – час, через який вихідний сигнал збільшується від 10 до 90% значення, що встановилося. Інше визначення часу нарощення – величина, зворотна нахилу кривої реакції

датчика на стрибок вимірювання величини в момент досягнення 50% від значення, що встановилося, помножене на значення, що встановилося.

Час досягнення першого максимуму (peak time) – час досягнення першого максимуму вихідного сигналу (перерегулювання).

2.2.3. Класифікація датчиків

Залежно від методу перетворення неелектричних величин в електричні сигнали розрізняють *параметричні* й *генераторні* датчики.

У параметричних датчиках зміна фізичної величини перетвориться в відповідну зміну будь-якого параметра електричного кола, що живиться від зовнішнього джерела. Такими параметрами зазвичай є опір, ємність або індуктивність [17].

У генераторних датчиках відбувається безпосереднє перетворення фізичних величин в електричні сигнали без використання зовнішніх джерел живлення.

За характером вихідних електричних сигналів розрізняють датчики *постійного* і *змінного* струму. Перші іноді ототожнюють тільки з потенціометричним датчиками, хоча клас датчиків постійного струму значно ширше.

За способом подання інформації датчики діляться на *аналогові* і *цифрові*.

В аналогових датчиках вихідний електричний сигнал є аналогом вимірюваної фізичної величини. Якщо параметр змінюється безперервно, то вихідний сигнал такого датчика зазнає такі ж зміни.

У цифрових датчиках при плавній зміні вхідної величини вихідний сигнал змінюється дискретно і представляється зазвичай у вигляді двійкового коду.

Більшість з вживаних в «Інтернеті Речей» датчиків відноситься до цифрових.

За видом отримуємої енергії датчики розділяють на: *активні* і *пасивні* перетворювачі. Пасивний отримує енергію від об’єкту дослідження (вимірюваної величини), наприклад, фотодіод. Активний – має зовнішнє живлення (наприклад, фоторезистор).

За розмірністю у просторі:

одиничні (точкові);

одномірні (розміщені вздовж однієї лінії, наприклад, датчик фотоприймачів сканера);

двомірні (наприклад, ПЗЗ – матриця цифрового фотоапарату);

об’ємні (набір шарів датчиків, наприклад, детектори космічних променів).

За видом дії на об’єкт виміру (локалізації):

контактні та ємнісні (для виміру фізичної величини необхідний контакт або близькість з нею).

дистанційні (випромінювач (не обов’язково) + приймач) здійснюють виміри на відстані.

За фізичним принципом дії (фізичним явищем) датчиків або за типом застосовуваних перетворювачів датчики діляться на: контактні (сигнальні), омічні, індуктивні, ємнісні, індукційні, фотоелектричні, термоелектричні, іонізаційні оптоелектричні, акустоелектричні, п’єзоелектричні, електромагнітні, магнітоелектричні і ін.

1) *Ємнісні перетворювачі*: зміна вхідної величини приводить до зміни ємності С, що змінює напругу електричного кола.

2) *Іонізаційні перетворювачі*. В газі, рідині або твердому тілі при дії іонізуючого випромінювання виникає струм іонізації між електродами.

3) *Електромагнітні перетворювачі*. Провідник рухається у магнітному полі, тому генеруються електрорушійна сила (ЕРС), наприклад – мікрофони.

4) *Електромеханічні перетворювачі* використовують механічний контакт, що керується зовнішнім впливом, наприклад – біметалічна пластинка), вимикачі – замикання і розмикання контакту.

5) *Магнітні, датчики на ефекті Холла*. Через напівпровідник, розміщений в магнітному полі, пропускають електричний струм. При цьому в поперечному напрямі до магнітного поля і струму виникає різниця потенціалів, яка пропорційна до напруженості магнітного поля (в магнітному полі розділяються електрони та дірки).

6) *П’єзоелектричні перетворювачі*. Механічна сила приводить до виникнення електричної напруги. П’єзоелектричні матеріали: природні (кварц); синтетичні (сульфат літія).

7) *Резистивні датчики* засновані на зміні опору певної області під дією зовнішньої дії (нагрівання, освітлення, вологість, деформація). Це наприклад: терморезистори та фоторезистори.

8) *Термоелектричні датчики* вимірюють фізичну величину на основі термоелектричного ефекту.

9) *Фотоелектричні перетворювачі (фотодіоди)*, засновані на явищі зовнішнього або внутрішнього фотоефекту, реагують на інфрачервоне, видиме, ультрафіолетове чи інше випромінювання в результаті чого генерується напруга пропорційна вхідній величині. Наприклад, ПЗЗ – матриці.

За фізичною природою вимірюваної величини чи області використання датчиків розрізняють датчики що вимірюють: електричні і магнітні характеристики; параметри переміщення; силу, момент і тиск; температуру; рівень заповнення ємності, витрати; щільність, в'язкість, консистенцію; концентрацію (газу, рідини, розчинених і зважених речовин); хімічну чи біохімічну активність та інші.

2.2.4. Приклади побудови датчиків

Термопари і температурні датчики

Датчики температури - це найбільш поширений тип датчиків. Вони застосовуються повсюдно: від інтелектуальних термостатів для холодних складів до охолоджувачів промислового обладнання, і, швидше за все, це перший сенсор, з яким ви зіштовхнетеся в IoT [1].

Термопара (ТС) - це пристрій для вимірювання температури, якому не потрібно джерело живлення, тому що воно саме генерує сигнал малої амплітуди (зазвичай мікровольт). Термопара - це два провідника, виготовлені з двох різних матеріалів, з'єднані в точці вимірювання температури. На металевому електроді, в залежності від його температури, виникає електричний потенціал. У різних металів рівень цього потенціалу різний. Цей ефект відомий як електрорушійний ефект Зеєбека, його суть полягає в тому, що різниця потенціалів між двома різними металами знаходиться в нелінійній залежності від їх температури (рис. 2.13).

Величина напруги залежить від властивостей вибраного металу. Вкрай важливо, щоб кінці проводів були термічно ізольовані від системи (і дроти повинні мати однакову контрольовану температуру).

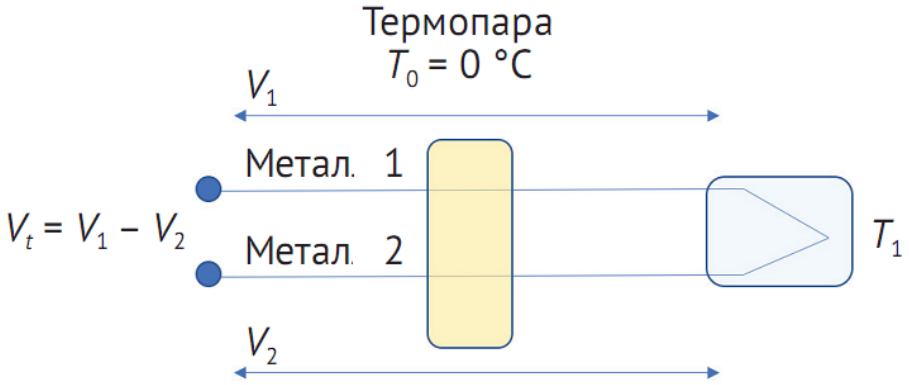


Рис. 2.13. Блок-схема вимірювання температури за допомогою термопари [1]

Термопари слід використовувати при виконанні не дуже відповідальних вимірювань, оскільки значення окремих термопар, при інших рівних умовах, можуть різнитися. Це викликано тим, що різні тонкі домішки, що входять до складу вимірювальних електродів, можуть призводити до невідповідності з довідковими таблицями. Можна, звичайно, скористатися високоточними (прецизійними) термопарами, але вони і коштувати будуть, відповідно, дорожче. Іншим ефектом, що впливає на точність вимірювань, є старіння. Так як термопари часто використовуються в промислових умовах, високотемпературні середовища з плином часу можуть погіршувати точність датчиків. Тому IoT-рішення повинні враховувати зміни, що відбуваються з датчиками в процесі їх експлуатації.

Термопари добре працюють в широкому діапазоні температур. Для різних комбінацій металів прийняті колірне і буквене маркування, що вказують тип термопари (наприклад, E, M, PT-PD). Зазвичай подібні датчики використовуються в промислових і високотемпературних середовищах при проведенні вимірювань в місцях, віддалених від оператора.

Таблиця 2.1

Основні типи термопар та їх характеристики

Тип термопари за МЕК*	Тип термопари за ДСТУ (ГОСТ)	Температурний діапазон °С (довготривало)	Температурний діапазон °С (короткотривало)
K	ТХА (хромель-алюмелеві)	0 до +1100	−180 до +1300
J	ТЖК (залізо-константанові)	0 до +700	−180 до +800
N	ТНН (ніхросил-нісилові)	0 до +1100	−270 до +1300
R	ТПП 13 (платинородій-платинові)	0 до +1600	−50 до +1700
S	ТПП 10 (платинородій-платинові)	0 до 1600	−50 до +1750
B	ТПР (платинородій-платинородієві)	+200 до +1700	0 до +1820
T	ТМКн (мідь-константанові)	−185 до +300	−250 до +400
E	ТХКн (хромель-константанові)	0 до +800	−40 до +900

На рис. 2.14 приведена залежність ЕРС від температури для декількох типів термопар.

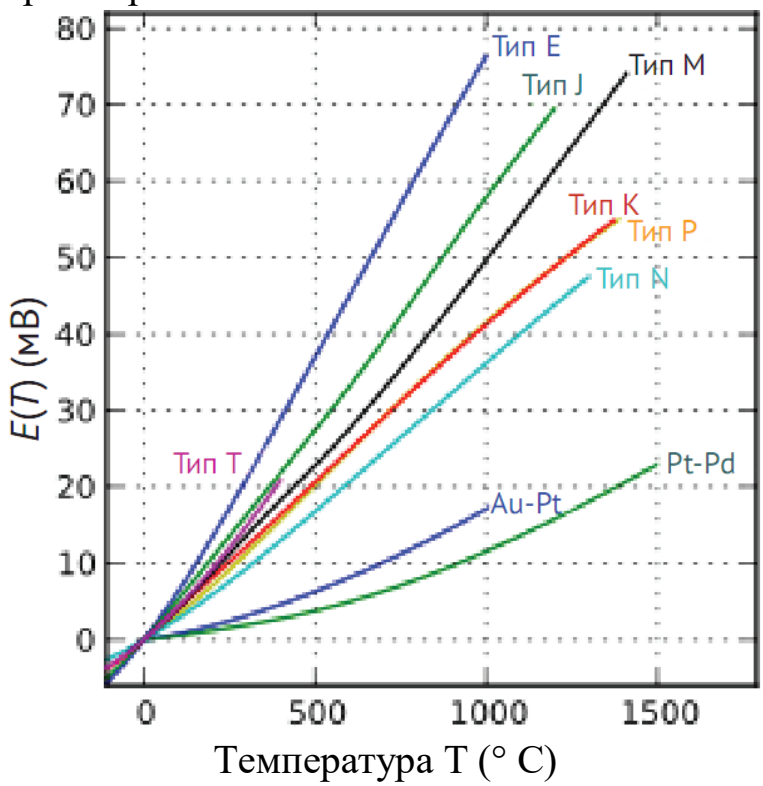


Рис. 2.14. Залежність ЕРС термопари *E*(*T*) від її температури T [1]

Резистивні датчики температури

Резистивні датчики температури (Resistance Temperature Detectors - RTD) працюють у вузькому діапазоні температур (нижче 600° С), але дозволяють виконувати вимірювання з більшою точністю, ніж термопари. Зазвичай вони виготовляються з дуже тонкого платинового дроту, щільно намотаного на керамічний або скляний сердечник. Електричний опір такої конструкції пропорційний її температурі. Оскільки в основі вимірів лежить вимірювання опору, для роботи з RTD необхідне зовнішнє джерело живлення з вихідною силою струму 1 мА. RTD виготовляються відповідно до прийнятих стандартів, в яких визначені допустимий діапазон і крок вимірів. Наприклад, для датчика 200 PT100 RTD крок вимірювань становить 0,00200 Ом/°С, а діапазон вимірювань лежить в межах від 0 до 100 °С. В межах цього діапазону залежність опору RTD від його температури зберігає лінійний характер. У відповідності зі стандартами RTD випускаються в двох-, трьох- і чотирьох провідних виконанні, чотирипровідні моделі використовуються виключно в системах високоточного калібрування. Для збільшення розрізнення вимірювань RTD часто використовують в

мостових схемах, при цьому зазвичай значення лінеарізуються програмно (рис. 2.15).

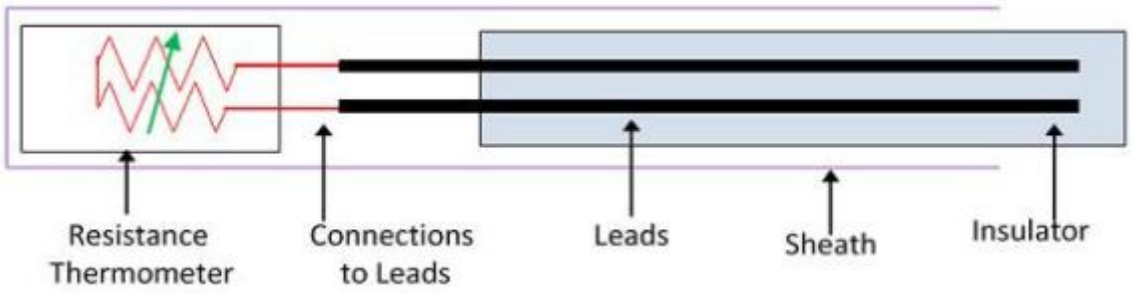


Рис. 2.15. Загальний вигляд резистивного датчика температури

Недоліки дротяних RTD: RTD рідко використовуються в діапазоні вище 600° C, що обмежує їх застосування в промисловості. При високих температурах платина може забруднюватися, що призводить до помилкових показань, однак при вимірах в межах заданого діапазону, RTD демонструють досить точні і стабільні результати.

Термістори

Термістор - це теж датчик температури, електричний опір якого залежить від його температури. Цей тип датчиків забезпечує більш високу точність вимірювань в порівнянні з RTD. По суті, це терморезистори, але з дуже нелінійною залежністю опору від температури. Їх часто використовують в якості згладжуючих фільтрів, для обмеження стрибків струму, а також у випадках, коли необхідна висока ступінь розрізнення вимірювань у вузькому діапазоні температур. Існує два типи термісторів: NTC (їх опір зменшується при підвищенні температури) і PTC (їх опір зростає з підвищенням температури). Основна відмінність від RTD полягає в тому, що термістори виготовляються з кераміки або полімерів, тоді як основою RTD завжди є метал.

Термістори знаходять застосування в медичній сфері і науковому обладнанні, в харчовій промисловості, інкубаторах і в таких побутових приладах, як термостати.

Ефект Холла і датчики струму

Датчик Холла - це смужка металу, через яку пропущений електричний струм. Потік заряджених частинок, що проходять через магнітне поле, відхиляється від прямолінійного напряму. Якщо напрямок магнітного поля перпендикулярно плоскому провіднику, то на його протилежних сторонах виникатиме різниця потенціалів,

обумовлена тим, що різнойменно заряджені частинки будуть збиратися на протилежних його сторонах.

Таким чином, одна сторона плоского провідника виявиться заряджена позитивно, а інша негативно, і виникне різниця потенціалів. Така різниця потенціалів називається напругою Холла, а сам ефект виникнення цієї напруги називається ефектом Холла. Це проілюстровано на рис. 2.16, наведеному нижче. Коли через металеву смугу (як показано на рис. 2.16), вміщену в магнітне поле, проходить струм, електрони притягуються до однієї її стороні, а дірки - в іншу (див. Криву на рис. 3.4).

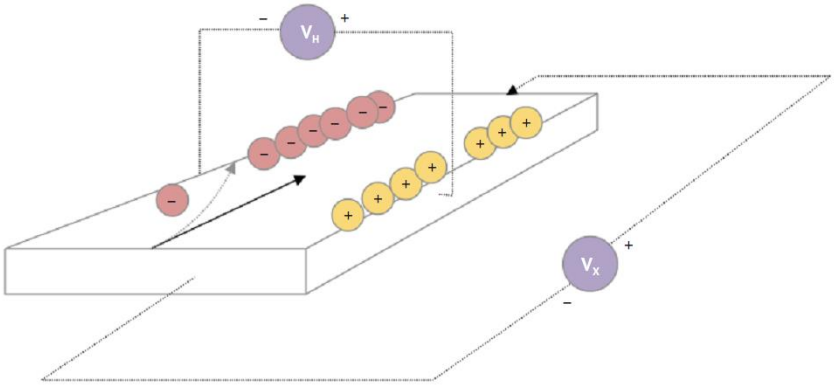


Рис. 2.16. Ілюстрація ефекту Холла [1]

Таке розшарування породжує електричне поле, яке можна виміряти. Якщо поле досить сильне, воно нейтралізує дію магнітного поля, і носії заряду зберігають прямолінійний рух:

Ефект Холла застосовується в датчиках струму, для вимірювання змінного і постійного струму. Існує два типи таких датчиків: з розімкненим і замкнутим контуром. Датчики з замкнутим контуром дорожчі, їх часто використовують у схемах з живленням від батарей.

Типова область застосування датчиків Холла: датчики положення, магнітометри, високонадійні перемикачі та покажчики рівня води. Вони використовуються також в промислових датчиках для вимірювання швидкості обертання різних вузлів і механізмів. Крім того, що ці датчики недорогі у виготовленні, вони не вимогливі до умов експлуатації та можуть працювати в найсуворіших умовах.

Фотоелектричні датчики

Датчики для виявлення світла або визначення його інтенсивності використовуються в багатьох пристроях IoT. Такі пристрої необхідні, наприклад, в системах безпеки, інтелектуальних комутаторах або в системах управління вуличним освітленням. Існує два типи таких датчиків, принцип дії яких зрозумілий з їх назви. Фоторезистор змінює

опір в залежності від інтенсивності світла, а фотодіод перетворює світло в електричний струм [1].

Фоторезистори виготовляються з напівпровідників з високим опором. Їх опір зменшується при збільшенні інтенсивності освітлення. У темряві опір фоторезистора може мати досить високий опір (порядку мегаом). Фотони, що поглинаються напівпровідником, переводять електрони в зону провідності, тим самим збільшуючи провідність матеріалу. Фоторезистори чутливі до довжини хвилі падаючого світла, тому їх типів і модифікацій існує величезна безліч. А ось фотодіоди - це повноцінні напівпровідникові прилади з p-n-переходом. Такі пристрої реагують на світло, створюючи електронно-діркову пару. Потік дірок, що рухаються до анода, і електронів, що рухаються до катода, створює електричний струм. Таким чином працюють традиційні сонячні батареї, що виробляють електрику під впливом сонячних променів. Якщо на фотодіод подати зворотну напругу, то можна регулювати його чутливість або час відгуку.

Датчики PIR

Піроелектричний інфрачервоний датчик (Pyroelectric Infrared - PIR) складається з двох слотів, заповнених матеріалом, який реагує на інфрачервоне випромінювання і тепло. Типові варіанти застосування таких датчиків - це теплові датчики руху систем безпеки. Зазвичай такі датчики оснащуються лінзою Френеля, за допомогою якої формується зона виявлення. Така зона має форму арки, що розкривається назовні. Коли тепле тіло входить або, навпаки, залишає зону виявлення, чутливі елементи формують електричний сигнал. В PIR-датчиках використовуються кристалічні матеріали, які здатні генерувати електричний струм під впливом інфрачервоного випромінювання. Все разом це утворює так званий польовий транзистор (Field Effect Transistor - FET), який фіксує зміну струму і посилає сигнал на підсилювальний пристрій. PIR-датчики добре працюють в діапазоні хвиль від 8 до 14 мкм, цей діапазон характерний для випромінювань людського тіла.

На рис. 2.17 нижче зображені два елементи PIR, що формують дві зони виявлення. Це дозволяє не тільки відслідковувати весь простір кімнати, але і визначати напрямок переміщень.

Для сканування більшої площі за допомогою одного датчика буде потрібно кілька лінз Френеля, які будуть фокусувати на PIR зображення окремих областей відслідковуємої території. Таке фокусування забезпечує концентрацію інфрачервоного випромінювання безпосередньо в області FET. Як правило, такі пристрої дозволяють проектувальнику контролювати чутливість (діапазон) і час реакції.

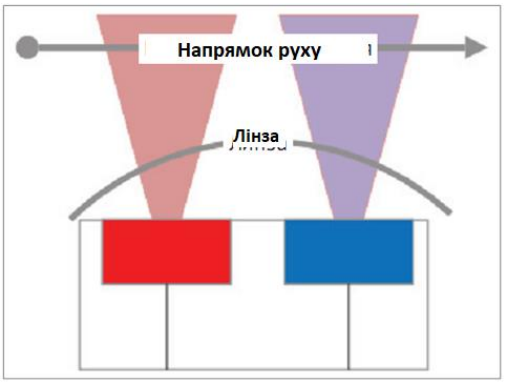


Рис. 2.17. Датчик PIR. Два елементи реагують на джерело ІЧ-випромінювання, що рухається в зоні виявлення [1]

Час реакції вказує, через який проміжок часу буде посланий сигнал після виявлення руху на території, що охороняється. Чим менше час реакції, тим більше подій може бути зафіксовано. На рис. 2.18 приведена діаграма типового PIR-датчика, оснащеного лінзою Френеля з фіксованою фокусною відстанню і сфокусованої на підкладку.

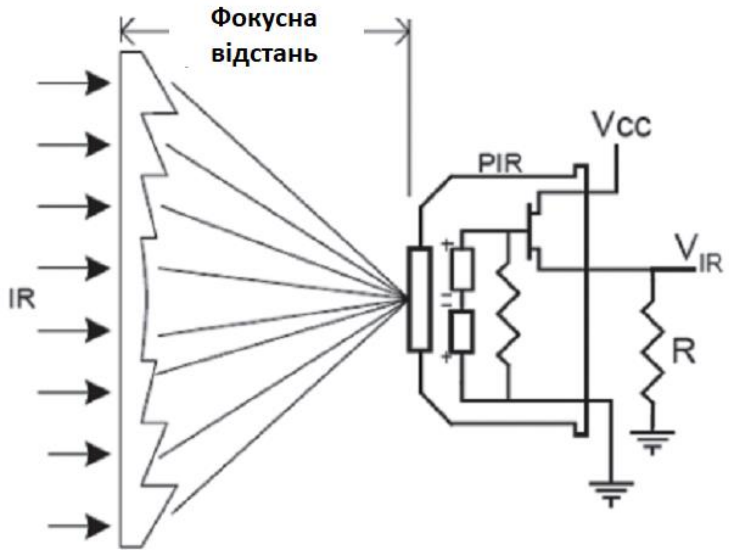


Рис. 2.18. Схема по застосуванню PIR Cypress Microsystems AN2105. Зліва: лінза Френеля фокусує ІЧ-випромінювання на PIR – датчик [1]

2.2.5. Датчики MEMS (Micro-Electro-Mechanical Systems)

На сьогоднішній день однією з інноваційних технологій у розвитку датчиків є технологія мікроелектромеханічних систем - MEMS (Micro-Electro-Mechanical Systems) [18].

Визначення: Під технологією MEMS розуміють технологію мікрообробки, що дозволяє виготовляти кремнієві мікросхеми з крихітними механічними елементами – інтелектуальними машинами з самими різними функціями. Відповідно MEMS – це об'єднання механічних елементів, датчиків, приводів і електроніки на одній кремнієвій пластині (підкладці).

Світовий ринок MEMS є дуже динамічним і згідно останнім прогнозам росте на 13,2% щороку. До речі, цю галузь індустрії в Японії називають мікромашинами (Micromachines), а в Європі - мікросистемними технологіями (Micro System Technology).

Всі елементи мікроелектромеханічних систем можуть бути реалізовані у вигляді єдиного виробу, причому відразу десятками або сотнями, як мікросхеми на кремнієвій пластині, в основі цього лежить апробована традиційна технологія виробництва напівпровідникових інтегральних мікросхем. [19]

Багато експертів, включаючи фахівців однієї з провідних фірм в цій області - Integrated Sensing Systems (http://www.mems-issys.com), - вважають, що MEMS-технологія привносить буквально революційні зміни в кожну сферу застосування шляхом поєднання мікроелектроніки на основі кремнію з мікромеханічною технологією, що дозволяє реалізувати систему на одному кристалі SOC (Systems-on-a-Chip). Так, технологія MEMS дала новий імпульс розвитку систем інерціальної навігації і інтегрованих систем, відкривши шлях до розробки "розумних" виробів, збільшивши обчислювальні здібності мікродатчиків і розширивши можливості дизайну таких систем.

Сьогодні MEMS-пристрої застосовуються практично всюди. Це можуть бути мініатюрні деталі (гідравлічні і пневмоклапани, струменеві сопла принтера, пружини для підвіски головки вінчестера), мікроінструменти (скальпелі і пінцети для роботи з об'єктами мікронних розмірів), мікромашини (мотори, насоси, турбіни величиною з горошину), мікророботи, мікродатчики і виконавчі пристрої, аналітичні мікролабораторії (на одному кристалі) і так далі.

Найважливіша складова частина більшості MEMS - мікроактуатор. Зазвичай даний пристрій перетворює енергію в керований рух. Розміри мікроактуаторів можуть досить сильно варіюватися. Діапазон застосування цих пристроїв надзвичайно широкий і при цьому постійно росте. Всі методи активації (рух, деформація, приведення в дію) в таких пристроях коротко можна звести до наступних: електростатичний, магнітний, п'єзоелектричний, гідравлічний і тепловий. При оцінці використання того або іншого методу часто застосовують закони пропорційного зменшення розмірів. Найбільш перспективними

методами вважаються п'єзоелектричний і гідравлічний, хоча і інші мають велике значення. Електростатична активація застосовується приблизно в одній третині мікроактуаторів, і це, ймовірно, найбільш загальний і добре розроблений метод; головні його недоліки - знос і злипання. Магнітні мікроактуатори зазвичай вимагають великого електричного струму на мікроскопічному рівні. При використанні електростатичних методів активації отримуваний вихідний сигнал на відносну одиницю розмірності кращий, ніж при використанні магнітних методів. Іншими словами, при одному і тому ж розмірі електростатичний пристрій видає декілька кращий вихідний сигнал. Теплові мікроактуатори теж споживають відносно багато електричної енергії; головний їх недолік полягає в тому, що тепло, що генерується, доводиться розсіювати.

Для оцінки мікроактуаторів використовують такі критерії якості, як лінійність, точність, погрішність, повторюваність, розрізнення, гістерезис, порогове значення, люфт, шум, зрушення, що несе здатність, амплітуда, чутливість, швидкість, перехідна характеристика, масштабованість, вихід по енергії [8].

Таким чином, цінність IoT в тому, що це комплексне рішення, а дані, що надаються датчиком, грають в цьому комплексі ключову роль. Тому проектувальнику необхідно усвідомлювати, що це за дані і як їх правильно інтерпретувати.

Крім розуміння того, які дані збираються і як вони утворюються в масиві IoT, корисно знати, що саме і в яких межах вимірюється. Наприклад, система повинна враховувати пристрої, які втратили зв'язок, і помилкові дані.

Проектувальник повинен розуміти причини, за якими дані, отримані від датчиків, можуть бути ненадійними, а також причини, внаслідок яких польовий датчик може вийти з ладу.

Кількість датчиків і виконавчих пристроїв, об'єднаних в єдину мережу, значно зросла, тому проектувальнику надзвичайно важливо розуміти їх взаємодії. Кожен проектувальник повинен запитати себе: «Який тип датчика або кінцевого пристрою слід використовувати для вирішення проблеми, яка стоїть переді мною? »

2.3. Виконавчі пристрої в IoT

Без виконавчих пристроїв (інколи їх називають «актуатори») Інтернет речей не зміг би самостійно вносити необхідні зміни у навколишній світ, а отже, зводився до простого управління та взаємодії з різними пристроями. Вихідні пристрої в IoT можуть бути практично будь-якими: від простого світлодіода до повноцінної відеосистеми чи потужного промислового приводу (рис. 2.19). Зрозуміло, що ці пристрої потребують різних систем керування різного рівня складності та потужності. Залежно від типу виходу та використовуваного варіанту застосування, також слід очікувати, що більша частина контролю і управління повинна проводитися безпосередньо поблизу пристрою (на противагу повного контролю в хмарі). Наприклад, відеосистема може передавати дані до хмарних провайдерів, але для цього потрібно обладнання виведення і буферизації [8].

У загальному випадку системам виведення потрібні значні обсяги електроенергії для її перетворення в механічний рух, теплову енергію або в світло. Наприклад, невеликий соленоїд для управління потоком рідини або газу при напрузі живлення 9-24 В споживатиме струм приблизно в 100 мА, при цьому створить направлене зусилля всього в п'ять ньютонів. А промислові соленоїди працюють від напруги в сотні вольт.

а)

б)

в)

Рис. 2.19. Приклади механізмів виводу: а – сервоприводи; б - крововий двигун; в – промислові клапани

Виконавчі пристрої можуть бути автономними (тобто пристроєм виведення) або можуть поєднуватися з вхідним датчиком IoT. Прикладом може служити інтелектуальна лампочка, призначена для нічного освітлення на відкритому повітрі, коли датчик визначає, що навколишнє світло впало до заданого рівня (який може бути запрограмований ззовні), і на додачу до передачі цих даних вгору за течією також безпосередньо запускає виконавчий механізм (сама лампочка) увімкнути. У багатьох випадках виконавчий механізм, крім обробки даних, відправлених йому по мережі IoT, також повертатиме додаткові дані, тому в певному сенсі може містити як датчик, так і виконавчий механізм. Приклад, знову ж таки з використанням лампочки: лампочка включається лише за спеціальною вказівкою зовнішніх даних, але якщо світловий елемент виходить з ладу, лампочка повідомить мережі, що цей пристрій більше не здатний випромінювати світло, навіть якщо воно отримує дані. Надійно спроектована мережа також вимагатиме використання актуаторів лампочок, які іноді видають «пульс», тому, якщо блок лампи повністю вийде з ладу, мережа дізнається про це і повідомить про відмову.

2.4. Живлення пристроїв IoT

Використання як малопотужних датчиків і так і потужних виконавчих пристроїв пов'язано з однією проблемою - живлення. Коли ми говоримо про мільярди елементів IoT, зрозуміло, що деякі з них будуть встановлені в дуже віддалених районах, і тоді їх живлення стає дуже проблематичним. Крім того, при розгортанні IoT деякі датчики можуть розташовуватися під водою або вбудовуватися в існуючі інфраструктури, що ще більше ускладнює питання живлення. Тому питанням концепції виробництва електроенергії та управління живленням в IoT приділяється надзвичайно велика увага.

Управління живленням - дуже широка тема, яка стосується як програмного, так і апаратного забезпечення. При розгортанні IoT дуже важливо розуміти роль ефективного управління енергоспоживанням віддалених пристроїв і пристроїв довготривалої експлуатації та володіти прийомами управління ними.

Проектувальник системи IoT повинен будувати бюджет енергоспоживання, з огляду на наступні фактори [1]:

активна потужність датчика;

частота збору даних;

споживана потужність бездротового радіозв'язку, необхідна для забезпечення покриття заданої площі;

частота зв'язку;

потужність мікропроцесора або мікроконтролера в залежності від тактової частоти ядра;

потужність пасивної складової;

втрати енергії від витоків або неефективності живлення;

резервування електроенергії для живлення приводів і двигунів.

Бюджет просто відображає сумарне енергоспоживання цих споживачів, що забезпечується джерелом живлення (батареєю). З плином часу режим роботи батареї не лінійний. Оскільки батарея втрачає енергоємність при розряджанні, її напруга падає криволінійно. Це створює проблеми для систем бездротового зв'язку. Якщо акумулятор розрядиться до деякої критично мінімальної напруги, радіолінія або мікропроцесор, не маючи порогової напруги живлення, просто відключиться.

На сьогодні застосовують різноманітні методи управління живленням, наприклад такі як: *використання неелектронних реле часу, зниження тактової частоти процесорів або мікроконтролерів, регулювання частоти, обмеження ширини смуги радіомовлення, стратегії відстрочки сеансів зв'язку і різні режими сну*. Ці методи широко використовуються, а в обчислювальній техніці вже стали загальноприйнятою практикою. Вони мінімізують енергоспоживання за допомогою динамічного управління напругою, частотного регулювання та інших схем. Але існують і нові методи, з якими слід познайомитися [ ]: це *наближений розрахунок* та *імовірністне проектування*. Обидві ці схеми засновані на тому факті, що абсолютна точність в показаннях датчика не завжди необхідна, особливо якщо сигнал підлягає обробці для передачі за допомогою бездротового зв'язку. Округлення обчислень може виконуватися на апаратному або програмному рівні простим округленням до цілого. Це тим більш прийнятно, коли мова йде про адреси або про множення чисел (наприклад, значення 18,962 досить близько до 18,970). *Імовірністне проектування* передбачає лише задану ступінь надійності, а не максимально можливу, що знімає багато конструктивних обмежень. Обидва методи в сукупності здатні знизити енергоспоживання майже експоненціально в порівнянні зі звичайними схемами.

Відтворення електроенергії також концепція не нова, але в плані IoT дуже важлива. По суті, будь-яка система, що стежить за змінами умов навколишнього середовища (наприклад, від спеки до холоду, отримання бездротових сигналів, освітленість), може перетворювати форми енергії, що спостерігаються, в електричну. У деяких пристроях це

використовується як єдине джерело енергії, а є і гібридні системи, які використовують таке перетворення для підзарядки батарей і продовження їх терміну служби. З іншого боку, вироблена таким чином енергія може зберігатися і використовуватися ощадливо для живлення низькоенергетичних пристроїв, наприклад датчиків в IoT. У будь-якому випадку, системи повинні бути ефективні при перетворенні і збереженні енергії. Це вимагає розширених можливостей в управлінні живленням. Наприклад, якщо система виробництва електроенергії використовує технологію п'єзоелектричної підкладки, вбудованої в тротуар, то повинна бути передбачена компенсація на випадок, коли пішохідний рух буде недостатнім. Постійний зв'язок з перетворювачами енергії теж веде до збільшення енергоспоживання, яке також треба компенсувати [1].

При розгортанні IoT, як правило, використовуються передові технології управління живленням, що виключають повну втрату функціональності систем. Часто використовуються такі методи, як очікування в режимі зниженого енергоспоживання, скорочення втрат, періодичні включення і виключення з використанням реле часу. На рис. 2.20 нижче показана область енергоспоживання, яка ідеально підходить для живлення від перетворювачів, а також технології, які можуть це забезпечити. Проектувальник повинен подбати про те, щоб система не відчувала дефіциту в електроенергії, але і не мала її в надлишку.

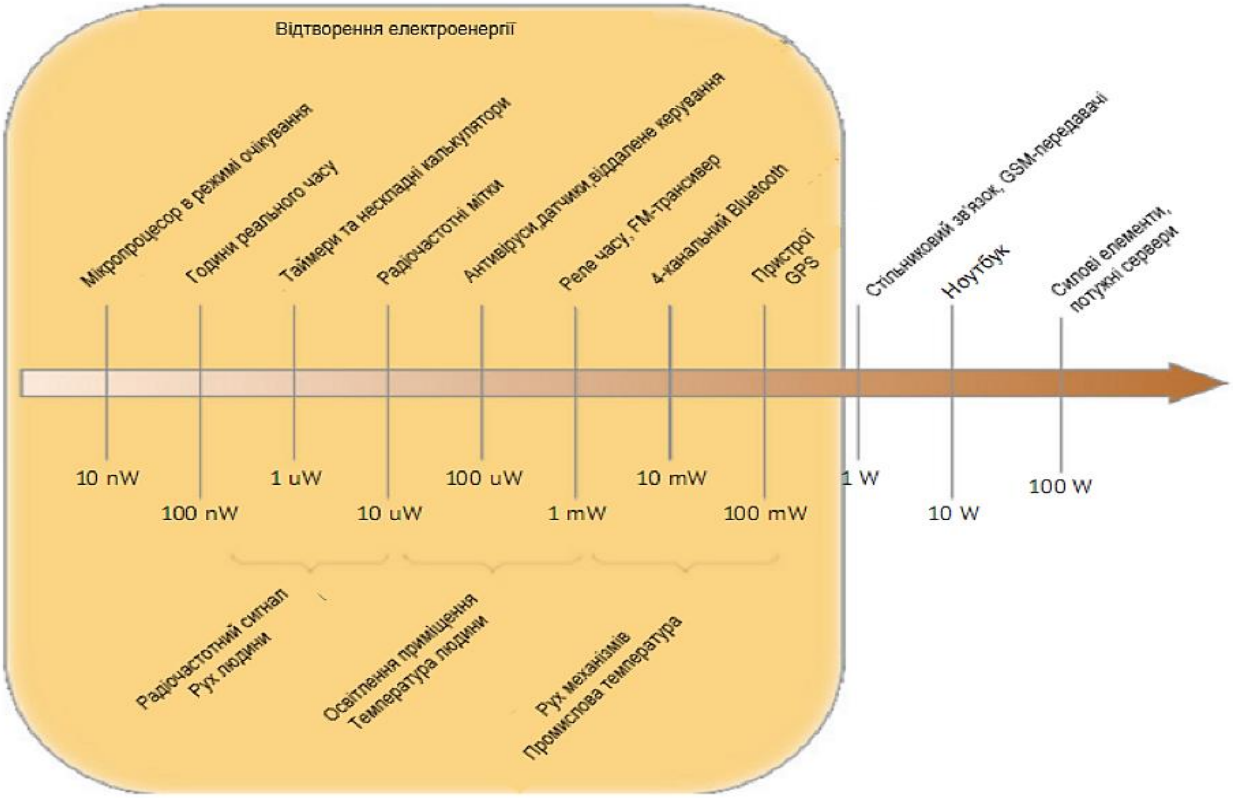


Рис. 2.20. Область енергоспоживання, що підходить для живлення відтвореною електроенергією. На малюнку показано типове споживання енергії для різних пристроїв [1]

Системи перетворення і відтворення електроенергії, в цілому, мають низький потенціал і ефективність. Тому питання про відтворення електроенергії доцільно розглядати лише в умовах дійсного надлишку невикористаної енергії, наприклад, в умовах промислового виробництва.

Перетворення сонячної енергії. Енергія природного або штучного світла легко може бути використана як джерело електроенергії. Наприклад. фотодіоди можна використовувати для створення традиційних сонячних батарей. Їх здатність генерувати енергію безпосередньо залежить від їх площі. Найбільш ефективні батареї, встановлені під прямими сонячними променями, а не в умовах штучного освітлення. Ефективність перетворення сонячної енергії залежить від сонячної активності, яка змінюється сезонно і географічно. Класифікуються такі панелі по максимально можливій потужності на вході, що оцінюється у Вт.

П’єзомеханічне перетворення. Як уже згадувалося раніше, п'єзоелектричні ефекти можуть використовуватися в якості датчиків, але вони ж можуть бути використані і для генерації енергії. Механічні деформації, викликані рухом, вібрацією і навіть звуком можуть бути перетворені в електроенергію. Такі генератори можна використовувати в розумних дорогах, навіть якщо вони вбудовані в бетон. Подібні пристрої генерують потужність порядку міліват і, отже, підходять для дуже невеликих систем, призначених для збору і зберігання енергії. Весь процес може бути виконаний із застосуванням п'єзомеханічних пристроїв MEMS, електростатичних і електромагнітних систем.

Принцип дії електромагнітної системи заснований на законі Фарадея, який стверджує, що зміна магнітного потоку через котушку з проводом індукує в ній електричний струм. Щоб забезпечити такі зміни, вібрація передається або на котушку, або на магніт. На жаль, така схема забезпечує занадто малу напругу.

Електростатичні системи використовують ефект, що виникає при зміні відстані між двома ємнісними пластинами, на яких підтримується постійна напруга або заряд. Будь-яка вібрація пластин викликає зміну відстані між ними, енергія (E), що виникає при цьому, може бути проілюстрована за допомогою наступної моделі:

$$E = \frac{1}{2} Q V^2 = \frac{Q^2}{2C}$$

де Q - постійний заряд на пластинах, V - постійна напруга, C - електрична ємність.

Перевага електростатичного перетворення полягає в тому, що воно може бути масштабованим і економічно доцільним в умовах мікромашино-будування і виробництва напівпровідників.

Останній метод механіко-електричного перетворення - це п'єзомеханічний метод, про який вже згадувалося. П'єзомеханічний пристрій MEMS, протидіючи прикладеному до нього зусиллю, генерує електричну енергію.

Перетворення радіочастот. Бездротове енергопостачання за допомогою радіочастот (RF) ведеться вже протягом багатьох років, наприклад радіомітки (RFID). RFID, як правило, знаходиться в безпосередній близькості від приймача, який не тільки обмінюється з нею інформацією, а й постачає її енергією. Але віддаленим пристроям потрібно відтворювати електроенергію з широкомовних трансляцій. Передача в широкомовному діапазоні ведеться майже повсюдно: це і телебачення і сигнали стільникового зв'язку, і радіо.

Відтворення електроенергії з радіочастот - це найбільш важке завдання в порівнянні з іншими формами генерації, оскільки радіочастотні сигнали мають найменшу щільність енергії. Прийом радіочастотних сигналів проводиться за допомогою антени, яка налаштована на певну смугу частот. Типовий діапазон частот, що використовується для радіомовлення, становить від 531 до 1611 кГц (діапазон радіочастот АМ).

Перетворення тепла. Теплова енергія будь-якого пристрою, якому потрібне відведення теплоти, може бути перетворена в електроенергію. Зазвичай для цього використовуються два основні процеси:

термоелектричний: пряме перетворення теплової енергії в електричну енергію внаслідок ефекту Зеєбека;

термічний: також відомий як термотунелювання. Електрони викидаються (емітуються) з нагрітого електрода і в напрямку холодного.

Термоелектричний ефект (ефект Зеєбека) виникає в провідних матеріалах при наявності градієнта температур. Потік носіїв з гарячою в холодну область між двома несхожими електричними провідниками створює різницю потенціалів. Термопара, або термоелектричний генератор (ТЕГ), може ефективно генерувати напругу навіть внаслідок різниці температури людського тіла і температури навколишнього середовища. Різниця температур в 5°С може генерувати потужність в 40 мкВт при напрузі 3 В. Коли тепло передається від гарячого електрода до холодного, гаряча сторона індукує потік електронів в напрямку холодного електрода. Енергія, вироблена термопарою, пропорційна квадрату напруги і різниці температур між електродами. Напруга, що виробляється, як правило, дуже мала, тому для утворення термоелемента послідовно з'єднується кілька термопар [1].

Однією з істотних проблем сучасних термопар є їх низька ефективність перетворення енергії (менше 10%). Однак і їх переваги очевидні: це і невеликі розміри, і простота виготовлення, і невисока вартість. Тривалість терміну їх служби - більше 100 000 годин.

Основна ж проблема полягає в знаходженні щодо постійного джерела теплової дисперсії (різниці температур). Використання такого пристрою протягом декількох сезонів з різними температурами є складним завданням. Потужність, що генерується, для пристроїв IoT, зазвичай лежить в діапазоні до 50 мВт.

Сховища енергії. Типовим сховищем енергії для датчика IoT служать батарея або суперконденсатор (іоністор). При розгляді архітектури потужного датчика необхідно враховувати кілька аспектів [6]:

енергія, необхідна для живлення силової підсистеми: чи зуміє батарея її забезпечити;

енергоємність акумулятора;

доступність: якщо пристрої замуровані в бетон, чи буде доцільна дорога заміна, якщо вони вийдуть з ладу, тим більше, що їх можливості регенерації енергії вельми скромні;

вага: цей параметр дуже критичний, якщо пристрій призначений, наприклад, для безпілотного літального апарату;

як часто буде заряджатися акумулятор;

чи є відновлювальна форма енергії постійно доступною або епізодичною, як, наприклад, сонячна;

характеристики потужності батареї: як енергія батареї буде змінюватися з часом, в процесі розрядки;

чи встановлений датчик в термічно обмеженому середовищі, що може вплинути на термін служби батареї і її надійність;

чи гарантує батарея мінімально допустимий струм.

На сьогоднішній день, літій-іонна (Li-ion) батарея, завдяки її щільності енергії, стала стандартним елементом живлення в мобільних пристроях. У такій батареї іони літію фізично переміщаються від негативного електрода до позитивного. Під час перезарядки іони повертаються в негативну область. Це явище відоме як іонний рух.

Батареї схильні до старіння. Це виражається у втраті ємності від початкової після деякої кількості циклів перезарядки (наприклад, втрата 30% ємності після 1000 циклів). Це погіршення майже безпосередньо корелює з температурою навколишнього середовища, в теплих середовищ втрати будуть рости. Тому проектувальнику необхідно подбати про дотримання температурного режиму, якщо він використовує літій-іонну батарею.

Іншим негативним фактором автономного режиму є саморозряд. В процесі експлуатації в батареї виникають небажані хімічні реакції, які

ведуть до втрати енергії. Величина і швидкість таких втрат залежить від хімічного складу і температури. Як правило, термін служби літій-іонного елемента становить 10 років (з втратою ~ 2% в місяць), а лужної батареї - тільки 5 років (15-20% втрат в місяць).

Суперконденсатори (або іоністори) можуть зберігати енергію в значно більших обсягах, ніж звичайні конденсатори. Звичайний конденсатор має густину енергії близько 0,01 Вт год/кг. Іоністор має щільність енергії від 1 до 10 Вт год/кг, за цим параметром він наближається до батарей, щільність енергії яких може досягати 200 Вт год/кг. Як і конденсатор, іоністор зберігає енергію в електростатично заряджених пластинах і, на відміну від акумулятора, не використовує хімічне перенесення енергії. Зазвичай іоністори виготовляються з досить екзотичних матеріалів, таких, наприклад, як графен, що, природно, впливає на їх вартість. Очевидна перевага іоністорів - їх швидкість зарядки. До повного потенціалу вони заряджаються за секунди, в той час як, щоб зарядити літій-іонні батареї на 80%, буде потрібно кілька хвилин. Крім того, іоністор неможливо перезарядити, а ось надмірне заряджання літій-іонної батареї може призвести до серйозних проблем безпеки. Іоністори бувають двох видів [1]:

електричні двошарові конденсатори (Electric double-layer capacitors - EDLC): його обкладки виготовляються з активованого вугілля, а енергія зберігається у вигляді електростатичного потенціалу;

псевдоконденсатори (Psuedocapacitors): в якості діелектрика між його обкладинками використовується оксид металу, з якого виготовлені обкладання, для збереження енергії використовується електрохімічний перенесення заряду.

Ще одна перевага іоністорів перед батареями - можливість точного прогнозування часу їх служби. Остаточний обсяг енергії іоністора легко розрахувати по напрузі на його клемах, вона змінюється з плином часу. На відміну від літій-іонної батареї, у якій профіль напруги зберігається постійним, що ускладнює оцінку решти обсягів енергії. Оскільки профіль напруги іоністора змінюється з плином часом, для компенсації виникаючих змін необхідний DC-DC перетворювач.

Основними проблемами іоністора, як і будь-якого іншого конденсатора, є витікання струму і вартість.

Таким чином, питання енергозабезпечення проектованих систем мають ключове значення, і це повинно бути зрозуміло проектувальникам. Погано спроектована система енергопостачання призводить до занадто короткого часу автономної роботи системи в цілому, а це тягне за собою істотні витрати на виправлення і доопрацювання.

Контрольні питання до розділу 2

1. Які методи ідентифікації ви знаєте?
2. В чому полягає суть оптичних методів ідентифікації?
3. В чому полягає суть радіочастотних методів ідентифікації?
4. Що називається датчиком в Інтернеті речей?
5. Назвати основні характеристики датчиків.
6. Навести класифікацію датчиків.
7. Призначення та особливості виконавчих пристроїв IoT?
8. Особливості забезпечення живлення пристроїв IoT?

3. ТЕХНОЛОГІЇ ПЕРЕДАЧІ ДАНИХ IoT

Одним із головних питань організації Інтернету речей є реалізація взаємодії між:

інтернет-речами,

користувачами та інтернет-речами,

віддаленим сервером та інтернет-речами.

IoT використовує велику кількість варіантів мереж зв'язку для передачі даних, починаючи від мережі на тілі людини BAN (Body Area Network), яка працює на відстані кілька десятків сантиметрів, аж до всесвітньої мережі Інтернет. Комунікації малої дальності використовують такі технології, як RFID, NFC, Bluetooth, Wi-Fi та ін. Комунікації великого радіусу дії реалізуються з урахуванням різних стільникових мереж (3G/4G/5G), мереж бездротового широкосмугового доступу WiMAX, мереж позиціонування GPS/ГЛОНАСС та ін.

По території охоплення телекомунікаційні мережі, що використовуються в Інтернеті речей, можна поділити на 4 основні типи [1]:

1. Персональна мережа PAN (Personal Area Network) - це мережа, побудована "навколо" людини. Дані мережі мають об'єднувати всі персональні пристрої користувача (телефони, смартфони, кишенькові персональні комп'ютери, ноутбуки, гарнітури та ін.).

Стосовно IoT така мережа будується «навколо» пристрою («речі»).

2. Локальна мережа LAN (Local Area Network) – мережа, що зазвичай покриває відносно невелику територію або невелику групу будівель (будинок, офіс, фірму). До локальних мереж можна віднести і мережу контролерів CAN (Controller Area Network) – промислову мережу, орієнтовану насамперед на об'єднання в єдину мережу різних виконавчих пристроїв та датчиків у рамках окремого підприємства.

3. Міська мережа MAN (Metropolitan Area Network) – об'єднує окремих користувачів та локальні мережі в межах міста, є мережею за розмірами більшою, ніж LAN, але меншою, ніж WAN.

4. Глобальна мережа WAN (Wide Area Network) – пов'язує користувачів та мережі, розосереджені на відстані сотень та тисяч кілометрів.

Інтернет речей практично не висуває особливих вимог до технологій LAN, MAN і WAN, крім того, вони досить добре висвітлені в технічній літературі та вивчаються на інших дисциплінах. Тому в цьому розділі розглянуті лише стандарти та протоколи мереж малого та середнього радіусу дії, які широко використовуються в IoT.

Всі технології передачі даних в Інтернеті речей залежно від використовуваного середовища передачі можна розділити на два великі класи: дротові та бездротові.

Дротові технології передачі даних у IoT можуть використовувати металевий (мідний) кабель зв'язку, електропроводку (технологія PLC – Power Line Communication) або волоконно-оптичний кабель. Однак через складності фізичної реалізації ліній зв'язку провідні технології для комунікацій інтернет-речей застосовуються меншою мірою, ніж бездротові. Бездротові мережі малого радіусу дії, що використовуються в IoT, можна розділити на три види:

1. Бездротові персональні мережі WPAN (Wireless Personal Area Network). Застосовуються для зв'язку різних пристроїв, включаючи комп'ютерну, побутову та оргтехніку, засоби зв'язку тощо. Фізичний та канальний рівні регламентуються стандартом IEEE 802.15.4. Радіус дії WPAN становить від кількох метрів до кількох десятків сантиметрів. Такі мережі використовуються як для об'єднання окремих пристроїв між собою, так і для зв'язку з мережами вищого рівня, наприклад, глобальною мережею інтернет. WPAN може бути розгорнута з використанням різних мережевих технологій, наприклад, Bluetooth, ZigBee, 6LoWPAN та інших.

2. Бездротові сенсорні мережі WSN (Wireless Sensor Network). Розподілені мережі, що самоорганізуються, множини датчиків (сенсорів) виконавчих пристроїв, об'єднаних між собою за допомогою радіоканалу. Область покриття подібних мереж може становити від кількох метрів до кількох кілометрів за рахунок можливості ретрансляції повідомлень від одного елемента до іншого.

3. Невеликі локальні мережі TAN (Tiny Area Network). Обчислювальні мережі, що розгортаються в межах невеликого офісу чи окремого житла. Їх часто називають домашніми мережами, оскільки вони поєднують комп'ютери, побутову електроніку та прилади сигналізації, що належать одній родині. Найчастіше такі мережі будуються з урахуванням технології Wi-Fi.

3.1. Бездротова персональна мережа WPAN, що не використовує IP

Спектр можливих технологій передачі даних охоплює всі можливі засоби бездротових і дротових мереж.

Існує безліч різних каналів зв'язку між кінцевою точкою та Інтернетом; одні з них можуть бути побудовані на традиційному стеку IP (6LoWPAN), а деякі, наприклад для максимізації економії енергії,

використовують інші (не-IP) підходи. Системи зв'язку, відмінні від IP, оптимізовані для економії витрат і енергії, тоді як рішення на базі IP зазвичай мають менше обмежень і більш детально вивчаються на інших дисциплінах [20].

3.1.1. Стандарт IEEE 802.15.4

Для бездротової передачі даних особливо важливу роль в побудові «Інтернету Речей» грають такі властивості, як ефективність в умовах низьких швидкостей, відмовостійкість, адаптивність, можливість самоорганізації. Основний інтерес в цій якості представляє стандарт IEEE 802.15.4, що визначає фізичний рівень і управління доступом для організації енергоефективних персональних мереж [21].

IEEE 802.15.4 - стандарт, який визначає фізичний рівень та управління доступом до середовища для бездротових персональних мереж із низьким рівнем швидкості. Апаратура, побудована на базі даного стандарту відноситься до пристроїв малого радіуса дії. Вона являється базовою основою для протоколів ZigBee, WirelessHART, MiWi, ISA100.11, кожен з яких, у свою чергу, пропонує рішення для побудови мереж за допомогою побудови верхніх рівнів, які не регулюються стандартом. Акцент стандарту робиться на:

дуже низьку вартість зв'язку з найближчими пристроями, з невеликою базовою структурою;

простоту реалізації;

експлуатацію з дуже низьким (досі небувалим низьким) рівнем споживання енергії.

Основна межа прийому визначається еквівалентною ізотропно-випромінюючою потужністю (ЕІВП) радіопристрою зі швидкістю передачі 250 кбіт/с. Дозволяється використання радіопристроїв без отримання окремих дозволів від організацій нагляду за частотами (в Україні – «Укрчастонагляд») на використання радіочастот, а також на безліцензійне ввезення пристроїв з максимальною ЕІВП – 100 мВт. Спочатку стандартом були визначені низькі швидкості передачі в 20 і 40 кбіт/с, швидкість в 100 кбіт/с була додана в поточному перевиданні. Більш низькі швидкості передачі можуть розглядатися тільки з суттєвим результуючим ефектом зниження енергоспоживання.

В ряді найважливіших функцій знаходяться: забезпечення роботи в режимі реального часу за допомогою збереження часових слотів, запобігання одночасному доступу та комплексне забезпечення захисту мереж. Пристрої також включають функції управління витрачанням енергії, такі як якість з'єднань і детектування енергії.

Архітектура протоколу

Пристрої розроблені для того, щоб взаємодіяти один з одним за допомогою концептуально простої бездротової мережі. Визначення рівнів мережі ґрунтується на мережевій моделі OSI, хоча тільки нижні рівні визначаються в стандарті. Взаємодія з верхніми рівнями передбачається, з можливим використанням підрівневого керування логічними зв'язками. Реалізовані пристрої можуть покладатися на зовнішні пристрої або бути просто вбудованими, як самостійно функціонуючі пристрої.

Фізичний рівень, в кінцевому рахунку, надає послуги передачі даних, також як і інтерфейс організації управління фізичним рівнем і забезпечує базу даних відповідної персональної мережі. Таким чином, фізичний рівень управляє трансиверною радіостанцією і виконує вибір каналів і енергії та сигнальні функції керування. Він діє в одному з трьох можливих неліцензуємих радіочастотних смуг:

• 868,0 - 868,6 МГц: Європа, дозволено один канал зв'язку;

• 902 - 928 МГц: Північна Америка розширений до тридцяти каналів;

• 2400 - 2483,5 МГц: використовується у всьому світі з більш ніж шістнадцяти каналів.

Початкова версія стандарту 2003 р. визначає два фізичних рівні, що базуються на широкосмуговій модуляції з прямим розширенням спектра, одна працює на ділянці 868/915 МГц зі швидкістю передачі в 20 і 40 кбіт/с, а інша на ділянці 2450 Мгц зі швидкістю 250 кбіт/с.

Версія 2006 підвищує максимальні швидкості передачі даних на частотах 868/915 Мгц, також придавши їм швидкості в 100 і 250 кбіт/с. Крім того, вона йде далі, визначаючи чотири фізичних рівня в залежності від методу модуляції. Три з них зберігають підхід широкосмугової модуляції, в діапазоні 868/915 МГц використовують як двосекційну, так і квадратурну фазову маніпуляцію (остання визнається більш оптимально) в діапазоні 2450 МГц, з допомогою останнього. Як альтернатива, оптимальний рівень на частоті 868/915 МГц визначається за допомогою комбінації двоїчного кодування та амплітудної маніпуляції (таким чином, на основі паралельного, а не послідовного розширення спектра). Можливо динамічне переключення між підтримуваними ланками 868/915 МГц.

У вересні 2009 стандарти IEEE 802.15.4c та IEEE 802.15.4d розширили доступні фізичні рівні, додавши:

рівень для частоти 780 МГц, що використовує квадратурну фазову маніпуляцію (Quadrature phase-shift keying, QPSK) або фазову маніпуляцію більш високих порядків (M- PSK);

рівень для частот 950 МГц, що використовує гаусівську частотну маніпуляцію (Gaussian frequency-shift keying, GFSK) або двоїчну фазову маніпуляцію (Binary phase-shift keying, BPSK).

Рівень механізму доступу (Media Access Control, MAC) здійснює передачу фрагментів даних структури MAC за допомогою використання фізичного каналу. Крім інформаційних послуг, він пропонує управління інтерфейсом і сам по собі управляє розміщенням маячків на каналах. Він також контролює перевірку фрагментів структури, гарантує множинний доступ із розподілом за часом і керує зв'язками вузлів. Також він пропонує точки-пастки для безпеки послуг.

Стандарт не визначає *інші більш високі рівні* та сумісність проміжних рівнів. Існують специфікації, такі як ZigBee, побудовані на даному стандарті для того, щоб запропонувати інтегровані рішення. Стеки TinyOS також використовують деякі види технічного забезпечення IEEE 802.15.4.

Модель мережі

Стандарт визначає два типи вузлів мережі. Перший - повнофункціональний пристрій (FFD - Full-Function Device). Він може служити як координатор персональних мереж, так само може функціонувати як загальний вузол. Він реалізує загальну модель зв'язку, яка дозволяє вести переговори з іншими пристроями, також може передавати повідомлення, в цьому випадку він називається координатором (координатор PAN, коли він відповідає за всю мережу).

Другий - пристрої з полегшеними функціями (RFD - Reduced-Function Device). Це надзвичайно прості пристрої з дуже скромним ресурсом і вимогами до мережі, у зв'язку з цим вони можуть тільки зв'язуватися з повнофункціональними пристроями і ніколи не можуть діяти як координатори.

Мережі можуть бути одноранговими (P2P, peer-to-peer, point-to-point), або мати топологію «зірка». Однак, люба мережа повинна мати по крайній мірі один FFD, який буде працювати як координатор мережі. Таким чином, мережі формуються з груп пристроїв, розділених відповідною дистанцією. Кожен пристрій має 64-бітний ідентифікатор, в деяких випадках може використовуватися 16-бітний ідентифікатор в обмеженій області. Тобто, всередині кожної персональної мережі (англ. PAN, персональна мережа), для з'єднання будуть використовуватися короткі ідентифікатори.

Однорангові мережі (P2P) можуть утворювати будь-які структури об'єднань, а їх розширення обмежені лише дистанцією між кожною парою вузлів. Вони призначені бути основою для бездротових мереж, здатних до самоорганізації та організації. Так як стандарт не визначає

мережевий рівень, маршрутизація не підтримується напряму, але такий додатковий рівень може здійснювати підтримку мереж з ретрансляторами.

Так само можуть бути додані додаткові топологічні обмеження: наприклад, дерево кластерів – структура, в якій RFD може бути пов'язана тільки з одним FFD одноразово, таким чином, RFD є виключно листям дерева, а більшість вузлів – FFD. Також можлива ситуація з мережевою топологією, коли вузли є мережами кластерних дерев з локальним координатором для кожного кластера, крім глобального координатора.

Більш структурована топологія «зірка», де координатор мережі обов'язково повинен бути центральним вузлом. Така мережа може виникнути, коли FFD вирішує створити власну персональну мережу (PAN) і оголосити її координатором, після чого вибирається унікальний ідентифікатор для PAN. Після цього інші пристрої можуть приєднатися до мережі, яка повністю незалежна від інших мереж з топологією «зірка».

Архітектура передачі даних. Фрагменти даних – основа для передачі даних, яка здійснюється за чотирма основними типами: (дані, підтвердження, маячок та фрагменти команд механізму доступу), що забезпечує раціональний баланс між простотою та надійністю. Додатково може використовуватися суперфрагментна структура, що визначається координатором, у цьому випадку два маяки діють як її межі та забезпечують синхронізацію інших пристроїв, також як і інформацію про конфігурацію. Суперфрагмент складається з шестидесяти слотів однакової довжини, які можуть бути в подальшому розділені на активні та неактивні частини, під час виконання яких координатор може входити в режим енергозбереження, в якому немає необхідності контролю мережі.

Затвердження меж суперфрагментів здійснюється системою Carrier Sense Multiple Access With Collision Avoidance (CSMA/CA). CSMA/CA, «множинний доступ з контролем несучої та уникненням колізій» – це мережевий протокол, у якому: використовується схема прослуховування несучої хвилі; станція, яка має намір почати передачу, посилає jam signal (сигнал затору); після тривалого очікування всіх станцій, які можуть надіслати jam signal, станція починає передачу кадру; якщо під час передачі станція виявляє jam signal від іншої станції, вона зупиняє передачу на час випадкової довжини і потім повторює спробу.

Кожна передача повинна закінчуватися перед появою наступного маячка. Додатки, що потребують чітко визначеної ширини діапазону, можуть використовувати сім областей з однієї або декількох

безсистемних гарантованих областей множинного доступу, що йдуть в кінці суперфрагмента. Зазвичай суперфрагменти використовуються при роботі пристроїв із низьким прихованим енергостаном, чиї зв'язки повинні зберігатися, навіть протягом тривалого періоду неактивності.

Передача даних координатору вимагає фаз маячкової синхронізації, за допомогою передачі режиму CSMA/CA, якщо це можливо (за допомогою множинного доступу, якщо використовуються суперфрагменти), сигнал підтвердження не обов’язковий. Передача даних від координатора звичайно супроводжує запити до пристроїв, якщо маячки використовуються, то використовуються сигнали-запити, координатор підтверджує запит і потім надсилає інформаційні пакети, які підтверджуються пристроєм. Те ж саме відбувається, якщо суперфрагменти не використовуються, тільки в цьому випадку немає маячків, щоб зберегти шляхи передачі інформації. Однорангові мережі також можуть використовувати режим CSMA/CA або механізми синхронізації, в цьому випадку зв'язок між двома пристроями можливий, тоді як в "структурованих" режимах один з пристроїв має бути координатором мережі. У загальному всі наступні процедури супроводжуються звичайним за просом/підтвердженням відповіді.

Надійність і безпека. Фізичний носій можна отримати через протокол CSMA/CA. Мережі, що не використовують маячкові механізми, використовують варіант, заснований на прослуховуванні носіїв, що піддані впливу алгоритму зниження швидкості передачі. Підтвердження не підкоряються цьому порядку. Загальна передача даних використовує вільні слоти, де використовуються маячки, процес не супроводжується підтвердженнями.

Повідомлення про підтвердження можуть носити необов'язковий характер при деяких обставинах, якщо робиться припущення про успіх. В будь-якому випадку, якщо пристрій не може обробляти фрагмент у даний момент, він просто не підтверджує його отримання: ретрансляція, заснована на перериві, може виконуватися кілька разів, супроводжуючи після цього рішення – або припинити, або продовжити спроби.

Так як передбачене обладнання для цих пристроїв вимагає максимального збільшення терміну служби батарей, для протоколів вибираються методи, що допомагають цьому: здійснюють періодичні перевірки для очікуваних повідомлень, частота яких залежить від застосування. Що стосується захисту зв'язків, підрівень MAC пропонує можливості, які можуть бути використані в верхніх рівнях для досягнення бажаного рівня безпеки. Процеси у вищих рівнях можуть визначати ключі для виконання симетричної криптографії для захисту навантаження та обмеження її для групових пристроїв або просто для

двосторонніх зв'язків, ці групи пристроїв можуть бути описані в списках контролю доступу.

Крім того, MAC обчислює час перевірки між послідовностями прийомів для запобігання можливості виходу старих кадрів, або дані (які більше не вважаються дійсними) не виходять на більш високі рівні. В додаток до цього захищеного режиму захисту є інший незахищений режим MAC, який дозволяє списки контролю доступу лише як засоби для рішення про прийняття фрагментів відповідно до їх передбачуваним джерелом.

ZigBee та IEEE 802.15.4

Стандарт IEEE 802.15.4 є одним найновішим в серії бездротових. На його основі ZigBee Alliance (www.zigbee.org) розробив специфікацію протоколів мережевого і прикладного рівня, які анонсував під назвою "ZigBee" [12]. ZigBee Alliance включає в себе більше 180 фірм, що працюють спільно над просуванням стандартів, стека протоколів і прикладних профілів для споживчого та промислового сектора економіки. Специфікація ZigBee описує побудову мережі, питання безпеки, прикладне програмне забезпечення.

Основною областю застосування ZigBee є передача інформації від рухомих і обертових частин механізмів (конвеєрів, роботів), промислових систем управління та моніторингу, бездротових мереж датчиків, відстеження маршрутів руху та місця розташування майна та інвентарю, "інтелектуальне" сільське господарство, системи охорони.

На відміну від інших бездротових технологій, де ставиться завдання забезпечити високу швидкість передачі, велику дальність або високу якість обслуговування, ZigBee/IEEE802.15.4 створювався спочатку за критеріями малої дальності дії, низьку ціну, низьку споживану потужність, низькій швидкості передачі і малих габаритів. Ці властивості ідеально відповідають вимогам до більшості промислових датчиків. Тому ZigBee часто ототожнюють з промисловими бездротовими сенсорними мережами WSN (Wireless Sensor Network). Пристрої ZigBee використовуються в застосуваннях, де Bluetooth виявляється занадто дорогим, і не потрібна висока швидкість передачі.

ZigBee використовує неліцензований діапазон 2,4 ГГц. Стандарт передбачає також використання частот 868 МГц в Європі і 915 МГц в США. Максимальна швидкість передачі становить 250 кбіт/с в діапазоні 2,4 ГГц. Смуга пропускання 2,4 ГГц розділена на 11 ... 26 каналів шириною по 5 МГц кожен.

Незважаючи на те, що вся ідеологія стандарту IEEE 802.15.4 побудована в припущенні, що типовий зв'язок буде здійснюватися на відстані близько 10 м, стандарт не встановлює вимог до потужності

передавача. Цей параметр регулюється нормативними документами в галузі радіозв'язку, специфічними для кожної держави. Найбільшого поширення на ринку мають передавачі з потужністю 1 мВт, які забезпечують зв'язок на відстані до 10 м в приміщенні, а також передавачі з потужністю 10 мВт, що збільшують цю відстань до 80 м в приміщенні і до 1 км в умовах прямої видимості. Дальність зв'язку можна збільшити застосуванням антен спеціальної конструкції.

Модель OSI мережі ZigBee представлена в табл. 3.1 Вона включає в себе фізичний рівень (PHY), канальний рівень, що складається з підрівня доступу до середовища передачі MAC і LLC, які визначаються стандартом IEEE 802.15.4, а також мережевий рівень NWK (NetWorK) і

Таблиця 3.1.

Рівні моделі OSI мережі ZigBee/IEEE 802.15.4

OSI модель	**Мережа**	**Функції**
Прикладний	APL (APS, ZDO и Application Objects) ZigBee	Передача повідомлень, виявлення пристроїв, визначення ролі пристроїв
Рівень представлення	-	-
Сеансовий	-	-
Транспортний	-	-
Мережевий	NWK ZigBee	Безпека, маршрутизація
Канальний (передачі даних)	LLC IEEE 802.15.4	CSMA/CA, передача маячків, синхронізація
	SSCS IEEE 802.15.4	
	MAC IEEE 802.15.4	
Фізичний	PHY IEEE 802.15.4	Радіоканал 2,4 ГГц

рівень додатків APL, що складається з підрівня підтримки додатків (APplication Support sub-layer - APS), підрівні об'єктів пристроїв ZigBee (ZigBee Device Object - ZDO) і об'єктів Application Objects, що визначаються виробником ZigBee-пристроїв.

Фізичний рівень моделі OSI забезпечує інтерфейс між стеком протоколів і середовищем передачі інформації (ефіром). Фізичний (PHY) і канальний (MAC) рівні моделі OSI визначені в стандарті IEEE 802.15.4. Вони мають такі основні характеристики:

- швидкість передачі: 250 кбіт / с;
- коротку 16-бітову адресу або розширену довжиною 64-біта;
- виділення інтервалу часу для передачі інформації кожним вузлом;
- метод доступу до каналу типу CSMA / CA;
- протокол обміну з повідомленням про отримання;

- мале споживання потужності;

- контроль рівня енергії;

- наявність індикатора якості зв'язку;

- 16 каналів в діапазоні 2,45 ГГц.

Стандарт IEEE 802.15.4 використовує модуляцію типу OQPSK – "Offset-Quadrature Phase-Shift Keying" – "зміщена квадратурная фазова маніпуляція".

Основним призначенням фізичного рівня є прийом і передача даних через радіоканал. Тут також вимірюється потужність радіосигналу, оцінюється якість зв'язку і чистота каналу, здійснюється вибір каналу.

Підрівень MAC управляє доступом до радіоканалу, використовуючи метод CSMA/CA. Він також відповідає за передачу маячкових фреймів, синхронізацію і забезпечення надійних методів передачі інформації. Підрівень SSCS (Service Specific Convergence Sublayer - "підрівень зближення специфічних сервісів") виконує роль інтерфейсу між підрівнями LLC і MAC. Підрівень LLC виконує зв'язок мережевого рівня з рівнем MAC.

Рівень NWK використовує методи, що забезпечують:

- реєстрацію в мережі нового пристрою і виключення його з мережі;

- забезпечення безпеки при передачі фреймів;

- вказівку маршруту фрейму до місця призначення;

- прокладку маршрутів між пристроями в мережі;

- виявлення в мережі найближчих сусідів;

- запам'ятовування необхідної інформації про сусідніх вузлах.

У ZigBee є три типи пристроїв:

- координатор – формує топологію мережі і може встановлювати мости з іншими мережами. У кожній ZigBee мережі є тільки один координатор;

- маршрутизатор – працює як проміжна ланка, передаючи в потрібному напрямку дані від інших пристроїв;

- кінцевий пристрій – передає дані координатору або маршрутизатору і не може зв'язуватися з аналогічними йому пристроями.

Рівень NWK координатора відповідає за організацію нової мережі, коли це потрібно і призначення адрес нових пристроїв, що підключаються до мережі.

Підрівень APS рівня додатків забезпечує:

- обслуговування таблиць для зв'язування пристроїв мережі на основі інформації про необхідність і можливість зв'язування;

- передачу повідомлень між пов'язаними пристроями;

- визначення групової адреси пристроїв, видалення і фільтрацію повідомлень з груповими адресами;

- відображення 64-бітної адреси в 16-бітну;

- фрагментацію, перекомпонування і транспортування даних.

Підрівень ZDO забезпечує:

- визначення ролі пристроїв в мережі (координатор, маршрутизатор або термінал);

- ініціювання або відповідь на запит з'єднання;

- захист інформації;

- виявлення пристроїв в мережі і визначення, який сервіс вони надають.

Топологія Zigbee-мережі підтримується рівнем NWK і може мати форму зірки, дерева або комірчастої мережі. У топології типу зірки мережа контролюється координатором. Координатор відповідає за ініціалізацію і обслуговування мережевих пристроїв і всіх кінцевих пристроїв, безпосередньо взаємодіють з координатором. У мережі комірчастої і деревовидної структури координатор відповідає за організацію мережі і вибір деяких ключових параметрів, але мережа може бути розширена за допомогою ZigBee маршрутизаторів. У мережі з деревоподібною топологією маршрутизатори переміщують дані і керуючі повідомлення по мережі, використовуючи ієрархічну стратегію маршрутизації. Деревовидні мережі можуть використовувати маячкову стратегію маршрутизації.

Чарункова (комірчаста) мережа повинна забезпечити повну однорангову комунікацію пристроїв, тобто в комірчастої мережі немає пристроїв різних рангів (координаторів, маршрутизаторів і т.п. - всі пристрої рівноправні) [22].

Стандарт допускає опціональне використання суперфреймовой структури повідомлень (рис. 3.1). Формат суперфрейма визначається мережевим координатором. Суперфрейм з двох сторін обмежується маячками, ділиться на 16 рівних по довжині слотів і надсилається мережевим координатором. Маячок поміщається на місце першого слота кожного суперфрейма. Координатор може відключити режим повідомлень з маячками. Маячки використовуються для синхронізації приєднаних пристроїв, для ідентифікації мережі і для опису структури суперфрейма. Будь-які пристрої, які бажають розпочати процес комунікації в проміжок часу між двома маячками, повинні

використовувати слотовий механізм доступу CSMA/CA. Передача повідомлень повинна бути закінчена до приходу наступного маячка.

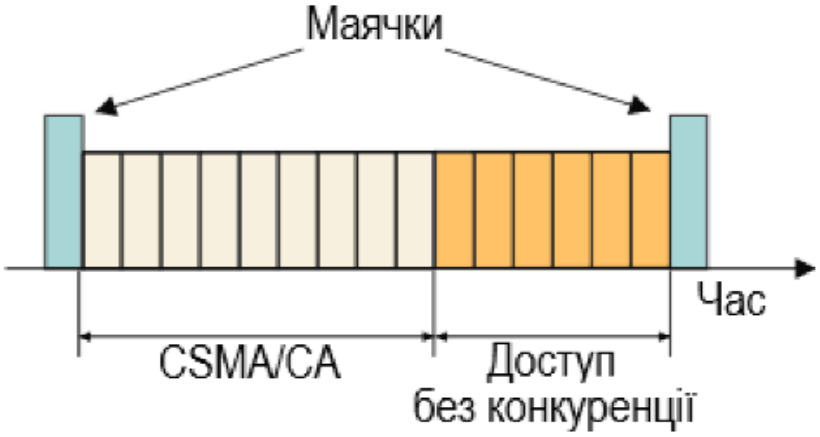


Рис. 3.1. Структура суперфрейма з гарантованими часовими слотами

IEEE 802.15.4 встановлює два механізми доступу до каналу CSMA/CA, в залежності від типу конфігурації мережі. У мережі без маячків використовується звичайний (безслотовий) механізм доступу CSMA/CA. Кожен раз, коли пристрій збирається почати передачу, він повинно витримати паузу випадкової величини після того, як канал звільниться. Випадкова затримка потрібна тому, що дуже ймовірно, що багато пристроїв мережі чекають звільнення каналу і тому після його звільнення можуть почати передачу одночасно. Якщо канал зайнятий, то пристрій може повторити спробу після повторної випадкової затримки. Фрейми підтвердження про отримання надсилаються відразу, без використання описаного алгоритму.

У мережі з маячками використовується слотовий (тактованний) механізм доступу CSMA/CA, в якому початок часового слота має співпадати з межею суперфрейма мережевого координатора, тобто початок слота для кожного пристрою має бути синхронізований з початком передачі маячка мережевим координатором. Оскільки пристрій не може почати передачу, поки не знайде маячок, а маячки розсилаються тільки мережевим координатором, то мережевий координатор з допомогою маячків виконує тактування актів обміну в усій мережі. При цьому PHY рівень повинен забезпечити, щоб всі передачі в мережі починалися одночасно з початком слотів. Введення описаної синхронізації дозволяє зменшити ймовірність одночасної передачі повідомлень декількома вузлами мережі.

Для пристроїв, які вимагають термінової доставки або великої пропускної здатності каналу, мережевий координатор може зарезервувати частину суперфрейма, в якому буде відсутня конкуренція за канал (рис. 3.2), оскільки в цей час мережевий координатор забороняє

будь-яку передачу всім іншим пристроям. Ця частина слотів суперфрейма називається гарантованими часовими слотами (Guaranteed Time Slots – GTSs).

Рис. 3.2. Процес передачі даних від координатора до пристрою

Модель передачі даних.

В IEEE 802.15.4 існує три типи обмінних процесів:

- передача від пристрою до мережевого координатора;
- передача від мережевого координатора до пристрою;
- передача між двома одноранговими пристроями.

У зірковій топології використовуються тільки два перші варіанти, оскільки в ній не існує обмінів між одноранговими пристроями.

Коли пристрій збирається передати дані координатору в мережі з маячками, він спочатку намагається виявити маячок. Коли маячок знайдений, пристрій підлаштовується до структури суперфрейма. Пристрій передає дані координатору, використовуючи слотовий механізм CSMA/CA. У відповідь координатор відсилає фрейм повідомлення про отримання. На цьому цикл обміну закінчується. Якщо пристрій збирається передати дані в мережі без маячків, він передає дані, використовуючи безслотовий метод CSMA/CA.

Коли координатор бажає передати дані пристрою в мережі з маячками, він поміщає в маячок інформацію про те, що є дані, готові до передачі (рис. 3.2). Пристрій періодично аналізує вміст маячка і, якщо в ньому є інформація про наявність повідомлення, готового до передачі, пристрій передає команду запиту даних, використовуючи CSMA/CA. Координатор підтверджує прийом запиту даних за допомогою фрейму повідомлення. Слідом за цим координатор відсилає дані, використовуючи CSMA/CA. Пристрій підтверджує прийом даних відправкою повідомлення.

Якщо координатор збирається передати дані без використання маячка, він запам'ятовує дані і чекає запиту від пристрою. Пристрій може передати команду запиту даних координатору, використовуючи безслотовий метод CSMA/CA. Координатор спочатку посилає повідомлення про отримання (в тому ж циклі обміну), потім, використовуючи CSMA/CA, відсилає дані та отримує повідомлення про отримання від пристрою.

Структура фреймів була спроектована за критерієм мінімальної складності, що забезпечує надійну передачу даних в зашумленому каналі. Відповідно до моделі OSI, кожен нижчий рівень додає до протоколу свій заголовок. Стандарт передбачає чотири типи фреймів: фрейм маячка; фрейм даних; фрейм повідомлення про отримання; фрейм команд MAC-підрівня.

Фрейм даних (рис. 3.3) починається з преамбули, яка спільно з полем "Старт" служить для синхронізації даних в приймачі, поле "Довжина" містить довжину поля MAC-підрівня в 8-бітових байтах (октетах). Поле "Управління" містить службову інформацію про управління кадрами, поле "Номер" містить порядковий номер даних, поле "Адреса" містить адресну інформацію, в тому числі 16-бітну коротку або 64-бітну розширену адресу. Завершується фрейм полем контрольної суми КС [23].

Мережевий рівень

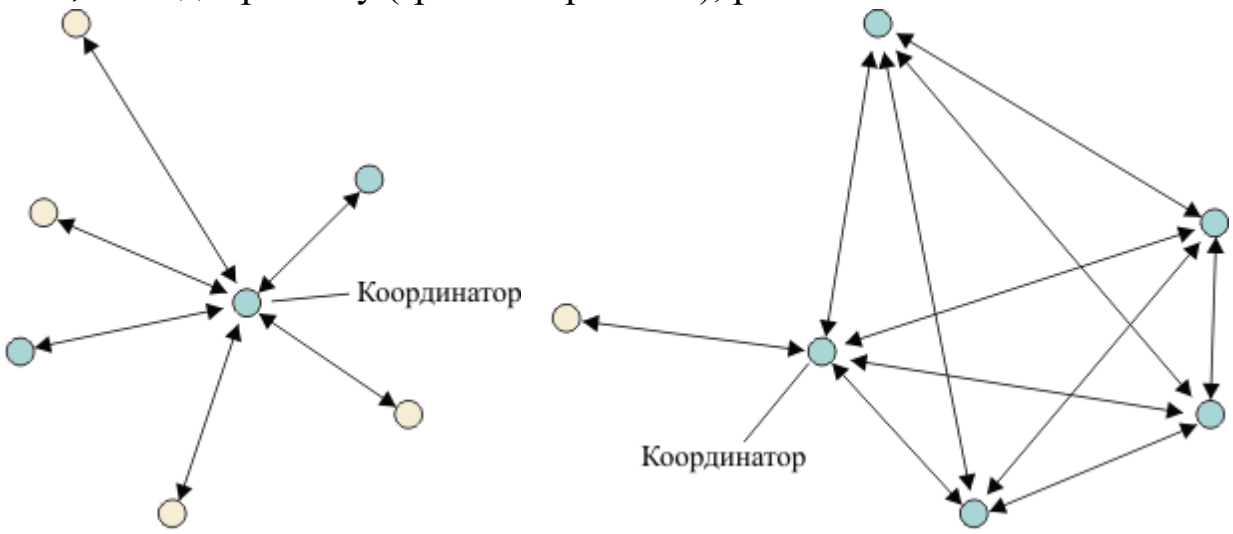
Особливістю мереж ZigBee є можливість виконувати ретрансляцію переданих даних через безліч проміжних вузлів в мережі, причому при виході з ладу або виключенні одного з вузлів мережі пристрій знаходить інший шлях для передачі інформації. При включенні живлення пристрою мережа заново включає його до свого складу.

Рис. 3.3. Формат фрейму даних по стандарту IEEE 802.15.4

Стандарт розрізняє два типи пристроїв: повнофункціональні пристрої (FFD - Full-Function Device) і пристрої зі скороченим набором функцій (RFD - Reduced-Function Devices). FFD можуть працювати в мережі з деревоподібної топологією в якості координатора мережі або в

якості пристрою. FFD можуть обмінюватися інформацією з іншими FFD або RFD, але RFD можуть зв'язуватися тільки з FFD. RFD набагато простіше і дешевше, ніж FFD. Будь-яка мережа повинна містити принаймні один повнофункціональний пристрій FFD.

Залежно від вимог конкретного застосування, мережа на основі стандарту IEEE 802.15.4 може мати одну з двох топологій: зіркову рис. 3.4а, або однорангову ("рівний з рівним"), рис. 3.4б.



а) б)

Рис. 3.4. Зіркова (а) і однорангова (б) топології мережі. Синє коло - повнофункціональний пристрій (FFD), жовте - з скороченою функціональністю (RFD)

Всі пристрої в мережі незалежно від топології повинні мати унікальну 64-бітну розширену адресу. Ця адреса використовується для комунікації в межах мережі або може бути змінена на коротку 16-бітову адресу, що виділяється координатором в процесі підключення пристроїв до мережі. Координатор може бути підключений до мережі живлення, а інші мережеві пристрої можуть мати батарейне живлення.

Однорангова мережа також має координатора, проте вона відрізняється тим, що будь-який пристрій може обмінюватися даними з будь-яким іншим, якщо він знаходиться в зоні досяжності радіозв'язку, в той час як в зірковій топології будь-який пристрій може взаємодіяти тільки з координатором. Відзначимо, що однорангова мережа виходить завжди дорожче, оскільки вона містить тільки FFD, але завдяки цьому вона дозволяє організовувати мережі більш складної топології, в тому числі пористі. Більшість промислових застосувань вимагають застосування однорангових мереж. До них відносяться управління і моніторинг, сенсорні мережі, відстеження місця розташування майна та товару, "інтелектуальне" сільське господарство, системи безпеки. Однорангові мережі можуть бути спеціалізованими, здатними до

самоорганізації та самовідновлення. Вони дозволяють передавати інформацію між вузлами мережі незалежно від відстані між ними, використовуючи проміжні вузли в якості ретрансляторів. Ці функції виконуються рівнем додатків моделі OSI.

Кілька мереж можуть взаємодіяти один з одною. Для цього кожна мережа повинна мати унікальний мережевий ідентифікатор. Завдяки йому всередині мережі можуть використовуватися скорочені адреси. Тому повна адреса пристрою для доступу ззовні (з іншої мережі) складається з адреси мережі і короткого адреси пристрою.

Базова структура мережі зіркової топології показана на рис. 3.4а. Після першого включення повнофункціонального мережевого пристрою він може організувати свою власну мережу і стати мережевим координатором. Всі мережі зіркової топології функціонують незалежно одна від одної. Це досягається вибором мережевого ідентифікатора, який не використовується іншими мережами, що знаходяться в межах радіусу дії даної мережі. Після вибору мережевого ідентифікатора координатор може прийняти інші пристрої до складу мережі, як FFD, так і RFD.

У одноранговій мережі (рис. 3.4 б) кожен пристрій може взаємодіяти з будь-яким іншим пристроєм, що знаходиться в межах його радіуса дії. Один з пристроїв призначається координатором, наприклад, той, що перший включений в мережу. При подальшому розширенні мережі можна відійти від одноранговой топології і створити гібридну топологію, в якій будуть міститися і пристрої з скороченою функціональністю.

Прикладом застосування одноранговой комунікації між пристроями може бути кластерне дерево (рис. 3.5). Кластерне дерево є спеціальним випадком одноранговой мережі, в якій більшість пристроїв є повнофункціональними. Пристрої з скороченою функціональністю можуть бути підключені до кластерного дерева тільки як кінцеві вузли на кінцях гілок, оскільки вони можуть бути підключені тільки до одного повнофункціонального пристрою. Один (будь-який) з повнофункціональних пристроїв в мережі має відігравати роль мережевого координатора і забезпечувати синхронізацію з іншими пристроями. Мережевий координатор повинен мати підвищені обчислювальні ресурси [21].

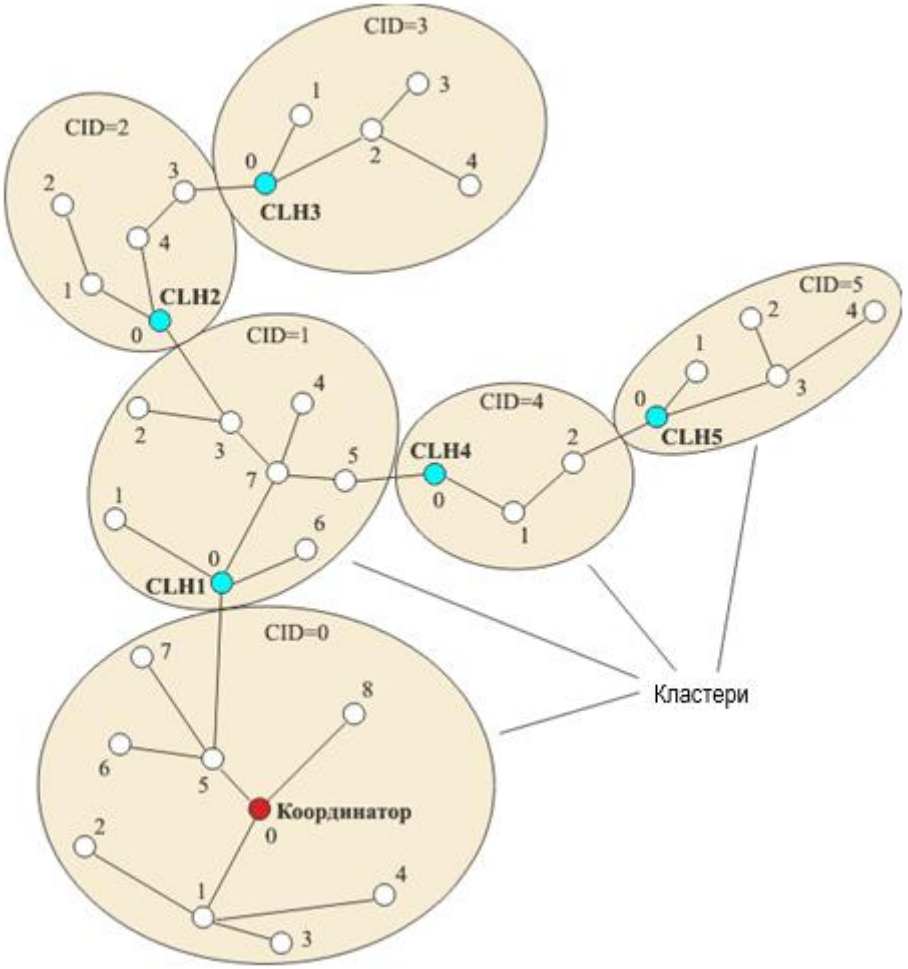


Рис. 3.5. Приклад мережі з топологією кластерного дерева. Гілки указують відношення підлеглості, а не канали зв’язку.

При формуванні мережі типу кластерного дерева мережевий координатор призначає собі головою першого кластера (CLaster Head - CLH) і присвоює своєму кластеру ідентифікатор 0 (Cluster IDentifierCID = 0) та вибирає ідентифікатор всього для кластерного дерева, що формується. Після цього координатор посилає всім сусіднім пристроям команду з маячковим фреймом. Пристрої, які отримали маячок, можуть запросити дозволу приєднатися до кластера, що формується. Якщо мережевий координатор дозволяє приєднання, він додасть новий пристрій до свого списку сусідніх пристроїв. Потім пристрій, що тільки приєднався, додасть CLH як батьківський пристрій до списку своїх сусідів і почне періодично посилати маячок. Тепер інші пристрої можуть приєднатися до нього. Якщо пристрій, який бажає приєднатися до мережі, не може знайти CLH, він може під’єднатися до будь-якого іншого пристрою, який може бути батьківським.

Найпростішим окремим випадком кластерного дерева є один кластер, проте кілька кластерів можуть об'єднуватися, утворюючи кластерне дерево (рис. 3.5). Для цього мережевий координатор

призначає один з повнофункціональних пристроїв главою сусідньої кластера і призначає йому номер кластера CID = 1. Детальніше процедура формування кластерного дерева описана в [7].

Рівень додатків

Рівень додатків пов'язує стек протоколів з кінцевим додатком користувача, наприклад, OPC сервером, який далі використовується для обміну даними з SCADA. Підрівень підтримки додатків APS забезпечує інтерфейс між мережним рівнем і рівнем додатків APL за допомогою загального набору сервісів, які використовуються як підрівні об'єктів пристроїв ZDO, так і прикладними об’єктами (Application Objects), обумовленими користувачем. Підрівень APS розподіляє між кінцевими мережевими пристроями інформацію, що поставляється додатком, наприклад, команди вмикання/вимикання лампочки в системі автоматизації будівлі.

Об'єкти додатків в ZigBee виконують такі функції, використовуючи загальнодоступний інтерфейс ZDO:

- контроль і координація різних рівнів протоколу для ZigBee пристроїв;

- ініціювання стандартних мережевих функцій.

Одним з компонентів ZigBee мережі є ZigBee пристрій. Прикладом може бути вимикач світла, термостат або віддалена система автоматичного управління, які мають доступ до радіоканалу. В одному і тому ж пристрої з одним радіоканалом можуть бути втілені логічно різні функції, наприклад, функція вимірювання ваги і функція вимірювання температури.

Кілька взаємодіючих пристроїв можуть утворювати автоматизовану систему управління, наприклад, АСУ "Розумний будинок" (рис. 3.6). У такій системі підрівень APS моделі OSI забезпечує розподіл інформації, що поставляється для користувача додатком, між пристроями. Такою інформацією можуть бути, наприклад, команди "Включити світло", що посилаються від додатку різним пристроям по радіоканалу.

Рівень підтримки додатків APS для реалізації своїх функцій використовує комунікаційні структури: профілі, кластери і кінцеві точки. Профіль описує колекцію (набір) пристроїв, що використовуються для деякого додатка, і, неявно, схему повідомлень між цими пристроями. Наприклад, в ZigBee є профілі для системи домашньої автоматизації та профілі для комерційних, промислових і офісних систем. Всі профілі використовують стандартні типи повідомлень, формати повідомлень і процедури їх обробки.

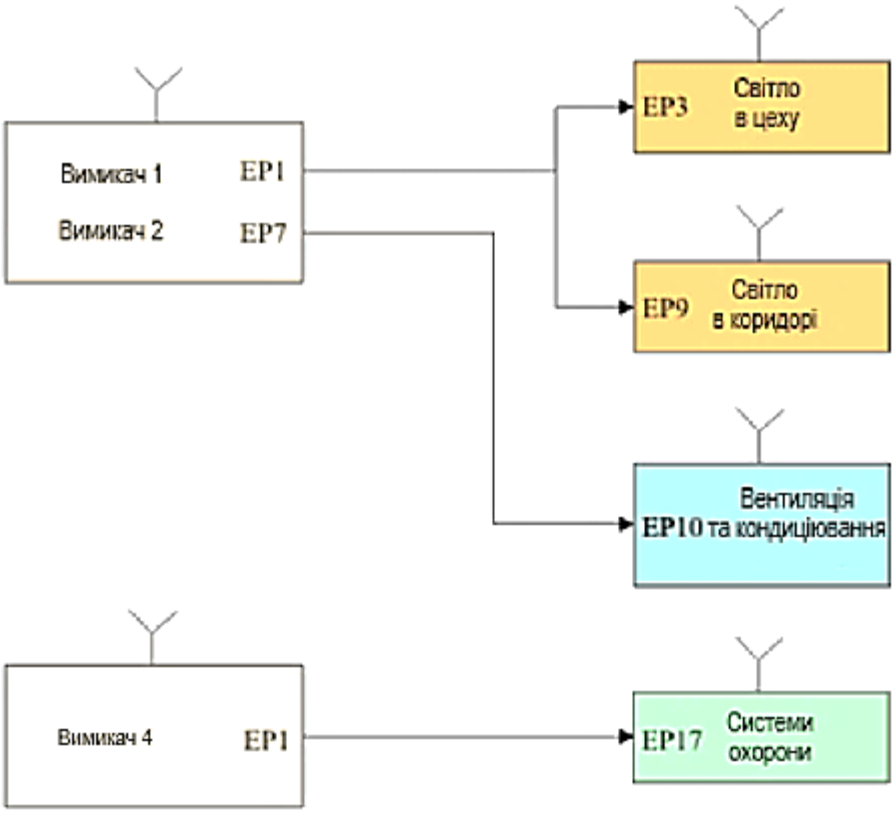


Рис. 3.6. Приклад зв'язування кінцевих точок в ZigBee мережу

В рамках профілів пристрою обмінюються між собою за допомогою кластерів, які можуть виходити або входити в пристрій. Кластер - це повідомлення або колекція повідомлень, до складу яких можуть входити команди і відповіді на них. Наприклад, в профілі для домашньої автоматизації є спеціалізований кластер для управління освітленням. До його складу можуть входити команди «Ввімкнути/Вимкнути». До складу кластера може входити набір команд для конфігурації пристрою. Кожен кластер має свій ідентифікатор і є унікальним тільки в межах певного профілю.

Кінцева точка вказує об'єкт в межах пристрою, з яким взаємодіє додаток. Наприклад, кінцева точка EP1 (EP – від слів "End Point") може призначатися для управління світлом в цеху і коридорі (рис. 3.6), кінцева точка EP7 – для управління системою вентиляції і кондиціонування, кінцева точка 1 другого пристрою – для управління системою охорони будівлі. Кінцеві точки виконують функцію адресації і дозволяють визначити, якому пристрою призначене надіслане повідомлення. В межах одного пристрою вони мають індекси від 1 до 240. Без кінцевих точок керувати кількома об'єктами в межах одного пристрою було б неможливо, оскільки адресу має тільки пристрій, а кінцеві точки – це суб-адреси з номерами від 1 до 240.

Зв'язки між кінцевими точками зберігаються у вигляді таблиці зв'язків, яка запам'ятовується в пристрої, від якого виходять команди управління, якщо пристрій має достатню для цього ємність пам’яті. Таблиця зв'язків може також зберігатися у допоміжному пристрої.

Прагнучи забезпечити сумісність (точніше, інтероперабельність) пристроїв різних виробників в системі ZigBee, стандарт пропонує стандартні профілі, які містять стандартні набори кластерів. У разі, коли стандартні профілі не задовольняють потребам системного інтегратора, він може створити свій, призначений для користувача, профіль, включаючи визначення кластерів.

Інтероперабельність (англ. interoperability — здатність до взаємодії) означає можливість створення систем з довільних неоднорідних, розподілених компонентів на базі уніфікованих інтерфейсів або протоколів [7]. Інтероперабельна інформаційна система складається з компонентів, що становлять довільні інформаційні ресурси (програмні компоненти, бази даних, бази знань, файли даних тощо), які розглядають незалежно від апаратно-програмної платформи і розміщення в просторі. Компоненти взаємодіють, обмінюючись заявками.

3.1.2 Bluetooth (IEEE 802.15.1)

Bluetooth (англ. Bluetooth) – технологія бездротового зв'язку, створена у 1998 році групою компаній: Ericsson, IBM, Intel, Nokia, Toshiba. Нині розробки в області Bluetooth ведуться групою англ. Bluetooth SIG (англ. Special Interest Group), до якої входять також Lucent, Microsoft та інші компанії, чия діяльність пов'язана з мережевими технологіями. Основне призначення Bluetooth – забезпечення економного (з точки зору споживаного струму) і дешевого радіозв'язку між різноманітними типами електронних пристроїв, таких як мобільні телефони та аксесуари до них, портативні та настільні комп'ютери, принтери та пристрої IoT. Причому, велике значення приділяється компактності електронних компонентів, що дає можливість застосовувати Bluetooth у малогабаритних пристроях (наприклад, вакуумні навушники). На сьогоднішній день існує вже 5 версій Bluetooth, які суттєво відрізняються між собою. З погляду IoT найбільш перспективними є технології Bluetooth 5.0 і вище. Bluetooth дозволяє пристроям спілкуватись, коли вони знаходяться один від одного в радіусі близько 10 м у старих версіях протоколу і до сотень метрів починаючи з версії Bluetooth 5. Дальність залежить від перешкод і завад, навіть у одному приміщенні.

Інтерфейс Bluetooth дає змогу передавати як голос (зі швидкістю 64 Кбіт/с), так і дані. Для передачі даних можуть бути використані асиметричний (721 Кбіт/с в одному напрямку і 57,6 Кбіт/с в іншому) та симетричний (432,6 Кбіт/с в обох напрямках) методи. Працюючи на частоті 2.4 ГГц, прийомопередавач (Bluetooth-chip) дає змогу встановлювати зв'язок у межах 10 або 100 метрів. Різниця у відстані, безумовно, велика, однак з'єднання в межах 10 метрів дає змогу зберегти низьке енергоспоживання, компактний розмір і досить невисоку вартість компонентів. Так, малопотужний передавач споживає всього 0.3 мА в режимі standby і в середньому 30 мА під час обміну інформацією. У стандарті Bluetooth передбачене шифрування даних, що передаються, з використанням ключа ефективної довжини від 8 до 128 біт і можливістю вибору односторонньої або двосторонньої аутентифікації. Додатково, до шифрування на рівні протоколу, може бути використано шифрування на програмному рівні.

Технологія Bluetooth працює за принципом FHSS (англ. Frequency-hopping spread spectrum). Коротко це можна пояснити так: передавач розбиває дані на пакети і передає їх за псевдовипадковим алгоритмом стрибкоподібної перебудови частоти (1600 разів в секунду), або за шаблоном (pattern), складеним з 79 підчастот. «Зрозуміти» один одного можуть тільки ті пристрої, які налаштовані на один і той самий шаблон передачі – для сторонніх приладів передана інформація буде звичайним шумом [23].

Основним структурним елементом мережі Bluetooth є так звана «пікомережа» (piconet) – сукупність від 2 до 8 пристроїв, що працюють на одному і тому ж шаблоні (рис. 3.7). У кожній пікомережі один пристрій працює як активний (master), а інші як пасивні (slave). Активний пристрій (Master) визначає шаблон, на якому працюватимуть усі пасивні пристрої (slave) його пікомережі, і синхронізує її роботу. Стандарт Bluetooth передбачає з'єднання незалежних і навіть не синхронізованих між собою пікомереж (до 10) в так звану «scatternet» (англ. to scatter звучить як "розсіювати"). Для цього кожна пара пікомереж повинна мати як мінімум один спільний пристрій, який буде активним в одній і пасивним в іншій. Таким чином, у межах окремої scatternet з інтерфейсом Bluetooth може бути одночасно пов'язано максимум 71 пристрій, однак ніхто не обмежує застосування пристроїв-гейтів, які використовують той же Internet для більш далекого зв'язку.

Частотний діапазон Bluetooth в більшості країн вільний від ліцензування, але у деяких країнах через законодавчі обмеження необхідно використовувати відмінні від зазначених вище частоти.

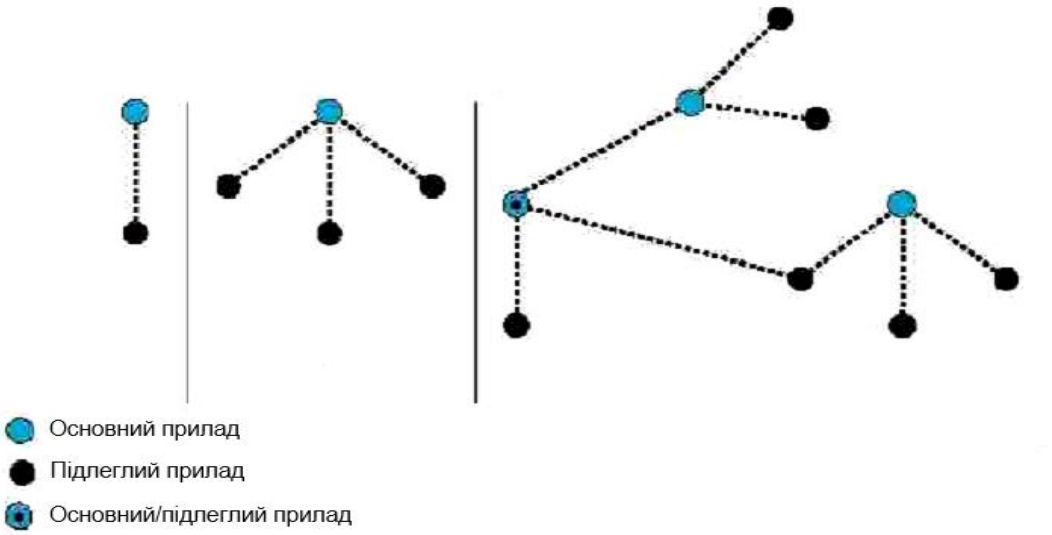


Рис. 3.7. Пікомережі Bluetooth

Стек Bluetooth 5.

Бездротовий зв'язок Bluetooth 5 складається з двох бездротових систем: класичної системи Basic Rate (BR/EDR) та енергоекономної системи Low Energy (LE або BLE). Пристрої Bluetooth постачаються в одно- та дворежимних версіях, що означає, що вони або підтримують лише стек BLE, або одночасно підтримують класичний режим та BLE.

Bluetooth має три основні компоненти: *апаратний контролер*, *програмне забезпечення для хоста* та *профілі програм*.

Поділ між контролером та хостом відбувається на рівні інтерфейсу хост-контролера (HCI). Bluetooth дозволяє підключати один або кілька контролерів до одного хоста. Стек складається з рівнів або протоколів та профілів.

Протоколи – горизонтальні рівні чи шари, які мають функціональні блоки. Bluetooth має багаторівневу архітектуру, що складається з основного протоколу, протоколів заміни кабелю, протоколів керування телефоном та запозичених протоколів. Обов'язковими протоколами для всіх стеків Bluetooth є LMP, L2CAP і SDP. Крім того, пристрої, що зв'язуються з Bluetooth, зазвичай використовують протоколи HCI та RFCOMM.

Link Management Protocol (LMP) – використовується для встановлення та керування радіоз'єднанням між двома пристроями. Реалізується контролером Bluetooth.

Host/controller interface (HCI) – визначає зв'язок між стеком хоста (тобто комп'ютера або мобільного пристрою) та контролером Bluetooth.

Logical Link Control and Adaptation Protocol (L2CAP) – використовується мультиплексування локальних з'єднань між двома

пристроями, використовують різні протоколи вищого рівня. Дозволяє фрагментувати та перезбирати пакети.

Service Discovery Protocol (SDP) – дозволяє виявляти послуги, що надаються іншими пристроями, та визначати їхні установки.

Radio Frequency Communications (RFCOMM) – протокол заміни кабелю, створює віртуальний послідовний потік даних та емулює керуючі сигнали RS-232.

Профілі – представляють вертикальні функції, які використовують протоколи. По суті це набір функцій або можливостей, доступних для певного пристрою Bluetooth. Виділяють загальні профілі атрибутів та загальні профілі доступу. Для спільної роботи Bluetooth-пристроїв необхідно, щоб всі вони підтримували загальний профіль.

Апаратне забезпечення

Основним напрямком використання Bluetooth має стати створення так званих персональних мереж (PAN, або private area networks), які включають такі різнопланові пристрої, як мобільні телефони, PDA, MP3-плеєри, комп'ютери і навіть мікрохвильові печі з холодильниками. Можливість передачі голосу дає змогу вбудовувати інтерфейс Bluetooth в бездротові телефони або, наприклад, бездротові гарнітури для телефонів. Можливості застосування Bluetooth на практиці безмежні: крім синхронізації PDA з настільним комп'ютером або під'єднування до низькошвидкісної периферії на зразок клавіатур або мишей, інтерфейс дає змогу дуже просто і з невеликими витратами організувати домашню мережу. Причому вузлами цієї мережі можуть бути будь-які пристрої, що мають потребу в інформації або володіють необхідною інформацією.

Безпека Bluetooth

Безпека Bluetooth існує як частина протоколу з 1.0 у тій чи іншій формі. Коротко розглянемо безпеку режиму BR/EDR та BLE окремо, оскільки механізми різні.

У режимі BR/EDR, існує кілька режимів автентифікації та з’єднання. Для з’єднання потрібна генерація секретного симетричного ключа. Для режиму BR/EDR це називається ключем з’єднання, тоді як для режиму BLE він називається довгостроковим ключем. Старі пристрої Bluetooth використовували режим з’єднання персонального ідентифікаційного номера (PIN) та ініціювали ключі з’єднання. Нові пристрої (4.1+) використовують безпечне просте з’єднання.

Безпечне просте сполучення (SSP) забезпечує процес з’єднання за допомогою низки різних моделей асоціацій для різних випадків використання. SSP також використовує криптографію з відкритим ключем для захисту від підслуховування та атак типу "людина-посередині" (MITM). Моделі, що підтримуються SSP, включають:

- числове порівняння – для випадків, коли обидва пристрої Bluetooth можуть відображати шестизначне числове значення, яке дозволяє користувачеві вводити відповідь «так»/«ні» на кожному пристрої, якщо числа збігаються;
- введення загального ключа – використовується в ситуаціях, коли один пристрій має цифровий дисплей, а інший має цифрову клавіатуру. У цьому випадку користувач вводить значення, яке відображається на дисплеї першого пристрою на клавіатурі другого пристрою;
- JustWorks – для ситуацій, коли один пристрій не має клавіатури чи дисплея. Він забезпечує мінімальну автентифікацію і не перешкоджатиме атаці MITM;
- позасмуговий (OOB) – використовується, коли пристрій має додаткову форму зв'язку, наприклад, NFC або Wi-Fi. Вторинний канал використовується виявлення і обміну криптографічними значеннями. Він захистить лише від підслуховування та MITM, якщо канал OOB захищений.

Автентифікація в режимі BR/EDR – це виклик-відповідь; наприклад, введення PIN-коду на клавіатурі. Якщо автентифікація не вдалася, пристрій буде очікувати деякий час, перш ніж вирішити нову спробу. Інтервал зростає експоненційно з кожною невдалою спробою. Це просто розчаровує людину, яка намагається вручну підібрати код ключа.

Шифрування в режимі BR/EDR може бути встановлене таким чином, щоб воно було відключено для всього трафіку, тоді трафік даних буде зашифрований, але широкомовний зв'язок не буде, або щоб весь зв'язок був зашифрований. Шифрування використовує криптографію AES-CCM.

Безпека BLE. З’єднання BLE починається з пристрою, що ініціює Pairing_Request та обмінюючись властивостями, вимогами тощо. На початковому етапі процесу з’єднання немає нічого, пов'язаного із профілями безпеки. У цьому відношенні безпека з’єднання аналогічна чотирьом методам BR/EDR (також відомим як моделі асоціації), але в Bluetooth BLE 4.2 трохи відрізняється:

- числове порівняння – це те саме, що і JustWorks, але в кінці обидва пристрої генерують значення підтвердження, яке відображається на екранах хосту та пристрою, щоб користувач міг перевірити відповідність;
- введення загального ключа – подібно режиму BR/EDR, за винятком того, що не ініціалізує пристрій створює випадкове 128-бітове початкове число, яке називається випадковим кодом

для автентифікації з'єднання. Кожен біт ключа доступу автентифікується окремо на кожному пристрої, генеруючи значення підтвердження цього біта. Значення підтвердження обмінюються і мають співпадати. Процес триває до того часу, поки всі біти не будуть оброблені. Це забезпечує досить надійне рішення для захисту від атак MITM;

- JustWorksTM – після того, як пристрої обмінюються відкритими ключами, пристрій, що не ініціалізує, створює випадковий код для створення значення підтвердження Cb. Він передає випадковий код і Cb на пристрій, що ініціює, який, у свою чергу, генерує свій власний випадковий код і передає його першому. Ініціюючий пристрій потім підтвердить справжність неініціалізуємого випадкового коду, створивши власне значення Ca, яке має відповідати Cb. Якщо воно не відповідає, з'єднання пошкоджено. Це знов-таки відрізняється від режиму BR/EDR;
- позасмуговий (OOB) – це те саме, що й у режимі BR/EDR. Він захистить лише від підслуховування та MITM, якщо канал OOB захищений.

У BLE генерація ключів використовує безпечні з'єднання LE. Безпечне з'єднання LE було розроблено для вирішення проблеми безпеки в парі BLE, що дозволило прослуховую ому пристрою побачити обмін з'єднанням. У цьому процесі шифрування з'єднання використовується довгостроковий ключ (LTK). Ключ заснований на криптографії з відкритим ключем алгоритмом Діффі-Хеллмана на еліптичних кривих (ECDH). І ведучий, і ведений генеруватимуть пари ключів публічний – приватний ECDH. Ці два пристрої обмінюватимуться публічними частинами своїх відповідних пар та оброблятимуть ключ Діффі-Хеллмана. На цьому етапі з'єднання може бути зашифроване з використанням криптографії AES-CCM.

Маякування

Маякування Bluetooth – вторинний ефект BLE; однак це важлива технологія для IoT. Маяк просто використовує пристрої Bluetooth у режимі LE для передачі повідомлень на періодичній основі. Маяки ніколи не з'єднуються і не сполучаються з хостом. Якби маяк підключався, всі повідомлення зупинялися, і жоден інший пристрій не міг чути цей маяк. Найчастіше маяки, використовуються для роздрібної торгівлі, охорони здоров'я, відстеження активів, логістики та багатьох інших ринків для: створення статичних точок інтересу (POI); поширення телеметричних даних; позиціонування у приміщенні та геолокації.

3.1.3. Стандарт Z-Wave

Z-Wave – це перший відкритий бездротовий стандарт домашньої автоматизації (системи «розумний» будинок), в основі якої лежить мережа. Він заснований на специфікації ITU G.9959 та визначає всі аспекти взаємодії пристроїв, що підтримують цей протокол, а також забезпечує їхню сумісність. Технологія використовує малопотужні та мініатюрні радіочастотні модулі, які вбудовуються в побутову електроніку та різні системи, такі як освітлення, опалення, контроль доступу, розважальні системи та побутову техніку. Стек протоколу Z-Wave представлений на рис. 5.13 [1].

Рис. 3.8. Стек протоколу Z-Wave

На відміну від Wi-Fi та інших стандартів передачі даних IEEE 802.11, призначених в основному для великих потоків інформації, стандарт Z-Wave працює в діапазоні частот до 1 ГГц та оптимізований для передачі простих керуючих команд (наприклад, увімкнути/вимкнути, змінити гучність, яскравість і т.д.). Вибір низького радіочастотного діапазону для Z-Wave обумовлюється малою кількістю потенційних джерел перешкод (на відміну від завантаженого діапазону 2,4 ГГц, в якому доводиться вдаватися до заходів, що зменшують можливі перешкоди від працюючих різних побутових бездротових пристроїв – Wi-Fi, ZigBee, Bluetooth). У Європі використовується частотний діапазон 868,42 МГц.

Іншими перевагами стандарту можна відзначити мале споживання енергії, низьку вартість виробництва та вбудовування модулів Z-Wave у різні побутові пристрої.

Швидкість передачі даних у мережі становить 9,6 кбіт/с або 40 кбіт/с із повною сумісністю. Використовується модуляція GFSK. Радіус дії приблизно 30 метрів в умовах прямої видимості в приміщенні

зменшується в залежності від форми і матеріалу стін, а також від виду антени.

У мережі Z-Wave вузли поділяються на три типи: *контролери* (Controllers), *ведені маршрутизатори* (Routing Slaves) і *ведені пристрої* (Slaves). У реальній мережі всі типи пристроїв можуть працювати в будь-якій комбінації.

Контролер – цей пристрій верхнього рівня забезпечує таблицю маршрутизації для mesh-мережі та є хостом/майстром mesh-мережі. Існує два основних типи контролерів:

– Головний контролер – основний контролер є провідним, і лише один майстер може існувати у мережі. Він має можливість підтримувати топологію та ієрархію мережі. Він також може включати чи виключати вузли з топології. Він також повинен виділяти ідентифікатори вузлів.

- Вторинний контролер - ці вузли допомагають первинному контролеру з маршрутизацією;

Ведений пристрій/вузол – ці пристрої виконують дії на основі команд, які вони отримують. Ці пристрої не можуть зв'язуватися з сусідніми підлеглими вузлами, за винятком випадків, коли це передбачено інструкцією команди. Ведені пристрої можуть зберігати інформацію про маршрутизацію, але не обчислювати або оновлювати таблиці маршрутизації. Як правило, вони будуть виступати як ретранслятор в mesh-мережі.

Контролери також можуть бути визначені як переносні та статичні. Портативний переносний контролер призначений для переміщення як пульт дистанційного керування. Як тільки він змінить положення, він перерахує найшвидші маршрути мережі. Статичний контролер призначений для встановлення, наприклад, шлюзу, підключеного до розетки. Статичний контролер завжди може бути включений та отримувати повідомлення про стан підлеглого пристрою.

Контролери також можуть мати різні атрибути в мережі:

контролер оновлення стану (SUC) – статичний контролер також має перевагу у ролі контролера оновлення стану. У цьому випадку він отримуватиме повідомлення від основного контролера щодо змін топології. Він також може допомогти в маршруттизації ведених;

SUC ID-сервер (SIS) – SUC також може допомогти увімкнути та вимкнути підпорядковані пристрої для основного;

Контролер моста – це, по суті, статичний контролер, який може діяти як шлюз між мережею Z-Wave та іншими мережевими системами (наприклад, WAN або Wi-Fi). Мосту дозволено керувати до 128 віртуальних підпорядкованих вузлів;

Контролер установника – це портативний контролер, який може допомогти в управлінні мережею та аналіз якості обслуговування.

Ведені також підтримують різні атрибути:

ведений пристрій маршрутизації - в основному, підлеглий вузол, але з можливістю відправки повідомлень, що не запитуються, іншим вузлам в mesh-мережі. Як правило, веденим пристроям заборонено надсилати повідомлення іншому вузлу без команди основного контролера. Вузол зберігає набір статичних маршрутів, які він використовує для надсилання повідомлення;

розширений ведений пристрій - вони мають ті ж здібності, що і ведений пристрій маршрутизації, з додаванням годинника реального часу і постійної пам'яті для даних програми. Прикладом може бути газовий лічильник;

Z-Wave використовує пористу топологію мережі з маршрутизацією повідомлень від джерела (англ. source routing) і має один основний контролер і нуль або більше вторинних контролерів, які керують маршрутизацією та безпекою. У пористій мережі Z-Wave кожен вузол або пристрій може приймати та передавати керуючі сигнали іншим пристроям мережі, використовуючи сусідні проміжні вузли. Це мережа з маршрутизацією, що самоорганізується, залежною від зовнішніх факторів – наприклад, при виникненні перешкоди між двома найближчими вузлами мережі, сигнал піде через інші вузли мережі, що знаходяться в радіусі дії.

Таким чином, мережа Z-Wave може мати радіус передачі набагато більший, ніж дальність передачі одного вузла. Однак через переприйом (hops) може бути отримана невелика затримка між командою управління та бажаним результатом. Для того щоб Z-Wave пристрої мали можливість маршрутизувати дані, що ними не запитуються, вони не можуть перебувати в сплячому режимі. Таким чином, пристрої з живленням від батарейок не призначені як пристрої ретрансляції. Мережа Z-Wave може включати до 232 пристроїв з можливістю розширення мережі, якщо потрібно ще кілька пристроїв. Додаткові пристрої в мережу можуть бути додані в будь-який час, так само як і кілька контролерів, що управляють.

Хоча технологія Z-Wave є простим та дешевим рішенням, низька швидкість передачі даних виключає передачу зображень, звуку та високошвидкісних даних.

Крім того, для рішень, де потрібно понад 30 пристроїв, Z-Wave-система є більш дорогою, ніж кабельні системи. Через свої конструктивні особливості такі системи мають обмежені масштаби і радіус дії, і вимагають використання повторювачів або навіть кабельних

з'єднань. У світі налічується понад 200 виробників, що пропонують товари з Z-Wave чіпами чи модулями. Особливістю Z-Wave є те, що всі ці продукти сумісні між собою.

3.2. WPAN та WLAN на основі IP

Було б дивним говорячі про Інтернет речей не розглянути основні класичні інтернет-технології та протоколи. Однак, враховуючи, що студенти проходять спеціалізовані курси по мережних технологіях у цьому курсі будемо розглядати лише ті технології які найбільше відносяться до IoT.

3.2.1. WPAN з IP – стандарт 6LoWPAN

6LoWPAN (англ. IPv6 over Low power Wireless Personal Area Networks) — стандарт взаємодії за протоколом IPv6 поверх малопотужних бездротових персональних мереж стандарту IEEE 802.15.4, а також назва робочої групи IETF, що проектує цей стандарт.

Основна мета розробників полягає в тому, щоб використовувати IP-мережі через системи зв'язку RF з низьким енергоспоживанням для пристроїв, які обмежені в потужності та просторі та не потребують високошвидкісних мережевих сервісів [1].

Основною перевагою 6LoWPAN є те, що найпростіший датчик може мати IP-адресність і діяти як учасник мережі через маршрутизатори 3G/4G/LTE/Wi-Fi/Ethernet. Додатковим ефектом є те, що IPV6 забезпечує значну теоретичну адресованість 2^{128} чи $3{,}4 \times 10^{38}$ унікальних адрес.

Мережі 6LoWPAN є змішаними (mesh-) мережами, розташованими на периферії великих мереж. Їх топології гнучкі, що дозволяє створювати спеціальні та незв'язані мережі без прив'язки до інтернету чи інших систем, або ж вони можуть бути підключені до магістралі чи інтернету з використанням граничних маршрутизаторів. Мережі 6LoWPAN можуть бути поєднані з кількома прикордонними маршрутизаторами – це називається *багатоточковим підключенням*.

Архітектура 6LoWPAN-мережі

На рис. 3.9 показаний приклад мережі IPv6, що включає комірчасту (mesh) 6LoWPAN-мережу. Вихідний канал до Інтернету забезпечується точкою доступу, що діє як маршрутизатор IPv6. У типовій конфігурації до точки доступу підключається кілька різних пристроїв, таких як PC, сервери і так далі. 6LoWPAN-мережа пов'язана з IPv6-мережею за допомогою використання *граничного*

маршрутизатора (edge router). Інколи також вживаються терміни: «крайовий»/«прикордонний» розтер чи маршрутизатор. Граничний маршрутизатор виконує три дії: обмін даними між пристроями 6LoWPAN та Інтернетом (або іншою IPv6-мережею), локальний обмін даними між пристроями у 6LoWPAN-мережі та формування та обслуговування радіопідмережі (6LoWPAN-мережі).

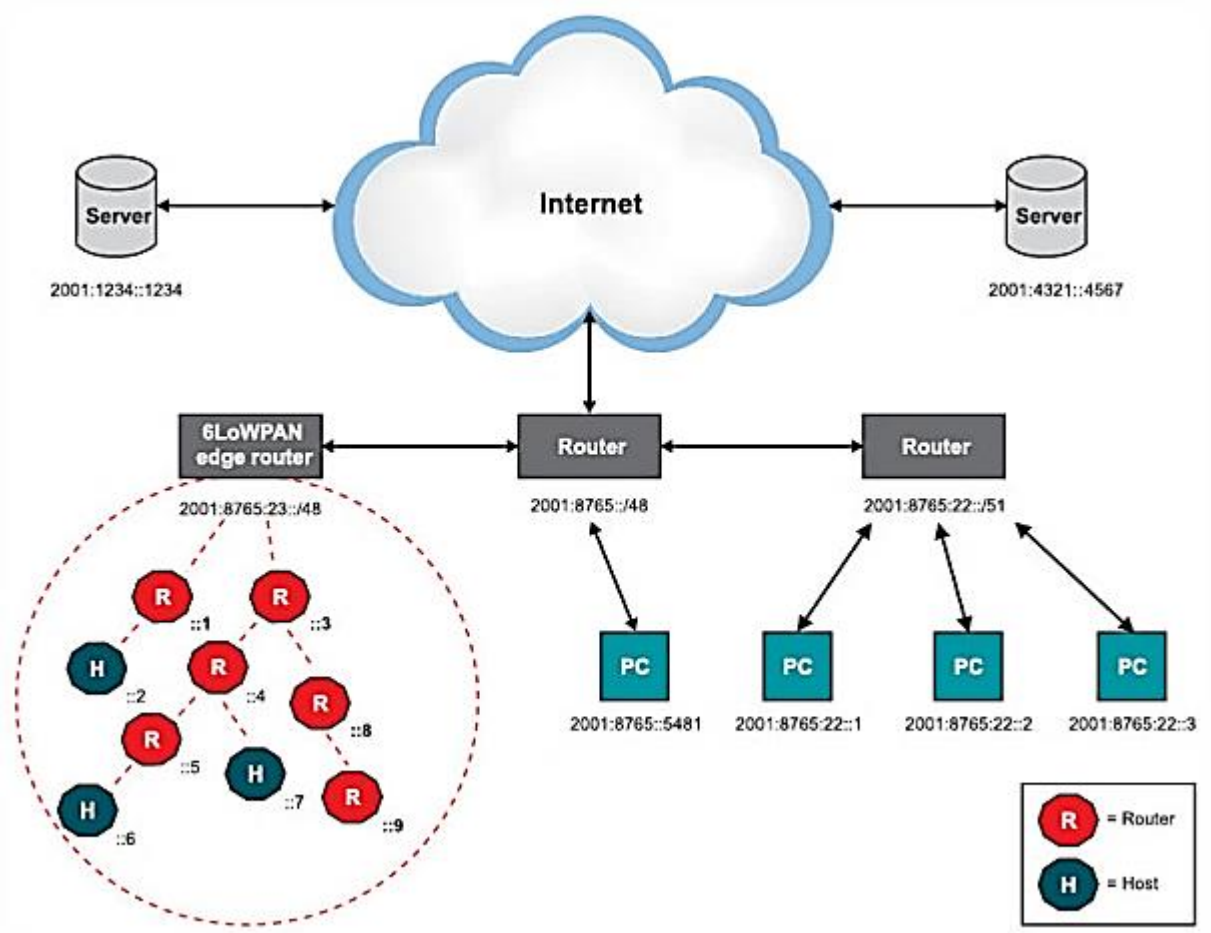


Рис. 3.9. Приклад мережі IPv6 з комірчастою 6LoWPAN-мережею

Взаємодіючи природним способом з IP у рідному форматі, 6LoWPAN-мережі зв'язуються з іншими мережами через стандартні IP-маршрутизатори. З рис. 3.9 слідує, що мережі 6LoWPAN, як правило, працюватимуть як кінцеві на межі глобальної мережі. Одна 6LoWPAN-мережа може бути пов'язана з іншими IP-мережами через один або більше граничних маршрутизаторів, які відправляють IP-датаграми між різними середовищами передачі. Забезпечення зв'язку з іншими IP-мережами може надаватися через будь-який довільний канал, такий як Ethernet, Wi-Fi або 3G/4G. Оскільки 6LoWPAN тільки конкретизує операції IPv6 поверх стандарту IEEE 802.15.4, граничні маршрутизатори можуть також підтримувати механізми переходу IPv6, щоб з'єднувати 6LoWPAN-мережі з IPv4-мережами, наприклад механізм NAT64, визначений в RFC 6146.

Оскільки граничні маршрутизатори надсилають датаграми на мережному рівні, вони не підтримують стан прикладного рівня. Мережі з іншою архітектурою, такі як ZigBee, Z-Wave, Bluetooth або фірмові мережі, вимагають, щоб шлюзи виконували складні прикладні програми, які дозволяють перетворити специфічний трафік бездротової мережі перед надсиланням даних у стандартний IP-канал. Ці прикладні шлюзи повинні розуміти будь-які прикладні профілі, які можуть використовуватися в мережі, і будь-які зміни прикладних протоколів бездротових вузлах повинні також супроводжуватися змінами на шлюзі. Навпаки, засновані на протоколах IP маршрутизатори на межах мереж, наприклад, граничний маршрутизатор, не залежить від прикладних протоколів, що використовуються в 6LoWPAN. Це знижує навантаження на граничний маршрутизатор, наприклад, на його обчислювальну потужність, дозволяючи таким чином використовувати пристрої з низькою вартістю, більш простим програмним забезпеченням і менш складними апаратними засобами. При цьому архітектура IP не перешкоджає використанню для оптимізації роботи мережі проксі та кеш-пам'яті, які широко використовуються у сучасному Інтернеті.

У типову 6LoWPAN-мережу включені ще два пристрої: *роутери* та *хости*. Роутери, як визначається за їх назвою, можуть ретранслювати дані, призначені іншим вузлам у 6LoWPAN-мережі. Хости, які також називаються кінцевими пристроями, не в змозі направляти дані на інші пристрої в мережі. Хост може також бути «сплячим» пристроєм, який «прокидається» періодично, щоб перевірити наявність даних у свого «батька» (роутера), що дозволяє споживати дуже малу потужність.

Системний стек

6LoWPAN радикально змінює підхід Інтернету речей (IoT). Як сказано вище, до цього часу був необхідний складний шлюз прикладного рівня, щоб з'єднати такі пристрої як ZigBee, Bluetooth та фірмові системи з Інтернетом. 6LoWPAN вирішує цю дилему, вводячи рівень адаптації між канальним рівнем стека IP (link) і мережевими рівнями (network layers) бездротового сегмента, щоб забезпечити передачу IPv6 датаграм по радіоканалу IEEE 802.15.4.

На рис. 3.10 показано спрощену модель OSI та пристрій, підключений до IoT і заснований на 6LoWPAN.

Фізичний рівень перетворює біти даних сигнали, які передаються і виходять за ефіром. У 6LoWPAN, наприклад, використовується IEEE 802.15.4. Крім широко відомої версії стандарту 2006 року існують дві

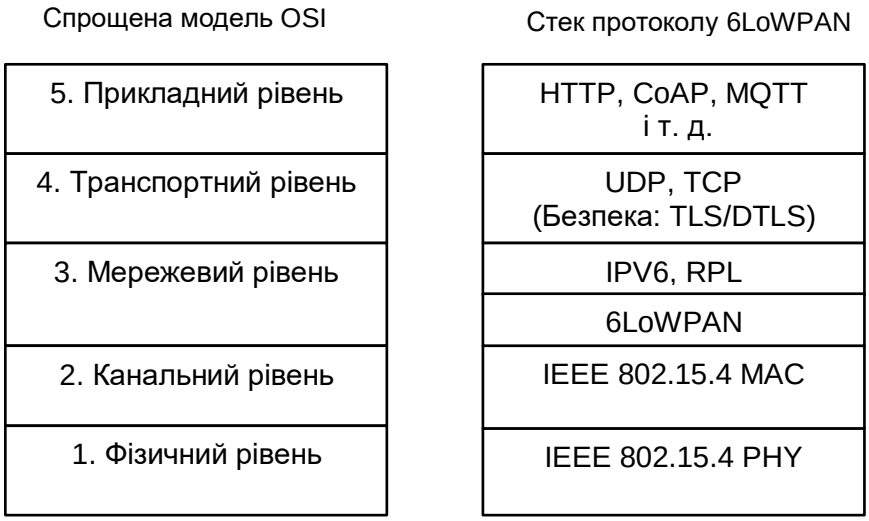


Рис. 3.10. Приклади моделі OSI та 6LoWPAN

важливі поправки: e і g. IEEE 802.15.4e – поправка до MAC, вона пропонує такі розширення, як перемикання каналів із виділенням квантів часу (TSCH – time slotted channel hopping) та координоване виборче прослуховування (CSL – coordinated sampled listening). Обидва розширення націлені на додаткове зниження витрати енергії та роблять бездротовий інтерфейс більш стійким. IEEE 802.15.4g – поправка до PHY (фізичного рівня), мета якої – забезпечити додатковий діапазон радіочастот, щоб дозволити їх використання по всьому світу навіть у діапазонах субгігагерцевих частот.

Канальний рівень забезпечує надійну передачу даних між двома безпосередньо пов'язаними вузлами за рахунок виявлення та виправлення помилок, які можуть виникнути фізично під час передачі та прийому. Канальний рівень включає рівень доступу до середовища передачі (MAC), що використовує метод множинного доступу з контролем несучої та виключенням зіткнень (CSMA-CA), відповідно до якого радіостанція прослуховує ефір, визначаючи, чи веде передачу якась інша станція, перед початком передачі даних у ефір. Цей рівень також забезпечує синхронізацію даних. У 6LoWPAN, наприклад, рівень MAC визначається стандартом IEEE 802.15.4. Рівень адаптації 6LoWPAN, забезпечуючи перехід від IPv6 до IEEE 802.15.4, також знаходиться на канальному рівні.

Мережевий рівень підтримує адресацію та маршрутизацію даних через мережу, якщо потрібно зробити кілька ретрансляцій. IP (міжмережевий протокол) — мережевий протокол, який використовується для надання всім пристроям IP-адрес для транспортування пакетів від одного пристрою до іншого.

Транспортний рівень відкриває сесії зв'язку між програмами, що запускаються на кінцевих пристроях. Транспортний рівень дозволяє декільком програмам на кожному пристрої мати власний канал зв'язку. TCP – домінуючий транспортний протокол в Інтернеті. Однак TCP заснований на протоколі з'єднання (включаючи впорядкування пакетів) з великим обсягом службової інформації і тому не завжди підходить для пристроїв, що потребують зниження споживання енергії. Для систем такого типу найкращою опцією може бути UDP – протокол без встановлення з'єднання з меншим обсягом службової інформації. Прикладом безпечного транспортного рівня є протокол TLS (безпека транспортного рівня), який виконується поверх TCP та DTLS та заснований на UDP.

Зрештою, прикладний рівень відповідає за форматування даних. Він також перевіряє, що дані транспортуються за оптимальною для застосування схемою. Широко використовуваний прикладний рівень в Інтернеті - HTTP, що запускається поверх TCP. HTTP використовує XML - текстова мова з великим обсягом службової інформації. Тому не оптимально використовувати HTTP у багатьох системах 6LoWPAN. Однак HTTP все ще може бути дуже корисним для зв'язку між 6LoWPAN та Інтернетом. Тому промислові та громадські організації розробили альтернативні протоколи прикладного рівня, наприклад, обмежений прикладний протокол (COAP) — протокол передачі повідомлень поверх UDP з біт-оптимізованим механізмом REST, дуже схожий на HTTP. COAP визначає ретрансляцію повідомлень з підтвердженням і без, підтримку «сплячих» пристроїв, передачу блоків, підтримку підписки та виявлення сервісів. COAP також легко перетворити на HTTP через проксі.

Ще один протокол прикладного рівня, який слід згадати, це транспортний протокол черг повідомлень (MQTT) – протокол з відкритим вихідним кодом, запропонований IBM. MQTT – це протокол типу «видавець/передплатник», що запускається поверх TCP. Оскільки він відіграє важливу роль у IoT, то будемо вивчати його пізніше більш детально.

Є ще багато доступних протоколів прикладного рівня, які працюють поверх TCP/UDP.

Інтернет-протокол версії 6 (IPv6) поверх IEEE 802.15.4

Сучасний Інтернет (і безліч автономних IP-мереж) заснований головним чином на IPv4 і використовує 32-бітові адреси. Вичерпання адрес IPv4 відбулося 3 лютого 2011 року. Це обмеження IPv4 стимулювало розвиток IPv6 у 1990-х роках, його комерційне розгортання розпочалося з 2006 року. IPv6 покриває адресний простір

2128, тобто має 3,4 1038 унікальних адрес. Цього має бути достатньо для масштабування Інтернету протягом багатьох десятиліть.

Щоб знайти можливість збільшення пропускної здатності, IPv6 збільшує максимальний розмір блоку, що передається (MTU) з 576 до 1280 байтів. В IPv6 також відображені зміни та вдосконалення технологій використання Інтернету на канальному рівні. Ethernet - домінуюча технологія зв'язку, і його пропускна здатність збільшується рік у рік. У Wi-Fi, як і в Ethernet, підтримуються MTU аналогічного розміру та дуже високі канальні швидкості. І Ethernet, і Wi-Fi працюють, зазвичай, на багатофункціональних пристроях із значним запасом обчислювальної потужності. З іншого боку, IEEE 802.15.4 був спроектований щоб обслуговувати різний ринок додатків, де потрібна велика кількість недорогих пристроїв, що мало споживають. Пропускна здатність у цьому стандарті обмежена 250 кбіт/с, а довжина кадру обмежена 127 байт, що гарантує низьку частоту появи помилок пакетів і бітів під час передачі радіоканалом. Крім того, IEEE 802.15.4 використовує дві адреси: 16-бітову коротку адресу та розширену адресу EUI-64. Ці адреси зменшують загальний обсяг службової інформації в заголовку пакета та мінімізують вимоги до пам'яті. Крім того, 6LoWPAN працює зазвичай у mesh-мережі з ретрансляцією пакетів через кілька вузлів, що є фундаментальною відмінністю від мережі на основі Ethernet або Wi-Fi. Нарешті, пристрої, реалізовані на основі 6LoWPAN, зазвичай мають обмежені ресурси, наприклад, оперативну RAM-пам'ять приблизно 16 кбайт і Flash 128 кбайт.

Внаслідок вищезгаданих обмежень ресурсів та багатоскачкової топології 6LoWPAN, підтримка IPv6 поверх мереж IEEE 802.15.4 створює кілька проблем:

Датаграми IPv6, звичайно, непридатні для мереж IEEE 802.15.4. Низька пропускна здатність, обмежена буферизація та датаграми, розмір яких дорівнює 1/10 мінімального MTU IPv6, роблять необхідним стиснення заголовків та фрагментацію даних. Наприклад, заголовки IEEE 802.15.4 можуть обмежити ефективне можливе корисне навантаження 81 байтом. У порівнянні з цим заголовки IPv6 (40 байтів), TCP (20 байтів) та UDP (8 байтів) здаються занадто великими.

Так як мережі IEEE 802.15.4 мають одночасно і малу потужність, і низьку пропускну здатність, і, крім того, як середовище передачі використовується радіоканал, вони більш схильні до паразитної інтерференції, відмов та асиметричності каналів (*A* може чути *B*, але *B* не може чути *A*). Ці характеристики вимагають, щоб мережевий рівень був адаптивним і гнучким, і водночас малоспоживаючим та ефективним.

Найбільш загальна мережева топологія для 6LoWPAN - малокористувальна мережа (low-power mesh network). Це заперечує, що канальний рівень є єдиним місцем для широкомовних повідомлень. Цей аспект дуже важливий, оскільки торкається самих основ IPv6, де засоби виявлення сусідніх вузлів покладаються саме на цей домен (link domain).

Всі вищезгадані проблеми стосуються і стандарту 6LoWPAN.

Рівень адаптації 6LoWPAN

Під час передачі даних за рівнями MAC та PHY завжди використовується рівень адаптації. Наприклад, RFC 2464 визначає, як пакет IPv6 інкапсулюється у кадр Ethernet. Для 6LoWPAN RFC 6282 визначає, як кадр даних IPv6 інкапсулюється поверх радіоканалу IEEE 802.15.4.

Основну увагу робоча група IETF, 6LoWPAN WG, приділяє оптимізації передачі пакетів IPv6 по малопотужних мереж з втратами (LLN), таким як IEEE 802.15.4. Це призвело до публікації документа RFC 6282, який визначає таке:

Стиснення заголовка. 40-байтні IPv6 та 8-байтні заголовки UDP стискаються з урахуванням використання загальних областей. Області заголовка ігноруються, якщо можуть бути отримані з канального рівня. Спосіб стиснення заголовків став одним із факторів, чому дійшли стандарту, що підтримує тільки IPv6, а не IPv4. Відзначимо, що немає нічого, що заважало б виконувати TCP у системі 6LoWPAN, але стиснення заголовка TCP не стало частиною RFC 6282.

Фрагментація та повторне складання. Канал зв'язку IEEE 802.15.4 з максимальною довжиною кадру 127 байтів не збігається з MTU IPv6, що становить 1280 байтів. Слід зазначити, що формат кадру IEEE 802.15.4g не має такого обмеження.

Безконтекстне автоконфігурування – це процес, завдяки якому у 6LoWPAN-мережі автоматично генерують власні адреси IPv6. Є методи, що дозволяють уникнути випадку, коли два пристрої отримують одну і ту ж адресу; це називають виявленням дублювання адреси (DAD).

Ключовим поняттям рівня адаптації 6LoWPAN є безконтекстний або спільно використовуваний контекстний стиск з метою ігнорування областей заголовка. Це дозволяє стискати всі заголовки (рівня адаптації, мережного та транспортного рівня) аж до кількох байтів. Області заголовка можна стискати, оскільки часто мають загальні значення. Загальні значення виникають через частого використання підмножини функціональних можливостей IPv6, а саме UDP, TCP і ICMP. Можна також зробити припущення щодо спільно використовуваного контексту, наприклад, загального мережевого префікса для всієї системи

6LoWPAN. Рівень адаптації 6LoWPAN також видаляє дубльовану інформацію, таку як IPv6 адреси і поле довжини UDP/IPv6, які можуть бути отримані з інших рівнів.

Стиснення заголовка

Традиційний спосіб стиснення заголовка IP є базовим і використовується в з'єднаннях «точка-точка» (point-to-point), де потік між двома точками стійкий. Це стиснення також дуже ефективно у статичних мережах із стійкими каналами. Комунікація з багаторазовими стрибками вимагає стиснення/відновлення даних при кожному стрибку. Протоколи маршрутизації (наприклад, RPL), зазвичай працюють у системах 6LoWPAN, забезпечують безліч шляхів доставки повідомлення одержувачу з допомогою технології прокладання маршрутів. Це потребує оновлення інформації про вузли, що суттєво зменшує ефективність стиснення. Для динамічно змінних мереж з багаторазовими стрибками та нечастими передачами, наприклад 6LoWPAN-радіомереж, має бути застосований інший метод. Натомість у 6LoWPAN використовується контекстно-незалежний або спільно використовуваний контекстний стиск (stateless and shared-context), при якому не потрібно ніякої інформації про поточний стан, а протоколи маршрутизації можуть динамічно вибирати маршрути, не торкаючись ступеня стиснення.

У прикладі на рис. 3.11 відображено три сценарії комунікації [1]:

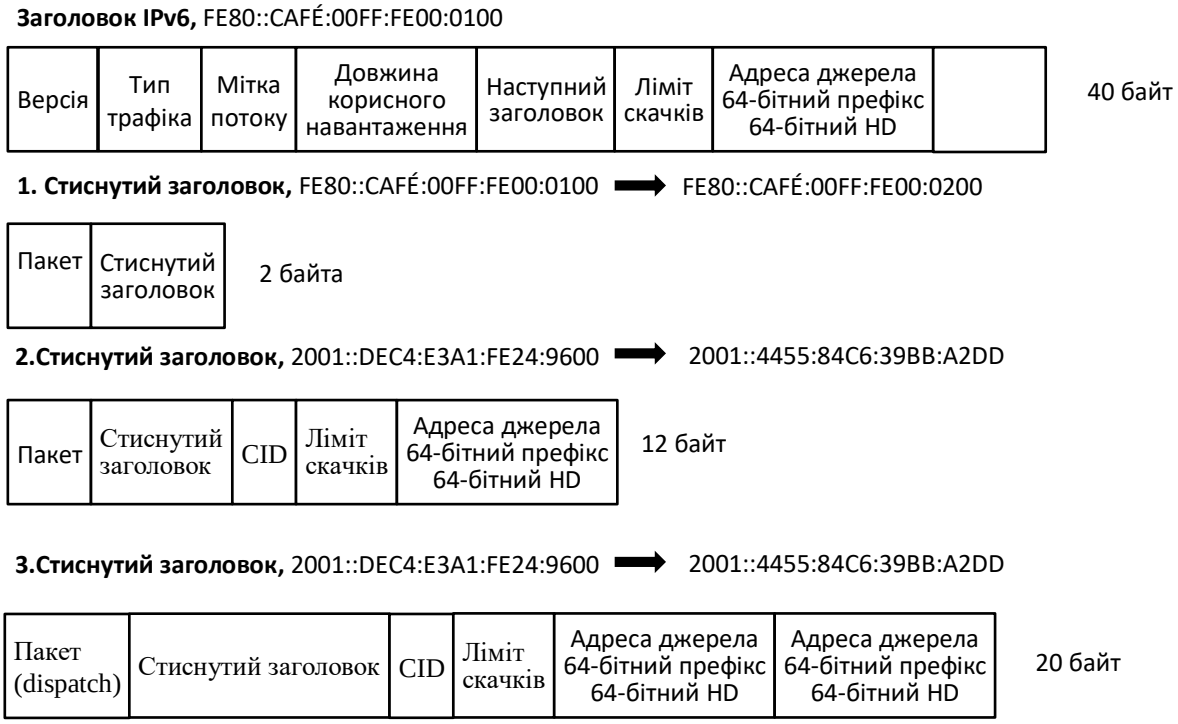


Рис. 3.11. Три сценарії комунікації

- Комунікація між двома пристроями в одній і тій же 6LoWPAN-мережі, що використовує локальні адреси каналів, IPv6 заголовок при цьому може бути стиснутий всього до 2 байтів.
- Комунікація у напрямку пристрою за межами 6LoWPAN-мережі, префікс зовнішньої мережі відомий, тоді заголовок IPv6 може бути стиснутий до 12 байтів.
- Аналогічно другому, але префікс зовнішнього пристрою не відомий, що дає назву IPv6 20 байтів.

Кращий випадок у цьому прикладі (перший) не придатний для передачі прикладних даних (оскільки це може використовуватися тільки для передачі даних найближчим сусідам), проте здатність стиснення заголовка при обміні даними між двома сусідніми пристроями є особливо важливою для протоколу маршрутизації. Найгірший випадок (третій) все ще дає ступінь стиснення 50%. У прикладі передбачається, що ID інтерфейсу (IID) отримано з MAC-адреси пристрою. Потрібно також зауважити, що стиснення заголовка UDP, як уже сказано вище, є частиною стандарту 6LoWPAN, що не відображено у цьому прикладі.

Фрагментація та повторне складання

Щоб допустити передачу кадрів IPv6 поверх радіоканалів IEEE 802.15.4, кадри IPv6 повинні бути розділені на декілька менших сегментів. З цією метою у заголовках формуються додаткові дані, щоб повторно зібрати пакети у правильній послідовності на приймальній стороні. Коли пакети даних повторно зібрані, додаткова інформація видаляється та пакети відновлюються у початковому форматі IPv6. Послідовність фрагментації різна залежно від того, яка маршрутизація використовується (різні методи маршрутизації обговорюються нижче). У разі маршрутизації mesh-under фрагменти повторно збираються тільки в кінцевій точці призначення, тоді як у разі маршрутизації route-over дані повторно збираються при кожному стрибку. Таким чином, у мережі mesh-under при кожному стрибку необхідно мати чималі ресурси для зберігання всіх фрагментів. Треба взяти до уваги, що в системі mesh-under більшість мережевого трафіку формується дуже швидко, тому що всі фрагменти передаються негайно. Якщо якісь фрагменти під час повторного збирання (у системі mesh-under) відсутні – має бути повторно передано повний пакет. Якщо можливо, фрагментації потрібно уникати максимально довго, оскільки вона негативно впливає на термін служби акумулятора пристрою, тому збереження низького корисного навантаження (включаючи вибір відповідних протоколів прикладного рівня) та використання стиснення заголовків мають важливе значення.

Формати заголовка

6LoWPAN використовує стек заголовків і, аналогічно IPv6, розширені заголовки. У заголовках 6LoWPAN (рис. 3.12) регламентовані функції кожного з трьох підзаголовків (адресація комірки, фрагментація та стиснення заголовка). Адресація осередку підтримує дворівневе пересилання (канал зв'язку), фрагментація підтримує передачу MTU IPv6. Формат заголовка визначається за допомогою поля типу заголовка, розміщеного на початку кожного заголовка. Стек заголовка просто розібрати та врахувати підзаголовки, які будуть видалені, якщо в них немає потреби. Заголовок фрагментації ігнорується для пакетів, які вкладаються в один кадр IEEE 802.15.4. Заголовок осередку не використовується, якщо дані надсилаються лише між сусідніми вузлами.

Рис. 3.12. Розташовані у вигляді стека заголовки 6LoWPAN

Заголовок фрагмента використовується, коли корисне навантаження занадто велике, щоб помістится в одному кадрі IEEE 802.15.4. Заголовок фрагмента містить три поля: розмір датаграми, тег датаграми та її усунення. Розмір датаграми визначає загальний обсяг (нефрагментованого) корисного навантаження. Тег датаграми ідентифікує набір фрагментів і використовується, щоб зіставити фрагменти одного і того ж корисного навантаження. Зміщення датаграми ідентифікує усунення фрагмента в межах нефрагментованого корисного навантаження. Довжина заголовка фрагмента становить 4 байти для першого заголовка та 5 байтів для всіх наступних.

Заголовок адреси осередку використовується для передачі пакетів через кілька стрибків у 6LoWPAN-мережі. Заголовок адреси осередку включає три поля: число стрибків, адреси джерела та одержувача. Поле числа стрибків використовується, щоб обмежити кількість ретрансляцій усередині мережі при пересиланні від вузла до вузла. Поле зменшується на одиницю при кожному стрибку. Як тільки лічильник стане рівним нулю, пакет відкидається. Поля адреси джерела та адреси одержувача вказують IP кінцевих точок. Обидва поля містять адреси IEEE 802.15.4 і можуть бути укорочені або розширені, як визначено у стандарті IEEE

802.15.4. Довжина заголовка адреси комірки має 5...17 байтів залежно від способу адресації, що використовується.

Маршрутизація

Маршрутизація [26] – функція передачі пакета даних з одного пристрою на інший, іноді – за допомогою ретрансляцій через кілька проміжних вузлів. Залежно від рівня, на якому розташований механізм маршрутизації, визначено дві категорії: mesh-under або route-over (рис. 3.13). Маршрутизація mesh-under використовує дворівневу (до канального рівня) адресацію (IEEE 802.15.4 MAC або коротку адресу) для передачі пакетів даних, тоді як маршрутизація route-over використовує трирівневу (до мережного рівня) адресацію (IP-адреси).

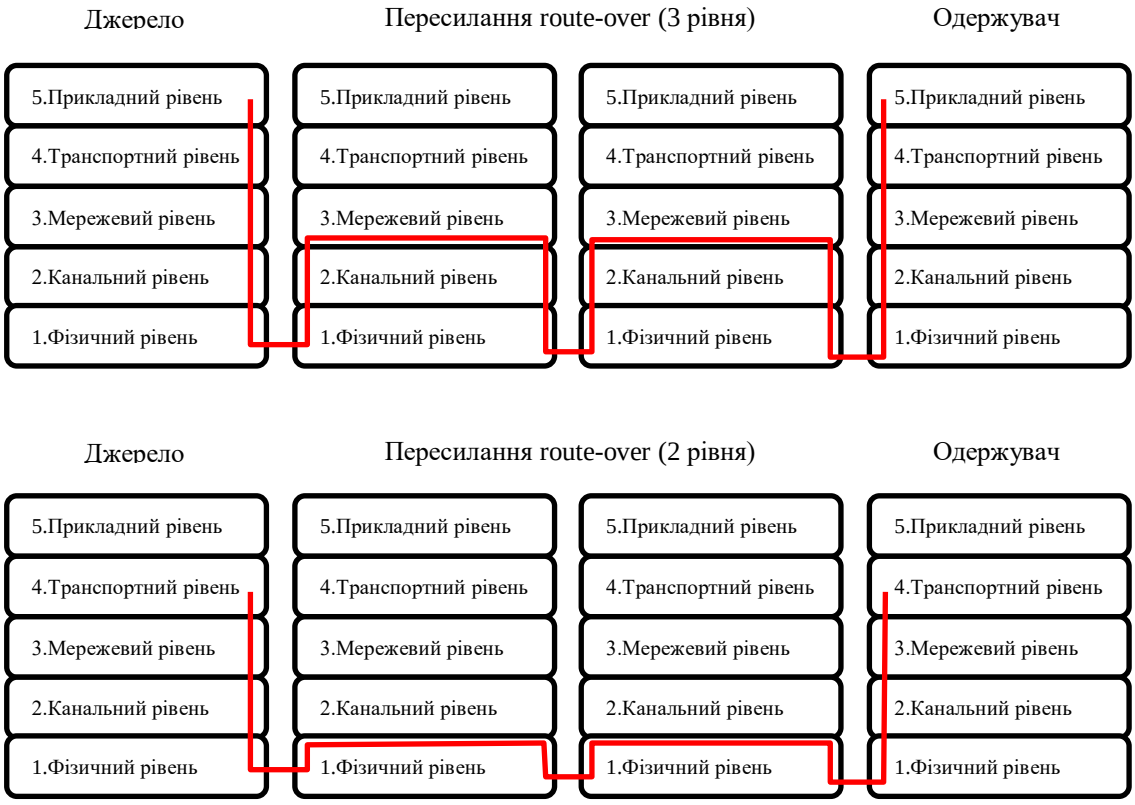


Рис. 3.13. Пересилання пакетів mesh-under та route-over

У системі mesh-under маршрутизація даних відбувається прозоро, отже, mesh-under-мережі можна вважати однією IP-підмережею. У такій системі є лише один IP-маршрутизатор — граничний, встановлений один широкомовний домен, щоб гарантувати сумісність з протоколом IPv6 вищого рівня, як-от виявлення дублювання адреси. Ці повідомлення потрібно надіслати у всі пристрої в мережі, що призводить до високого мережного навантаження. Mesh-under найкраще підходить для малих та локальних мереж.

У route-over-мережах маршрутизація має місце на рівні IP, як описано вище. Таким чином кожен стрибок у таких мережах представляє один IP-маршрутизатор. Використання IP-маршрутизації створює передумови для створення більших і потужніших мереж, оскільки кожен маршрутизатор повинен реалізувати всі функції (DAD та інші), що підтримуються нормальним IP-маршрутизатором. Найбільш широко використовуваним протоколом маршрутизації в route-over-мережах 6LoWPAN на сьогодні є RPL ([raipl]; Routing Protocol. У порівнянні з mesh-under, перевага маршрутизації route-over полягає в тому, що більшість протоколів, що використовуються сьогодні в стандартному стеку TCP/IP, може бути реалізовано та використовуватися у незмінному вигляді. RFC 6550 визначає протокол маршрутизації IPv6 для малопотужних мереж з втратами (RPL), який підтримує або трафік «багато до одного» (multipoint-to-point) від пристроїв 6LoWPAN-мережі в центральний пункт управління (наприклад, сервер в Інтернеті), або трафік «один-багатьом» (point-to-multipoint) із центрального пункту управління до пристроїв у 6LoPWAN-мережі.

Також є підтримка трафіку між двома сусідніми вузлами (point-to-point). Однак RPLнеоптимальний вибір для такого трафіку, оскільки дані у багатьох випадках мають передаватись через граничний маршрутизатор. RPL підтримує два різні режими маршрутизації: режим збереження та режим без збереження. У режимі збереження всі пристрої в 6LoWPAN-мережі конфігуруються як маршрутизатори, які підтримують таблицю маршрутизації і зберігають таблицю сусідніх вузлів (neighbor table). Таблиця маршрутизації використовується для того, щоб прокладати шлях для передачі даних на конкретні пристрої в бездротовій мережі, а таблиця сусідніх вузлів – щоб відстежувати найближчих сусідів вузла. У режимі без збереження є єдиний пристрій із таблицею маршрутизації – граничний маршрутизатор. Отже, використовується маршрутизація джерела. Маршрутизація від джерела означає, що пакет включає повний маршрут (або стрибки), необхідний для досягнення адресата. Наприклад, при передачі даних від одного пристрою на інший в одній і тій же 6LoWPAN-мережі дані спочатку надсилаються з вихідного пристрою в граничний маршрутизатор, який, у свою чергу, виконує пошук у своїй таблиці маршрутизації і додає в пакет повний маршрут до адресата. Режим із збереженням накладає більш високі вимоги на пристрої, що діють як маршрутизатори (тобто у них мають бути ресурси, достатні для зберігання таблиць маршрутизації та сусідніх вузлів), тоді як при використанні режиму без збереження службова інформація збільшується пропорційно до кількості стрибків, які пакет має подолати, щоб досягти адресата.

Автоконфігурація та виявлення сусіднього вузла

Автоконфігурація — автономна генерація IPv6-адреси пристрою. Цей процес для IPv4 і IPv6 значно відрізняється. Для IPv6 він дозволяє пристрою автоматично генерувати свою IPv6-адресу без будь-якої взаємодії із зовнішнім сервером DHCP або аналогічним пристроєм. Щоб отримати адресу, хост може зв'язуватися через протокол виявлення сусідніх вузлів (NDP, neighbor discovery protocol). Втім, багато функцій NDP також включені в RPL. Процедура, описана тут, справедлива також для RPL і включає чотири типи повідомлень:

запит у пошуках маршрутизатора (RS, router solicitation);

оголошення роутера (RA, router advertisiment);

запит у пошуках сусідніх вузлів (NS, neighbor solicitation);

оголошення сусіднього вузла (NA, neighbor advertisement).

Виявлення сусіднього вузла IPv6 (ND, network discovery) дозволяє пристрою виявляти сусідів, підтримувати інформацію про їх досяжність, конфігурувати за замовчуванням маршрути і поширювати параметри конфігурації. Повідомлення RS включає, крім іншого, префікс IPv6-мережі. Усі маршрутизатори в мережі періодично надсилають ці повідомлення. Якщо хост хоче брати участь у 6LoWPAN-мережі, він призначає собі індивідуальну адресу локального каналу (FE80::IID), потім посилає цю адресу повідомлення ND всім іншим учасникам підмережі, щоб перевірити, чи використовується адреса кимось ще. Якщо хост не почує повідомлення NA за певний період часу, той припускає, що адреса унікальна. Цю процедуру називають виявленням дублювання адреси (DAD). Тепер, щоб отримати префікс мережі, хост надсилає повідомлення RS-маршрутизатору, щоб отримати правильний префікс. Використовуючи ці чотири повідомлення, хост може призначити собі міжнародну унікальну адресу IPv6.

Використовуючи автоконфігурування адреси джерела, кожен хост генерує IPv6-адресу локального каналу, використовуючи його адресу IEEE 802.15.4 EUI-64, 16-бітову коротку адресу або обидва одночасно. У конфігурації mesh-under діапазон адрес локальних каналів покриває всю 6LoWPAN-мережа, навіть з кількома стрибками, і адрес локальних каналів достатньо для комунікації в 6LoWPAN-мережі. Єдиний раз, коли необхідна маршрутизована IPv6-адреса – це передача даних за межі 6LoWPAN-мережі. У конфігурації route-over достатньо локальних адрес канального рівня для взаємодії з вузлами в межах радіопокриття. Маршрутизована адреса потрібна у разі спілкування з пристроями, віддаленими на кілька стрибків.

Для всіх індивідуальних адрес найефективніше – отримати їх з локальної адреси IEEE EUI-64. У 6LoWPAN зв'язка між каналом, адаптацією та IP-заголовком дозволяє ігнорувати та усувати потребу у визначенні адрес, що призводить до менших заголовків. Так само автоконфігурування має встановити адресацію інтерфейсу так, щоб використовувати загальний префікс, а 6LoWPAN його ігноруватиме. 6LoWPAN може використовувати коротку адресу каналу, щоб отримати IPv6-адресу і, як результат, більш короткі заголовки.

Безпека 6LoWPAN

У загальному випадку безпека 6LoWPAN повинна дозволяти гарантувати конфіденційність даних, а також їх цілісність та доступність мережі. Виділено різні можливості атак на мережі 6LoWPAN, які поділяються на дві категорії:

Зовнішні атаки:

пасивна зовнішня атака: наприклад, прослуховування для виявлення конфіденційної інформації (проблема конфіденційності).

активна зовнішня атака: приклад, відмова у обслуговуванні шляхом глушення радіосигналу з метою паралізувати мережу (проблема доступності).

Внутрішні атаки:

Приклад: проникнення в 6LoWPAN для збору або розповсюдження інформації в зловмисних цілях (проблеми цілісності, конфіденційності та доступності).

Щоб зробити зв'язок по 6LoWPAN якомога безпечнішим, безпека повинна бути реалізована на різних рівнях стека протоколів.

На рівні MAC: алгоритм AES повинен використовуватися для захисту канального рівня.

На мережному рівні:

використання IPsec можливе, але шифрування споживає багато ресурсів, і метод обміну ключами IKEv2 (RFC 5996) використовувати не можна. Необхідною умовою є керування ключем шифрування за допомогою мінімального корисного навантаження, а також обмеження повідомлень, якими обмінюються вузли.

Для мереж 6LoWPAN було реалізовано розширення протоколу SEND (RFC 3971) (скорочення від SEcure Neighbor Discovery protocol) для захисту механізму виявлення сусідів, яке називається LSEND (від англійської Lightweight Secure Neighbor Discovery Protocol, «полегшене відправлення»).

На рівні додатків: наприклад, одним із рішень є встановлення безпеки через SSL.

Були зроблені різні пропозиції щодо оптимізації безпеки 6LoWPAN, такі як:

використання параметрів Timestamp і Nonce заголовка IPv6 для захисту мереж 6LoWPAN від атак фрагментації IP (in) (по-французьки: «атакує фрагментовані IP-пакети»);

стиснення заголовка IPsec для оптимізації корисного навантаження;

реалізація проміжного програмного забезпечення, що дозволяє аналізувати ризики безпеки у 6LoWPAN.

оптимізація шифрування.

6LoWPAN використовує переваги потужної системи безпечної передачі даних AES-128 канального рівня, визначеної IEEE 802.15.4. Безпека канального рівня забезпечується автентифікацією та шифруванням. Крім безпеки канального рівня в 6LoWPAN-системах, демонструє відмінну роботу механізм безпеки транспортного рівня (TLS). TLS, як визначено RFC 5246, працює поверх TCP. У разі обмежених ресурсів, де UDP обрано протоколом транспортного рівня, для забезпечення безпеки на транспортному рівні може використовуватися RFC 6347 (безпека датаграм транспортного рівня). Однак слід зазначити, що реалізація TLS/DTLS вимагає, щоб пристрої мали необхідні ресурси, такі як апаратний механізм кодування, щоб забезпечити використання розширеного набору шифрів та інші. Пристроєм, спеціально розробленим для цих цілей, є бездротовий MCU TI CC2538, в який вбудований потужний центральний процесор ARM® Cortex®-M3 та радіо IEEE 802.15.4. Пристрій має Flash-пам'ять до 512 кбайт та оперативну пам'ять 32 кбайт, також оснащений апаратним механізмом кодування, здатним до підтримки TLS/DTLS.

3.2.2. WPAN з IP – Thread

Thread - відносно новий мережевий протокол для IoT і заснований на IPV6 (6LoWPAN). Його основною метою є домашній зв'язок та домашня автоматизація. Тему було запущено у липні 2014 року з формуванням Thread Group [1].

Грунтуючись на протоколі IEEE 802.15.4 і 6LoWPAN, він має спільність із Zigbee та іншими варіантами 802.15.4, але із суттєвою різницею в тому, що Thread є IP-адресованим. Цей IP-протокол ґрунтується на прикладному та фізичному рівні, що надаються 802.15.4, та таких функціях, як безпека та маршрутизація з 6LoWPAN. Thread також заснований на mesh-мережі, що робить його привабливим для домашніх систем освітлення із 250 пристроями в одній мережі.

Філософія Thread полягає в тому, що, дозволяючи IP-адресацію в найменшому з датчиків і домашніх систем автоматизації можна зменшити енергоспоживання, тому що не потрібно зберігати стан програми, оскільки протокол використовує дейтаграми на мережному рівні. Це також передбачає, що граничний маршрутизатор, на якому розміщується мережа потоків Thread, не повинен обробляти протоколи прикладного рівня і може знизити свою потребу в потужності та обробці. Нарешті, сумісний з IPV6, він по суті безпечний при шифруванні всіх повідомлень з використанням стандарту розширеного шифрування (AES). У ланцюжку потоків може існувати до 250 вузлів із повністю зашифрованим транспортом та автентифікацією. Оновлення програмного забезпечення дозволяє вже існуючому пристрою 802.15.4 бути сумісним із Thread.

Архітектура та топологія Thread

На основі стандарту IEEE 802.15.4-2006, Thread використовує специфікацію для визначення рівнів управління доступом (MAC) та фізичного (PHY). Він працює зі швидкістю 250 Кбіт/с у смузі 2,4 ГГц.

З точки зору топології Thread встановлює зв'язок з іншими пристроями через граничний маршрутизатор (зазвичай це сигнал Wi-Fi в домашньому господарстві). Решта повідомлення заснована на 802.15.4 і утворює сітку, що самовідновлюється. Приклад такої топології показано на рис. 3.14.

Нижче наведено ролі різних пристроїв в архітектурі Thread.

Граничний маршрутизатор по суті є шлюзом. У домашній мережі це буде кросовер зв'язку від Wi-Fi до Thread, він формує точку входу в інтернет із мережі Thread, що проходить під граничним маршрутизатором. Декілька граничних маршрутизаторів допустимі відповідно до специфікації Thread;

Взаємозв'язки є змінними та само відновлюваними. Провідний пристрій керує реєстром призначених ідентифікаторів маршрутизатора. Ведучий також контролює запити на кінцеві пристрої, які підключені для маршрутизаторів (REED), для рекламування на маршрутизатори. Лідер може також виступати у ролі маршрутизатора і мати нащадків – кінцеві пристрої. Протокол призначення адрес маршрутизатора – це протокол обмежень програм (CoAP). Інформація про стан, якою керує провідний пристрій, може зберігатися на інших маршрутизаторах потоків. Це дозволяє забезпечити самовідновлення та перехід на інший ресурс у разі, якщо лідер втрачає зв'язок.

Маршрутизатори потоків керують службами маршрутизації мережі. Маршрутизатор потоків ніколи не входить у стан очікування,

але специфікацією дозволяється, щоб вони знижували свій рівень до REED.

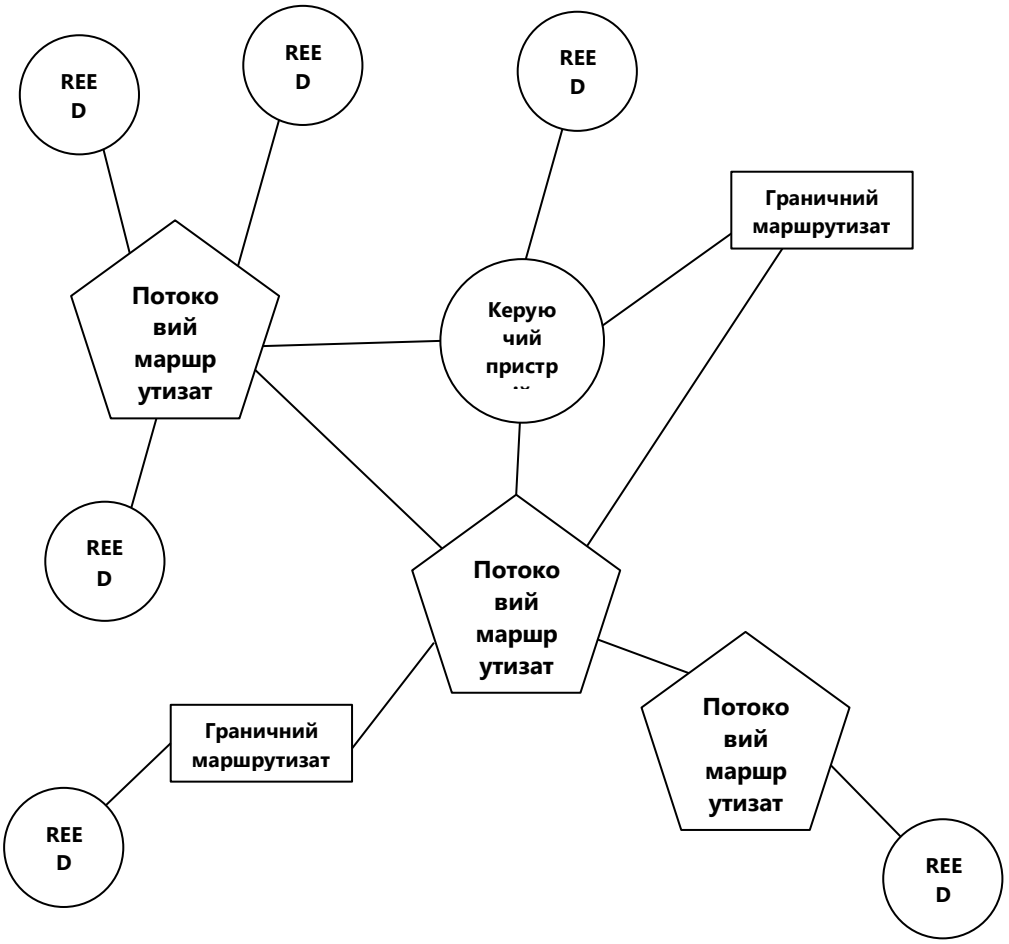


Рис. 3.14. Приклад мережної топології Thread, містить граничні маршрутизатори, маршрутизатори Thread та відповідні пристрої IoT, які можуть об'єднуватися в mesh-мережі.

REED: хост-пристрій, який є REED може стати маршрутизатором або лідером. REED не відповідають за маршрутизацію до mesh мережі, якщо вони не призначаються маршрутизатором або лідером. REED також не можуть надсилати повідомлення або приєднуватися до мережі. По суті REED є кінцевими точками або листовими вузлами мережі.

Кінцеві пристрої: деякі точки не можуть стати маршрутизаторами. Ці типи REED мають дві інші категорії, на які можуть підписатися: повнорозмірні пристрої (FED) і мінімальні кінцеві пристрої (MED).

Сплячі кінцеві пристрої: хост-пристрої, які увійшли в стан очікування, обмінюються даними тільки з їх зв'язаним потоковим маршрутизатором і не можуть надсилати повідомлення.

Стек протоколу Thread

Thread буде використовувати всі переваги 6LoWPAN та користуватися перевагами стиснення заголовків, адресації IPv6 та

безпеки. Thread також використовує схему фрагментації 6LoWPAN, як описано в попередньому розділі, але додає два додаткові компоненти стека (табл. 3.2): дистанційно-векторна маршрутизація; створення mesh-зв'язку.

Таблиця 3.2. Стек протоколу Thread

Стек протоколу Thread	Спрощена модель OSI
HTTP, CoAP, MQTT та ін.	5. Прикладний рівень
Створення mesh-зв’зку (MLE) і TLS/DTLS	4.Транспортний рівень
UPD	
Дистанційно-векторна маршрутизація	3.Мережевий рівень
IPv6	
6LoWPAN	
Рівень MAC IEEE 802.15.4	2. Канальний рівень
PHY IEEE 802.15.4	1.Фізичний рівень

Маршрутизація Thread

Thread використовує обхідну маршрутизацію, як описано в попередньому розділі, маршрутизації 6LoWPAN. У мережі Thread дозволено до 32 активних маршрутизаторів. Обхід маршруту ґрунтується на маршрутизації наступного переходу. Таблиця основних маршрутів підтримується стеком. Усі маршрутизатори мають оновлену копію маршрутизації для мережі.

Створення mesh-зв'язку (MLE) – це спосіб оновлення вартості проходження шляху від одного маршрутизатора до іншого у мережі. Крім того, MLE забезпечує спосіб ідентифікації та налаштування сусідніх вузлів у мережі та забезпеччення їх безпеки. Оскільки mesh-мережа може динамічно розширюватися, стискатися та змінювати форму, MLE забезпечує механізм відновлення топології. MLE розділить інформацію про вартість шляху з усіма іншими маршрутизаторами у стислому форматі. Повідомлення MLE транслюватимуться в мережу широкомовним чином протоколом багатоадресної розсилки для мереж з низьким енергоспоживанням і втратами (MPL).

Типові мережі 802.15.4 використовують знаходження маршруту на вимогу. Це може бути дорогим (пропускна здатність через наповнення мережі повідомленнями про пошук маршруту), а Thread намагається уникнути цієї схеми. Періодично мережевий маршрутизатор Thread обмінюватиметься рекламними пакетами MLE з відомостями про вартість зв'язку зі своїми сусідами, по суті, змушуючи маршрутизатори мати поточний список шляхів. Якщо маршрут переривається (хост

залишає мережу Thread), маршрутизатори намагаються знайти наступний найкращий шлях до місця призначення [1].

Thread також вимірює якість зв'язку. Пам'ятайте, що 802.15.4 WPAN і рівень сигналу може динамічно змінюватися. Якість вимірюється вартістю з'єднання вхідних повідомлень для сусіда зі значенням нуль (невідома вартість) до значення три (хороша якість). Табл. 3.3 підсумовує відношення якості до вартості. Ці якість і вартість постійно контролюються і, як згадувалося, періодично поширюються по мережі для самовідновлення.

Таблиця 3.3. Відношення якості до вартості WPAN

Якість з’єднання	Вартість з’єднання
0	Невідома
1	6
2	2
3	1

Адресація Thread

Щоб знайти маршрут до дочірнього вузла, просто перевіряються старші біти адреси дочірнього елемента, щоб знайти адресу батьківського маршрутизатора. На цьому етапі джерело передачі знає, як дістатися до нащадка, а також інформацію про наступний перехід, щоб розпочати маршрут.

Дистанційно-векторна маршрутизація використовується для пошуку шляхів до маршрутизаторів у мережі Thread. Верхні 6 біт 16-розрядної адреси вказують маршрутизатор призначення – це префікс. Якщо нижні 10 біт адресата встановлені 0, кінцевим пунктом призначення є цей маршрутизатор. Інакше маршрутизатор призначення перенаправлятиме пакет на основі молодших 10 біт (рис. 3.15).

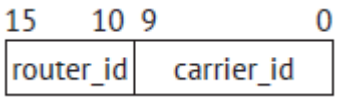


Рис. 3.15. 2-байтова коротка адреса Thread специфікації 802.15.4-2006

Якщо маршрут виходить за межі мережі Thread, граничний маршрутизатор сигналізуватиме лідерам конкретних префіксних даних, увімкнувши дані префікса, контекст 6LoWPAN, граничні маршрутизатори та сервер DHCPv6. Ця інформація передається за допомогою пакетів MLE через мережу Thread.

У межах мережі Thread вся адресація ґрунтується на UDP. Якщо потрібна повторна спроба, мережа Thread спиратиметься на:

повторні спроби рівня MAC: де кожний пристрій, який використовує підтвердження MAC, не отримав ACK від наступного переходу;

повторні спроби рівня додатків: де прикладний рівень забезпечуватиме власний механізм повтору.

Виявлення сусіда

Протокол виявлення сусідів (ND) у потоці вирішує, до якої мережі 802.15.4 приєднатися. Процес виглядає так:

1) приєднання маршрутизатора контактів до входу;

2) пристрій, що з'єднується, сканує всі канали і видає запит маяка на кожному каналі. Чекає на відповідь маяка;

3) якщо з'явиться маяк, що містить корисне навантаження з мережним ідентифікатором набору послуг (SSID) та повідомлення про дозвіл, пристрій тепер приєднається до мережі Thread;

4) як тільки пристрій буде виявлено, повідомлення MLE будуть передані для ідентифікації сусіднього маршрутизатора пристрою. Цей маршрутизатор виконає введення в експлуатацію. Існує два режими введення в експлуатацію:

– конфігурація: використовує позашляховий метод введення пристрою в експлуатацію. Дозволяє підключати пристрій до мережі Thread, як тільки його буде введено в мережу;

– створення: створює сеанс введення в експлуатацію між пристроєм та комісійним додатком, що працює на смартфоні, планшеті чи в інтернеті;

5) пристрій, який з'єднується з батьківським маршрутизатором та підключається до мережі через MLE-обмін.

Пристрій існуватиме як REED або кінцевий пристрій, а батьківській адресі призначається 16-бітова коротка адреса.

3.2.3. Wi-Fi та IEEE 802.11

Основне призначення технологи Wi-Fi (Wireless Fidelity - "бездротова точність") - бездротове розширення мереж Ethernet. Вона використовується також там, де небажано або неможливо використовувати провідні мережі. Наприклад, для передачі інформації від рухомих частин механізмів; якщо не можна свердлити стіни; на великому складі, де комп'ютер потрібно носити з собою.

Wi-Fi розроблений консорціумом Wi-Fi на базі серії стандартів IEEE 802.11 (1997 г.) [ANSI] і забезпечує швидкість передачі від 1 ... 2 до 54 Мбіт/с. Wi-Fi консорціум розробляє прикладні специфікації для втілення стандарту Wi-Fi в життя, виконує тестування і сертифікацію продукції інших фірм на відповідність стандарту, організовує виставки, забезпечує необхідною інформацією розробників Wi-Fi обладнання.

Незважаючи на те, що стандарт IEEE 802.11 був ратифікований ще в 1997 році, мережі Wi- Fi набули широкого поширення тільки в останні роки, коли істотно знизилися ціни на серійне мережеве обладнання. У промисловій автоматизації з безлічі стандартів серії 802.11 використовуються тільки два: 802.11b зі швидкістю передачі до 11 Мбіт / с і 802.11g (до 54 Мбіт / с) [26].

Фізичний і канальний рівень

Модель OSI для стандартів Wi-Fi і IEEE 802.11 показана в табл. 3.4. Основне призначення фізичних рівнів - забезпечення інтерфейсу з бездротовою середовищем передачі (з ефіром), а також оцінка стану ефіру і взаємодія з рівнем MAC.

Таблиця. 3.4. Рівні моделі OSI для Wi-Fi / IEEE 802.11

Номер рівня	**OSI модель**	**Мережа**	**Функції**
7	Прикладний	-	-
6	Рівень представлення	-	-
5	Сеансовий	-	-
4	Транспортний	-	-
3	Мережевий	-	-
2	Канальний (передачи даних)	Підрівень LLC	-
		Підрівень MAC	
1	Фізичний	Підрівень PLCP	Безпровідна передача, оцінка стану ефіра
		Підрівень PMD	

Фізичний рівень складається з двох підрівнів:

- PLCP (Physical Layer Convergence Protocol) - виконує процедуру відображення PDU рівня MAC у фрейм формату FHSS або DSSS . Ця процедура виконує передачу, виявлення несучої і прийом сигналу;

Рис. 3.16. Приклади моделі OSI, стеків Wi-Fi та 6LoWPAN

- PMD (Physical Medium Dependent) - "підрівень, що залежить від середовища передачі". Цей рівень буде різним для різних швидкостей передачі і різних стандартів із серії 802.11. Підрівень PMD забезпечує дані і сервіс для підрівня PLCP і функції радіопередачі і прийому, результатом яких є потік даних, інформація про час, параметри прийому.

Основним робочим станом рівнів PLCP є виявлення несучої і оцінка незайнятості каналу. Для виконання передачі PLCP перемикає PMD з режиму "прийом" в режим "передача" і посилає елемент даних PPDU (PLCP Data Unit).

Фізичний рівень виконує скремблювання, кодування і чергування.

Передача сигналів по радіоканалу виконується двома методами: FHSS і DSSS. При цьому використовується диференціальна фазова модуляція DBPSK і DQPSK із застосуванням кодів Баркера, комплементарних кодів (CCK - Complementary Code Keying) і технології подвійного згорточного кодування (PBCC).

Розширення спектра (англ. Spread Spectrum) - один із способів підвищення ефективності передачі інформації за допомогою модульованих сигналів через канал з сильними лінійними спотвореннями (згасаннями). Розширення спектра приводить до збільшення бази сигналу.

В існуючих на сьогоднішній день системах для цієї мети використовуються три методи:

псевдовипадкове перестроювання робочої частоти (англ. FHSS - Frequency Hopping Spread Spectrum). Суть методу полягає в періодичній стрибкоподібній зміні частоти опорного сигналу за певним алгоритмом, відомим приймачу і передавачу. Перевага методу — простота реалізації, недолік — затримка в потоці даних при кожному стрибку. Метод використовується в Bluetooth; GSM;

розширення спектра методом прямої послідовності (англ. DSSS - Direct Sequence Spread Spectrum). Метод за ефективністю перевершує FHSS, але складніший для реалізації. Суть методу полягає в підвищенні тактової частоти модуляції, при цьому кожному символу переданого повідомлення ставиться у відповідність деяка досить довга псевдовипадкова послідовність. Метод використовується в таких системах як CDMA і системах стандарту IEEE 802.11;

розширення спектра методом лінійної частотної модуляції (ЛЧМ) (англ. CSS - Chirp Spread Spectrum). Суть методу полягає в перебудові частоти-носія за лінійним законом. Метод використовується в радіолокації і в деяких радіомодемах.

Wi-Fi 802.11g на швидкості 1 і 2 Мбіт/с використовує модуляцію DBPSK. При швидкості передачі 2 Мбіт/с використовуються ті ж метод, що і при швидкості 1 Мбіт/с, проте для збільшення пропускної здатності каналу використовується 4 різних значення фази для фазової модуляції несучої.

Протокол 802.11b, використовує додатково швидкості передачі 5,5 і 11 Мбіт/с. На цих швидкостях передачі замість кодів Баркера використовуються комплементарні коди (CCK).

Wi-Fi використовує метод доступу до мережі CSMA/CA, в якому для зниження ймовірність колізій використані наступні принципи:

- перш, ніж станція почне передачу, вона повідомляє, як довго вона буде займати канал зв'язку;

- наступна станція не може почати передачу, поки не закінчиться зарезервоване раніше час;

- учасники мережі не знають, чи прийнятий їх сигнал, поки не отримають підтвердження про це;

- якщо дві станції почали працювати одночасно, вони зможуть дізнатися про це тільки по тому факту, що не отримають підтвердження про прийом;

- якщо підтвердження не отримане, учасники мережі вичікують випадковий проміжок часу, щоб почати повторну передачу.

Запобігання, а не виявлення колізій, є основним в бездротових мережах, оскільки в них, на відміну від провідних мереж, передавач трансивера заглушає сигнал, що приймається.

Формат фрейму на рівні PLCP моделі OSI (табл. 3.4) в режимі FHSS показаний на рис. 3.17. Він складається з наступних полів:

- «Синхронізація." - містить чергуються нулі і одиниці. Служить для підстроювання частоти на приймаючої станції, синхронізує розподіл пакетів і дозволяє вибрати антену (при наявності декількох антен);

- "Старт" - прапор початку фрейма. Складається з рядка 0000 1100 1011 1101, яка служить для синхронізації фреймів на приймаючої станції;

- "PLW" - "Psdu Length Word" - "слово довжини службового елемента даних PLCP", PSDU - "PLCP Service Data Unit" - елемент даних підрівня PLCP; вказує розмір фрейма, що надійшов з рівня MAC, в октетах;

- "Швидкість" - вказує швидкість передачі даних фрейму;

- "КС" - контрольна сума;

- "MAC-фрейм" - фрейм, що надійшов з MAC-рівня моделі OSI і містить PSDU;

- "Тема PLCP" - поля, додані на підрівні PLCP.

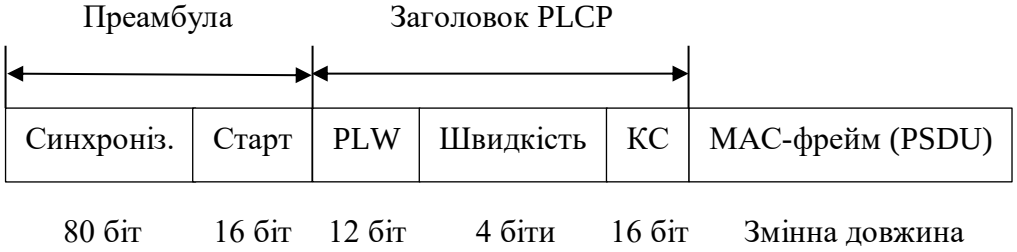


Рис. 3.17. Формат фрейму PLCP для режиму FHSS

Формат фрейму на рівні PLCP моделі OSI (табл. 3.4) в режимі DSSS показаний на рис. 3.18. У ньому поля мають такий зміст:

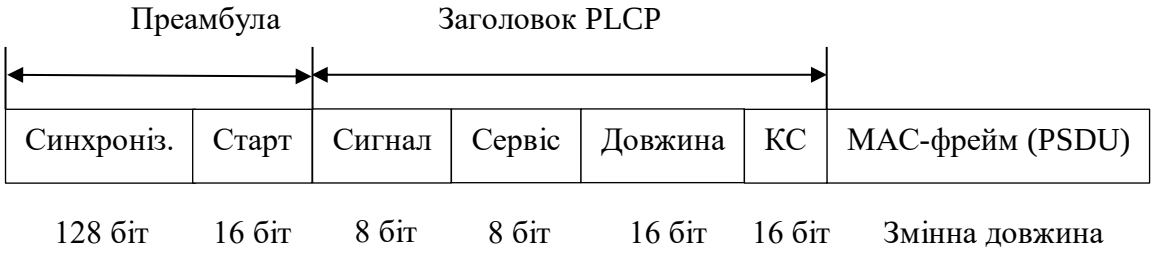


Рис. 3.18. Формат фрейму PLCP для режиму DSSS

- "Синхронізація" - містить тільки одиниці і забезпечує синхронізацію в приймальній станції;

- "Старт" - прапор початку фрейма. Містить рядок 0 xF3A0, яка вказує початок передачі параметрів, що залежать від фізичного рівня;

- "Сигнал" - вказує тип модуляції і швидкість передачі даного фрейму;

- "Сервіс" - зарезервовано для майбутніх модифікацій стандарту;

- "Довжина" - вказує час в мікросекундах, необхідне для передачі MAC-фрейма;

- "КС" - контрольна сума;

- "MAC-фрейм" - фрейм, що надійшов з MAC-рівня моделі OSI і містить PSDU;

- "Тема PLCP" - поля, додані на підрівні PLCP.

Дальність зв'язку засобами Wi-Fi сильно залежить від умов поширення електромагнітних хвиль, типу антени і потужності передавача. Типові значення, що вказуються виробниками Wi-Fi обладнання, складають 100-200 м в приміщенні і до декількох кілометрів на відкритій місцевості із застосуванням зовнішньої антени і при потужності передавача 50...100 мВт.

Архітектура мережі Wi-Fi

Стандарт IEEE 802.11 встановлює три варіанти топології мереж:

- незалежні базові зони обслуговування (Independent Basic Service Sets, IBSS);

- базові зони обслуговування (Basic Service Sets, BSS);

- розширені зони обслуговування (Extended Service Sets, ESS).

Під зоною обслуговування тут розуміється набір логічно згрупованих пристроїв. Кожна зона обслуговування має свій ідентифікатор (Service Set Identifier, SSID). Станція-приймач використовує SSID для визначення того, з якої зони обслуговування приходить сигнал.

В архітектурі IBSS станції зв'язуються безпосередньо одна з одною, без використання точки доступу і без можливості приєднання до дротової локальної мережі. Зона обслуговування SSID використовується зазвичай для об'єднання в мережу малої кількості станцій, оскільки в ній не передбачена можливість ретрансляції сигналу для збільшення дальності зв'язку і механізми для вирішення проблеми прихованого вузла.

При використанні BSS станції спілкуються один з одним через загальний центральний вузол зв'язку, названий точкою доступу. Точка доступу зазвичай підключається до дротової локальної мережі Ethernet.

Розширена зона обслуговування виходить при об'єднанні кількох BSS в єдину систему за допомогою розподільної системи, в якості якої може виступати дротова мережа Ethernet.

3.3. Протоколи дальнього зв’язку

3.3.1. LoRa та LoRaWAN

Протокол LoRaWAN

У січні 2015 року було створено некомерційну організацію LoRa Alliance з метою прийняття та просування протоколу LoRaWAN, як єдиного стандарту для глобальних мереж з низьким енергоспоживанням (LPWAN - від англ. Low Power Wide Area Network). Учасниками різних рівнів LoRa Alliance є виробники програмного забезпечення, мікроелектроніки, оператори зв'язку: і т.д. У LoRa Alliance входять такі компанії, як: IBM, Semtech, Cisco, Inmarsat, Swisscom і інші [1].

Коли говорять про технології LoRa, то найчастіше мають на увазі і метод модуляції LoRa, розроблений Semtech, і відкритий протокол LoRaWAN.

Якщо *модуляція LoRa є, фізичним рівнем* (OSI media layer 1), то *LoRaWAN (Long Range Wide-Area Networks) - це MAC протокол канального* рівня (OSI media layer 2) для мереж з великою кількістю вузлів з великим радіусом дії і низьким власним споживанням потужності. Мережа LoRaWAN має просту архітектуру типу "зірка" (від англ. Star) без ретрансляторів і mesh-зв'язків. Для вузлів мережі характерні: низьке енергоспоживання (до 10 років роботи від звичайних батарей AA), невисока швидкість обміну даними, велика дальність зв'язку (15 км в сільській місцевості і 5 км в щільній міській забудові) та низька вартість кінцевого обладнання.

Протокол LoRaWAN оптимізований для кінцевих пристроїв, що працюють від батарей і включає в себе різні класи вузлів, забезпечуючи компроміс між швидкістю доставки інформації і часом роботи пристроїв при використанні живлення від батарей (акумуляторів). Протокол забезпечує повний двосторонній зв'язок, а архітектура, за допомогою спеціальних методів шифрування, забезпечує загальну надійність і безпеку всієї системи. Архітектура LoRaWAN розроблялася з урахуванням можливості активної роботи з мобільними кінцевими пристроями (end-node), що є одним з швидко зростаючих напрямків «Інтернету речей».

У типовій LoRaWAN мережі шлюзи передають зашифровані дані, отримані від кінцевих пристроїв (end-node) на центральний сервер мережі провайдера (Network Server) і далі на сервер додатків (App Server) сервіс-провайдера, з якого дані надходять кінцевому користувачеві (рис. 3.19).

У LoRaWAN мережах, на відміну наприклад, від Ethernet, шлюзи (від англ. Gateway) також називають концентраторами. Кінцеві пристрої називають кінцевими вузлами (end-node) або просто вузлами (node) або точками (point). Мережевий стек LoRaWAN іноді ще називають LoRaMAC, щоб підкреслити, що він працює на другому рівні моделі OSI media layer 2 - MAC (від англ. Media access control). Саме так - LoRaMAC, називався стандарт LoRaWAN, коли він розроблявся тільки командою Semtech до створення LoRa Alliance.

На даний момент існують дві open source реалізації стека LoRaWAN: перша розроблена Semtech - LoRaMAC, а друга IBM (LoRaWAN in C). Обидві реалізації використовують універсальний HAL драйвер пристроїв, цей відкритий LoRaWAN код портується для застосування в різному устаткуванні.

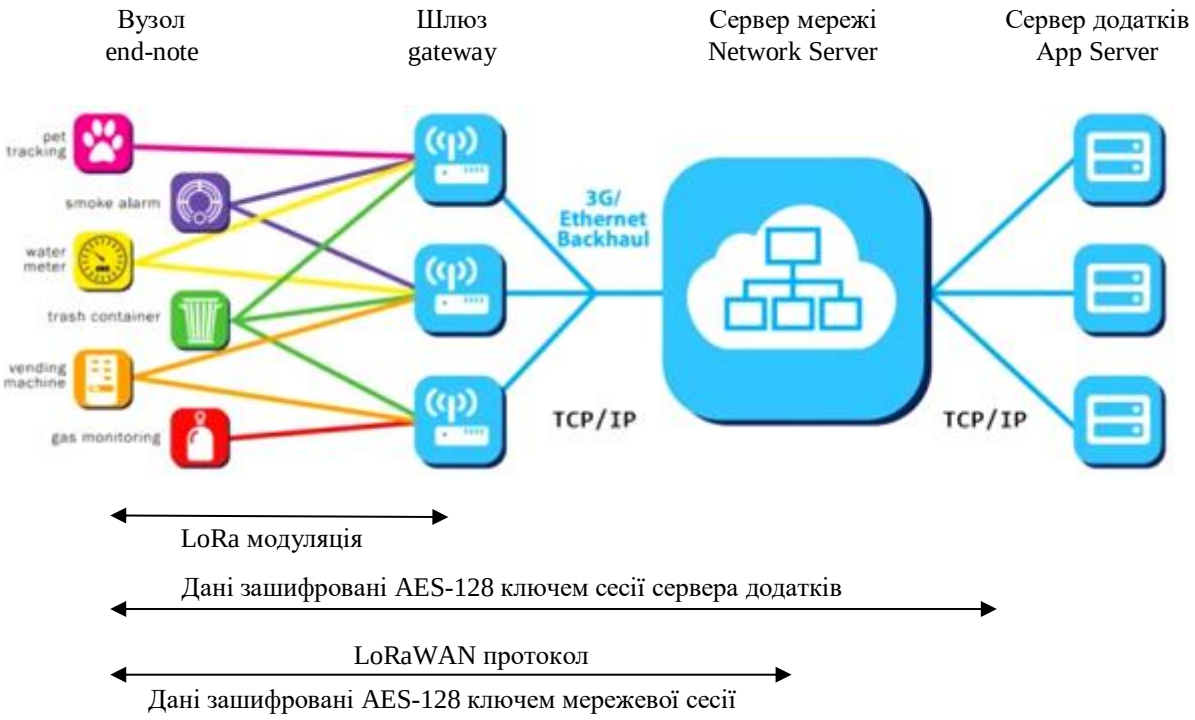


Рис. 3.19. Формування LoRaWAN мереж

Архітектура LoRaWAN мереж

LoRaWAN end-node - кінцеві пристрої

Кінцеві пристрої LoRa (кінцеві вузли, end-node) є елементами LoRaWAN мережі системи LoRa, де вони виконують такі функції, як вимір або управління і контроль. Вони розташовуються віддалено і мають, як правило, батарейне живлення. Використовуючи мережевий протокол LoRaWAN, ці кінцеві точки (end-node) можуть бути налаштовані для зв'язку з шлюзом LoRa (концентратором або базовою станцією).

Дані в LoRaWAN мережі можуть передаватися в обидві сторони, як від кінцевих точок (end-node) до сервера, так і назад. Точки (end-node) передають дані не постійно, а включають передачу лише на деякий проміжок часу (як правило на 1-5 секунд), після закінчення якого відкривається два часових вікна для прийому даних. Решту часу трансивери кінцевих вузлів (end-node) знаходяться або в неактивному стані (sleep), або в стані прийому, в залежності від класу пристрою (A, B або C).

Клас A

Вузол (end-node) передає дані на шлюз короткими посилками за заданим графіком. Ініціатором обміну виступає сам кінцевий вузол (end-node). Точка (end-node), як правило, не вимагає отримання підтвердження свого повідомлення додатком (повідомлення без квитанцій), однак протокол передбачає і повідомлення, на які сервер додатків формує спеціальну відповідь, "квитанцію", а мережевий сервер вибирає найкращий маршрут (шлюз) для відправки підтвердження (ACK від англ. acknowledgment - підтвердження) в момент відкриття вузлом вікна прийому (повідомлення з квитанціями). Вузол (end-node) переходить в режим прийому (відкриває вікно прийому) відразу після відправки даних на деякий нетривалий час, в іншому випадку, більш тривалий час, знаходиться в режимі енергозбереження або сну (sleep). Сервер накопичує для точок (end-node) повідомлення і пересилає їх відразу, як точка (end-node) виходить на зв'язок. Цей клас кінцевих (end-node) вузлів найбільш економічний у використанні енергії та найбільш поширений на практиці.

Клас B

Вузол (end-node) включає приймач за графіком, заданому сервером. Сервер відправляє повідомлення вузлу (end-node) відповідно до розкладу. Ініціатором обміну може бути і сервер LoRaWAN мережі. Пристрої (end-node) цього класу синхронізують внутрішній час з часом мережі за допомогою маяків (від англ. Beacon), які пристрій регулярно отримує від шлюзу. Вузли (end-node) цього класу мають відносно низьку часову затримку в обміні даними і відкривають більш широке часове вікно прийому, в порівнянні з класом A. Точки (end-node) класу B також мають всі можливості пристроїв (end-node) класу A .

Клас C

У точок (end-node) цього класу вікно прийому відкрито постійно і закривається тільки на період короткочасної передачі даних. Сервер може ініціювати обмін в будь-який час, і передати повідомлення вузлу (end-node) відразу, у міру їх появи. Цей клас пристроїв (end-node) споживає найбільшу кількість енергії (в порівнянні з класами A і B), тому зазвичай не використовує батарейне живлення, але отримує дані

від сервера LoRaWAN мережі з найменшими затримками (lowest latency). Пристрої класу C (end-node) мають всі можливості пристроїв класу A і B.

Точки (end-node) можуть проводити обмін як з одним, так і з декількома шлюзами, вузли можуть працювати в двох режимах: точка-точка (від англ. P2P - point to point), коли обмін відбувається між кінцевим пристроєм (end-node) і шлюзом (також цілком реалізуємий обмін тільки між двома вузлами (end-node) без використання концентраторів і навіть сервера) і в гібридному режимі, коли один з вузлів підключений, з одного боку, по радіоканалу до інших вузлів, а з іншого боку, має дротове підключення до мережі по TCP/IP і виступає в ролі шлюзу (використовуючи програмне забезпечення packet_forwarder). Такий одноканальний міні-шлюз може обслуговувати від одного до декількох десятків кінцевих пристроїв, які будуть конкурувати між собою за вільні тайм слоти (від англ. Time slot) міні-шлюзу для прийому і передачі даних.

Шлюзи (концентратори) LoRa

Шлюзи LoRa призначені для використання в радіальних зіркоподібних мережевих архітектурах великого радіусу дії в системі LoRaWAN. Через властивості технології LoRa ці шлюзи можуть являти собою багатоканальні мультимодемні трансивери, які здатні виконувати демодуляцію відразу декількох каналів одночасно, і навіть одночасну демодуляцію множини сигналів на одному і тому ж каналі. Ці шлюзи використовують інші радіочастотні компоненти, ніж ті, які застосовуються в кінцевій точці (end-node) для забезпечення високої ємності мережі. Шлюзи служать в якості інтерфейсу у вигляді прозорого моста для передачі повідомлень між кінцевими вузлами (end-node) і центральним сервером.

Зв'язок між концентраторами і центральним сервером LoRaWAN мережі оператора (транспортна backhaul мережа) здійснюється за допомогою традиційних технологій (Ethernet, WiFi, GSM) по протоколу TCP/IP.

Якщо шлюзи підключаються до мережевого сервера через стандартні IP-з'єднання, то кінцеві вузли (end-node) використовують бездротове підключення до одного або кількох шлюзів. Всі кінцеві точки (end-node), як правило, є двонаправленими, але вони також підтримують і функціонування в режимі, що забезпечує можливість здійснення групового оновлення програмного забезпечення через стільникову мережу або передачу інших масових повідомлень (Broadcast), що дозволяє скоротити час на їх передачу. Залежно від бажаної їх канальної ємності і місць установки доступні різні версії

шлюзів, вони можуть встановлюватися як всередині приміщень (indoor), так і на вишках або будівлях (outdoor).

Вузли (end-node) LoRaWAN мережі можуть бути в зоні покриття як одного шлюзу так і декількох. Шлюзом в мережах з високою щільністю абонентських пристроїв виступають спеціальні багатоканальні концентратори, які мають можливість приймати дані від декількох вузлів одночасно. Саме ця можливість шлюзу безпосередньо впливає на максимальну щільність абонентських пристроїв (end-node) на ділянці місцевості, яка обслуговується одним концентратором.

Концентратори на базі Semtech SX1301 мають можливість обслуговувати до 5 тисяч абонентських пристроїв на один квадратний кілометр (на борту 2 чіпа SX1257, що забезпечують подвійний RF фронтенд (від англ. Front-end) на 8 незалежних каналів, які можуть працювати одночасно, і одним транспортним backhaul каналом).

Ємність мережі залежить від того числа пакетів, які можуть бути отримані в даний момент часу. Один шлюз на SX1301 з 8 каналами, використовуючи протокол LoRaWAN, здатний отримати близько 1,5 млн. пакетів в день. Так що, якщо ваш вузол відправляє один пакет на годину, то один шлюз на SX1301 може з успіхом обслуговувати до 62500 таких кінцевих пристроїв.

Зараз ведеться серйозна боротьба серед прихильників різних IoT технологій і в порівняльних таблицях, де кожен нахвалює сам себе, ви побачите різну кількість вузлів (end-node), що обслуговуються одним шлюзом: від декількох сотень до мільйонів. Такі дані неінформативні і можуть ввести читача в оману, оскільки кожен вузол (end-node) може відправляти дані з різною періодичністю, обсяг даних і швидкість передачі можуть істотно відрізнятися, тому говорити про теоретичну ємності мережі досить складно і для точних розрахунків потрібно брати до уваги безліч чинників.

Якщо ємності сегмента мережі недостатньо, то LoRaWAN мережа масштабується: більш високу щільність вузлів (end-node), що досягається шляхом установки додаткових шлюзів. При появі нового шлюзу, центральний сервер мережі перерозподіляє навантаження, відправляючи кінцевим вузлам "новий графік" включення режиму передачі.

Центральний сервер LoRaWAN мережі

Проблему можливих колізій при одночасній передачі даних декількома точками вирішує центральний сервер LoRaWAN мережі, який адресно відправляє вузлам (end-node) мережі керуючі команди через шлюзи, виділяючи тайм-слоти для передачі і прийому індивідуально для кожної кінцевої точки (end-node). Адресація

відбувається по 32-бітному DevAddr, унікальному для кожного вузла (end-node).

Центральний сервер LoRaWAN мережі приймає рішення про необхідність зміни швидкості передачі даних точками (end-node), потужності передавача, виборі каналу передачі, її початку і тривалості за часом, контролює заряд батарей кінцевих вузлів (end-node), тобто повністю контролює всю мережу і управляє кожним абонентським пристроєм окремо.

Кожен LoRaWAN пакет даних, що відправляються кінцевим вузлом, (end-node) має в своєму складі унікальний ідентифікатор додатку AppEUI, що належить додаткам на сервері сервіс-провайдера, для якого він призначений і цей ідентифікатор використовується центральним сервером LoRaWAN мережі для подальшої маршрутизації пакета і його обробки додатком на сервері (App Server) сервіс-провайдера.

На практиці, як правило, послуги сервіс-провайдера надає виробник кінцевих пристроїв (end-node), який підтримує сервіс для обробки даних, куди маршрутизуються пакети з сервера LoRaWAN мережі для роботи.

Як окремий випадок, сервер додатків, сервер мережі і єдиний шлюз мережі (у вигляді одноканального LoRa трансивера) можуть бути об'єднані для побудови спрощеної моделі мережі в лабораторних умовах. Програмне забезпечення ядра мережевого сервера вільно не поширюється, але може бути отримано після укладення угоди з Lora Alliance, в загальному доступі знаходиться демонстраційна документація.

Стійкість до радіоперешкод

Висока проникаюча здатність радіосигналу субгігагерцового діапазону в будівлях і підвалах забезпечує стабільний зв'язок там, де інші бездротові технології виявляються безсилі.

Модем LoRa на суміщеному GMSK каналі має можливість придушення перешкод до 19,5 дБ (за рахунок гауссової фільтрації) або, кажучи іншими словами, він може приймати і демодулювати сигнали на 19,5 дБ нижче рівня перешкод або шумів при тому, що для правильної демодуляції більшості систем з частотною маніпуляцією FSK (від англ. Frequency Shift Keying) потрібна потужність сигналу, як мінімум на 8-10 дБ вище рівня шуму.

Цей імунітет до перешкод дозволяє використовувати просту і недорогу систему з LoRa в тих місцях, де є важка спектральна обстановка (як в будь-якому сучасному мегаполісі) або в гібридних мережах зв'язку. У цих випадках використання технології LoRa дозволяє розширити діапазон покриття мережі зв'язку.

Швидкість в LoRaWAN мережах

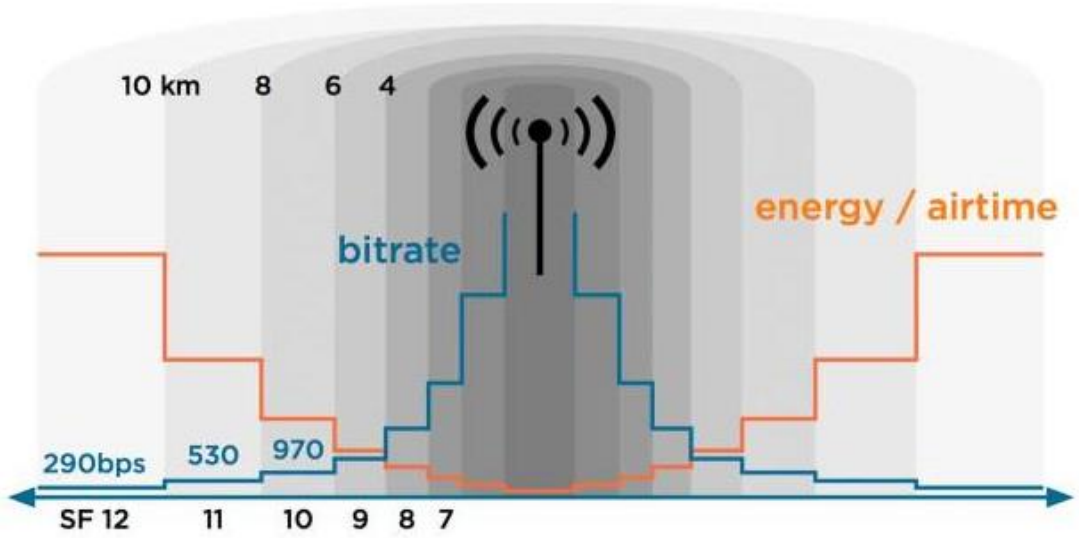


Рис. 3.20. Адаптивна швидкість передачі даних в LoRaWAN мережах

Адаптивна швидкість передачі даних ADR (Adaptive Data Rate) в LoRaWAN мережах (рис.3.20).

LoRaWAN протокол регламентує швидкість радіообміну від 300 біт/с до 50 кілобіт в секунду, швидкість падає зі збільшенням відстані між приймачем і передавачем. Фактично в існуючих пристроях, швидкість, може не перевищувати 11 кілобіт в секунду, що цілком достатньо для розв'язуваних даною технологією завдань.

Для Європи доступний один GFSK канал (від англ. Gaussian Frequency-Shift Keying - модуляція у вигляді частотної маніпуляції, при якій використовується фільтр Гаусса для згладжування) передачі інформації з потоком даних до 50 кбіт/с. У Північній Америці, через обмеження, що накладаються FCC (Federal Communications Commission - Федеральна Комісія Електрозв'язку США), мінімальна швидкість передачі даних становить 0,9 кбіт/с.

Щоб продовжити термін служби батареї (акумулятора) в кінцевому вузлі (end-node) і оптимізувати загальну пропускну здатність мережі, мережевий сервер LoRaWAN управляє швидкістю передачі даних і потужністю радіочастотного виходу кожного кінцевого пристрою (end-node) окремо на підставі відстані від шлюзу. Управління здійснюється за допомогою алгоритму адаптивної швидкості передачі даних ADR (від англ. Adaptive Data Rate). Це має вирішальне значення для високої продуктивності мережі і дозволяє здійснювати її необхідну масштабованість.

Адаптивна швидкість передачі даних ADR (Adaptive Data Rate) представляє собою метод, при якому фактична швидкість передачі даних регулюється таким чином, щоб забезпечити надійну доставку

пакетів, забезпечити оптимальну продуктивність мережі і необхідний масштаб для її завантаження. Так, наприклад, вузли (end-node) ближчі до шлюзу використовуватимуть і більш високу швидкість передачі даних (а, отже, більш короткий час активної передачі по радіоканалу) і меншу вихідну потужність. Тільки найвіддаленіші точки (end-node) будуть використовувати низьку швидкість передачі даних і високу вихідну потужність передавача. Технологія адаптивної швидкості передачі даних ADR може внести необхідні зміни в мережеву інфраструктуру і, таким чином, компенсувати різні втрати на трасі передачі сигналу.

LoRaWAN мережа може бути розгорнута з мінімальними інвестиціями в інфраструктуру і з тією її ємністю, яка конкретно потрібна для даного застосування. Якщо розгорнуто багато шлюзів, то технологія ADR буде зміщувати швидкість передачі даних в сторону підвищення, що забезпечить масштабування ємності мережі в межах від 6 до 8 раз.

Протокол LoRaWAN визначає конкретний набір швидкостей передачі даних, але крайовий чіп або так званий фізичний шар (PHY, OSI media layer 1), тобто сама інтегральна схема, призначена для виконання функцій фізичного рівня мережевої моделі OSI, здатна дати більше варіантів. Наприклад, Semtech SX1272 підтримує швидкості передачі даних від 0,3 до 37,5 кбіт/с, а SX1276 від 0,018 до 37,5 кбіт/с.

Безпека в LoRaWAN мережах

Для захисту від несанкціонованого доступу і спотворення або перехоплення даних, переданих кінцевими пристроями (end-node), в LoRaWAN мережах стандарту передбачено обов'язкове дворівневе шифрування даних двома різними AES-128 ключами по RFC-4493.

Забезпечується повна конфіденційність даних при проходженні всіх задіяних в ланцюжку пристроїв, тому вміст пакета доступно тільки відправнику (кінцевій точці) і одержувачу, для якого воно призначене, тобто додатком сервіс-провайдера. Сервер мережі оперує даними в зашифрованому вигляді, виробляє аутентифікацію і перевіряє цілісність кожного пакету, але при цьому не має доступу до корисного навантаження (від англ. Payload), тобто до інформації від підключених до вузла сенсорів.

3.3.2. Sigfox

Sigfox – це вузькосмуговий протокол LPWAN (як NB-IOT), розроблений у 2009 р. у Тулузі, Франція. Установча компанія має однойменну назву. Це ще одна технологія LPWAN, яка використовує не

ліцензовані лінії ISM для пропрієтарного протоколу. Sigfox має деякі риси, які значно звужують його використання [1]:

до 140 повідомлень на пристрій щодня по висхідній лінії зв'язку (робочий цикл 1%, 6 повідомлень на годину);

розмір корисного навантаження становить 12 байтів для кожного повідомлення (висхідна лінія зв'язку) і 8 байтів (низхідна лінія зв'язку);

пропускна здатність до 100 біт/с для висхідної лінії зв'язку і 600 біт/с по низхідній лінії зв'язку.

Спочатку Sigfox був односпрямований та призначений як чиста сенсорна мережа. Це означає, що підтримується тільки зв'язок зі висхідною лінією датчика. З цього часу став доступний канал низхідної лінії зв'язку.

Sigfox – запатентована та закрита технологія. Хоча їхнє обладнання відкрите, однак у мережу не потрібно підписуватися. Партнери з обладнання Sigfox включають Atmel, TI, Silicon Labs та інші. Sigfox будує та керує своєю мережевою інфраструктурою, подібною до розташування несучих LTE. Це зовсім інша модель, відмінна від LoRaWAN. LoRaWAN вимагає використання власної мережі PHY у своїй мережі, у той час як Sigfox використовує кілька аппаратних виробників, але одну керовану мережеву інфраструктуру. Sigfox обчислює ставки за кількістю пристроїв, підключених до передплати мережі клієнта, профілю трафіку на пристрій та тривалості контракту.

Хоча існують суворі обмеження Sigfox з точки зору пропускної здатності та використання, він призначений для систем, що відправляють невеликі та нечасті сплески даних. Кандидатами можуть бути пристрої IoT, такі як системи сигналізації, прості вимірювачі потужності та датчики навколишнього середовища. Дані для різних датчиків зазвичай можуть вписуватися в обмеження (наприклад, дані про температуру/вологість 2 байти при точності 0,004 °). Потрібно бути обережним з тим ступенем точності, яку надає датчик, та кількістю даних, які можуть бути передані. Один трюк для використання – дані стану; стан чи подія можуть бути повідомленням без будь-якого корисного навантаження. І тут протокол споживає 0 байт. Хоча це не усуває обмежень у мовленні, це можна використовувати для оптимізації живлення.

Фізичний рівень Sigfox

Як згадувалося раніше, Sigfox – ультравузька смуга (UNB). Як випливає із назви, передача використовує дуже вузький канал для зв'язку. Замість того, щоб поширювати енергію по широкому каналу, вузьке використання цієї енергії обмежується такими смугами:

868 МГц - Європа (правила ETSI 300-200);

902 МГц - Північна Америка (правила 15 частини FCC).

Смуга шириною 100 Гц і використовує *спектр ортогонального поширення послідовності (OSSS)* для сигналу висхідної лінії зв'язку і 600 Гц з використанням *гаусової синхронізації зсувної частоти* (GFSK) для низхідної лінії зв'язку. Sigfox відправить короткий пакет на випадковий канал із випадковою часовою затримкою (від 500 до 525 мс). Цей тип кодування називається випадковою частотою та множинним доступом з часовим поділом (RFTDMA). Sigfox, як заявлено, має суворі параметри використання, зокрема, обмеження розміру даних.

У табл. 3.5 показані ці параметри для каналів висхідної та низхідної лінії зв'язку.

Таблиця 3.5. Параметри каналів висхідної та низхідної лінії зв'язку

	Висхідний зв'язок	Низхідний зв'язок
Межа корисного навантаження (байти)	12	8
Пропускна спроможність (біт/с)	100	600
Максимальна кількість повідомлень на день	140	4
Схема модуляції	DBPSK	GFSK
Чутливість (дБм)	< 14	< 27

Двонаправлений зв'язок є важливою характеристикою протоколу Sigfox та інших протоколів. Однак двонаправлений зв'язок у Sigfox вимагає деякого пояснення. Режим пасивного прийому відсутній, що означає, що базова станція не може просто надіслати повідомлення на пристрій кінцевої точки у будь-який час. Вікно прийому відкривається для зв'язку лише після завершення вікна передачі. Вікно прийому відкривається тільки після 20-секундного періоду, коли перше повідомлення надіслано вузлом кінцевої точки.

Вікно залишиться відкритим протягом 25 с, що дозволить отримати коротке (4 байти) повідомлення від базової станції.

У Sigfox використовуються 333 канали шириною 100 Гц. Чутливість приймача становить -120 дБм/-142 дБм. Частотний стрибок підтримується з використанням псевдовипадкового методу 3-х із 333 каналів. Зрештою, потужність передачі зазначена як +14 дБм та +22 дБм у Північній Америці (рис. 3.21).

Три копії корисного навантаження передаються на трьох унікальних випадкових частотах із різними часовими затримками. Вікно

передачі по низхідній лінії відкривається тільки після останньої висхідної лінії зв'язку

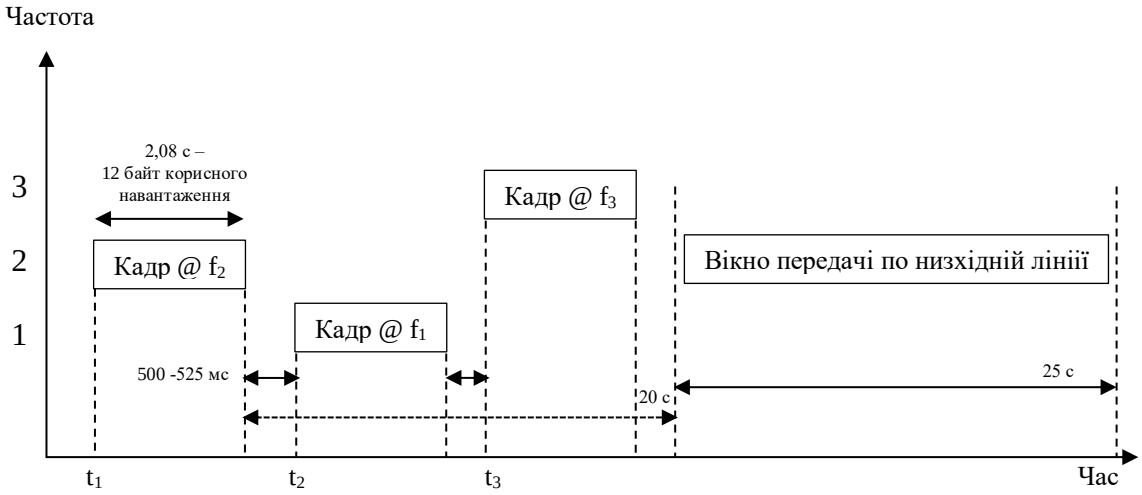


Рис. 3.21. Часова шкала передачі Sigfox.

Рівень MAC Sigfox

Кожен пристрій у мережі Sigfox має унікальний ідентифікатор Sigfox. Ідентифікатор використовується для маршрутизації та підписування повідомлень. Також він використовується для автентифікації Sigfox.

Ще одна характеристика зв'язку Sigfox полягає в тому, що він діє як «спалити та забути». Отримувачі не підтверджують повідомлення. Швидше, повідомлення відправляється тричі на трьох різних частотах у три різні напрямки вузлом. Це допомагає забезпечити цілісність передачі. Модель «спалити та забути» не має можливості гарантувати, що повідомлення дійсно отримане, тому передавач повинен робити якнайбільше, щоб забезпечити точну передачу (рис. 3.22).

Висхідний кадр Sigfox MAC

32 біти	16 біт	32 біти	від 0 до 96 біт	Варіюється	16 біт
Преамбула	Синхронізація кадру	ІД адресата	Корисне навантаження	Аунтентифікація	Контрольна сума

Низхідний кадр Sigfox MAC

32 біти	13 біт	2 біти	8 біт	16 біт	Варіюється	від 0 до 64 біт
Преамбула	Синхронізація кадру	Прапори	Контрольна сума	Аунтентифікація	Коди помилок	Корисне навантаження

Рис. 3.22. Структура пакету кадрів Sigfox MAC

Кадри містять преамбулу наперед визначених символів, що використовуються для синхронізації при передачі. Поля синхронізації кадру визначають типи кадрів, що передаються. *FCS – це контрольна послідовність кадрів (FCS)*, яка використовується для виявлення помилок.

Пакет не містить адресу призначення або інший вузол. Всі дані надсилатимуться різними шлюзами до хмарної служби Sigfox.

Межу даних можна зрозуміти та змоделювати з формату пакета рівня MAC, він складає с.

Оскільки кожен пакет передається три рази і ми знаємо, що європейські правила (ETSI) обмежують передачу в 1% робочого циклу, ми можемо розрахувати кількість повідомлень за годину, використовуючи максимальний розмір корисного навантаження 12 байт – 6 повідомлень за год.

Хоча 12 байт є межею корисного навантаження, це повідомлення може зайняти одну секунду для передачі. Ранні версії Sigfox були односпрямованими, але протокол тепер підтримує двонаправлений зв'язок.

Стек протоколу Sigfox

Стек протоколу аналогічний іншим стекам, наступним моделі OSI (табл. 3.6).

Таблиця 3.6. Стек протоколу Sigfox та спрощена модель OSI

Стек протоколу Sygfox	**Спрощена модель OSI**
Прикладний рівень	7. Прикладний рівень
	6. Представницький рівень
	5. Прикладний рівень
Frame	4. Транспортний рівень
	3. Мережевий рівень
MAC Layer	2. Канальний рівень
PHY Layer (868 МГц / 902 МГц Radios)	1. Фізичний рівень

Існує три рівні стеку:

рівень PHY – синтезує та модулює сигнали з використанням DBPSK у напрямку висхідної лінії зв'язку та GFSK у напрямку низхідної лінії зв'язку, як описано раніше;

рівень MAC – додає поля для ідентифікації/автентифікації пристрою (HMAC) та коду виправлення помилок (CRC). Sigfox MAC не

надає сигналізації. Це означає, що пристрої не синхронізуються із мережею;

рівень кадру – генерує радіокадр з даних програми. З іншого боку, систематично прикріплює порядковий номер до кадру.

Що стосується *безпеки*, повідомлення не шифруються в жодному місці стека протоколів Sigfox. Клієнт повинен надати схему шифрування даних корисного навантаження (рівень 7 або 6 моделі OSI). Ніякі ключі не обмінюються через мережу Sigfox, проте кожне повідомлення підписується ключем, унікальним для ідентифікації пристрою.

Топологія Sigfox

Мережа Sigfox може бути щільною як один мільйон вузлів на базову станцію (рис. 3.23). Щільність – це функція кількості повідомлень, надісланих мережевими пристроями. Усі вузли, які приєднуються до базової станції, утворюють зіркову мережу.

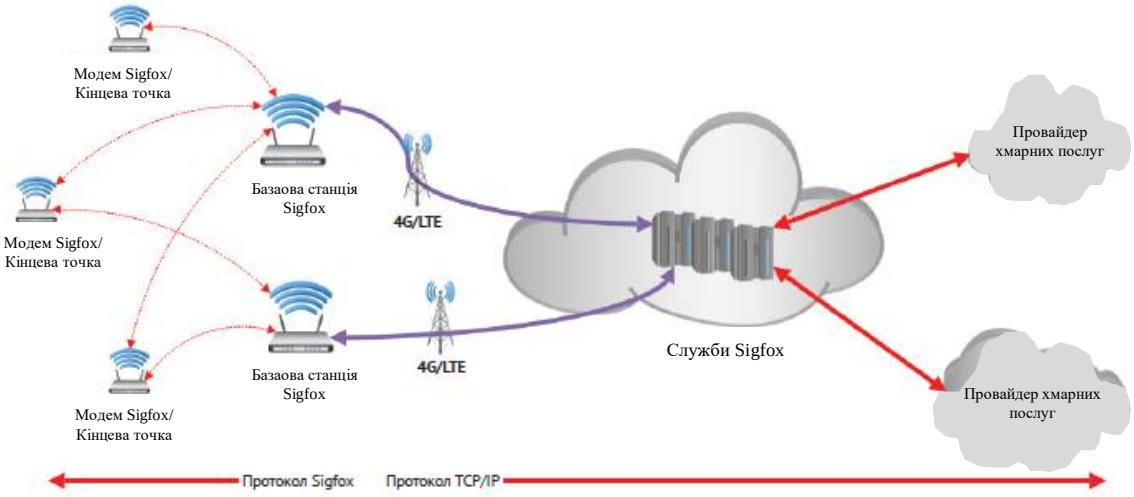


Рис. 3.23. Топологія Sigfox.

Sigfox використовує свій власний протокол, відмінний від IP, і об'єднує дані в мережевий сервер Sigfox як дані IP

Всі дані управляються через веб-мережу Sigfox. Всі повідомлення з базової станції Sigfox повинні надходити на внутрішній сервер через з'єднання IP.

Служба хмарних обчислень Sigfox є єдиним місцем призначення пакета. Бекенд зберігатиме і надсилатиме повідомлення клієнту після його аутентифікації та перевірки відсутності дублікатів. Якщо дані мають бути передані вузлу кінцевої точки, бекенд-сервер вибиратиме шлюз з найкращим підключенням до кінцевої точки та пересилати повідомлення по низхідній лінії. Бекенд вже ідентифікував пристрій за ідентифікатором пакета, і попереднє надання даних змусить дані відправляти до кінцевого пункту призначення. В архітектурі Sigfox неможливо отримати доступ до пристрою безпосередньо. Ні бекенд, ні

базова станція не будуть безпосередньо підключатимуться до кінцевого пристрою.

Бекенд – це також адміністрування, ліцензування та надання послуг для клієнтів. Хмара Sigfox передає дані до пункту призначення, обраного клієнтом. Хмарний сервіс пропонує API із можливістю вставки для інтеграції хмарних функцій Sigfox на сторонній платформі. Пристрої можуть бути зареєстровані через іншу службу хмар. Sigfox також пропонує зворотні дзвінки для інших хмарних сервісів. Це кращий метод отримання даних.

Щоб допомогти забезпечити цілісність даних у моделі зв'язку, що забувається, кілька шлюзів можуть отримувати передачу від вузла; всі наступні повідомлення будуть перенаправлені на бекенд Sigfox, і дублікати буде видалено. Це додає рівень надмірності прийому даних.

Прикріплення вузла кінцевої точки Sigfox призначене для полегшення встановлення.

3.4. IoT протоколи передачі даних від граничних пристроїв до хмари

Існує безліч складнощів і тонкощів у побудові та підключенні мереж PAN, не заснованих на IP, до мереж WAN на основі IP. Існують також перетворення протоколів, які потрібно розуміти. Стандартні протоколи – це інструменти, які пов'язують та інкапсулюють необроблені дані від датчика до чогось значущого та форматують їх для хмари. Одна з основних відмінностей між системою IoT і системою M2M полягає в тому, що M2M може зв'язуватися через WAN з виділеним сервером або системою без будь-якого інкапсульованого протоколу. За визначенням IoT заснований на зв'язку між кінцевими точками та службами, причому інтернет є загальною мережевою основою.

HTTP забезпечує значні послуги та можливості для інтернету вже понад 20 років, але він був розроблений та сконструйований для загального використання у моделях клієнт/сервер. Пристрої IoT можуть бути дуже обмеженими, віддаленими та обмеженими по смузі пропускання. Тому для управління множиною пристроїв у різних мережевих топологіях, таких як mesh-мережі, необхідні більш ефективні, безпечні та масштабовані протоколи.

При передачі даних в Інтернет технологія переноситься на фундаментальні рівні TCP/IP. Протоколи TCP і UDP є очевидним і єдиним вибором передачі даних, TCP значно складніше у реалізації, ніж UDP (що є протоколом багатоадресної розсилки). Однак UDP не має

стабільності та надійності TCP, змушуючи в деяких розробках компенсувати це додаванням відмовостійкості на рівнях додатків вище UDP.

В IoT досить часто використовується ***зв'язуюче програмне забезпечення, орієнтоване на обробку повідомлень*** [1] (**message-oriented middleware, MOM**) - підпрограмне забезпечення, орієнтоване на обмін повідомленнями в розподіленому оточенні [2]. Насамперед воно призначено для реалізації відкладеного обміну повідомленнями, тоді як однорангова мережа (peer-to-peer) та віддалені виклики процедур (RPC) підтримують синхронний режим. Основна ідея MOM полягає в тому, що зв'язок між двома пристроями відбувається з використанням розподілених черг повідомлень. MOM відправляє повідомлення від однієї програми в просторі користувача іншому. Деякі пристрої виробляють дані для додавання до черги, тоді як інші споживають дані, що у черзі. Деякі реалізації вимагають, щоб центральним сервісом був брокер чи посередник. У цьому випадку виробники та споживачі мають публіцистичні та підписні зв'язки з брокером. AMQP, MQTT та STOMP є реалізаціями MOM; інші включають служби обміну повідомленнями CORBA та Java. Реалізація MOM із використанням черг може використовувати їх для стійкості у розробці. Дані можуть зберігатися в чергах, навіть якщо сервер відмовить.

Альтернативою реалізації MOM є **RESTful**. У моделі **RESTful** сервер має стан ресурсу, але стан не передається у повідомленні від клієнта на сервер. RESTful використовує HTTP-методи, такі як GET, PUT, POST та DELETE для розміщення запитів щодо універсального ідентифікатора ресурсу (URI) (див. рис. 3.24). Ця архітектура не потребує брокера або посередника. Оскільки вони засновані на стеку HTTP, вони користуються більшістю пропонованих сервісів, таких як безпека HTTPS. Проекти RESTful типові для клієнт-серверних архітектур. Клієнти ініціюють доступ до ресурсів через синхронні шаблони запиту-відповіді.

Крім того, клієнти відповідають за помилки, навіть якщо сервер падає. На рис. 3.24 показано MOM порівняно з сервісом RESTful. Ліворуч знаходиться служба обміну повідомленнями (на основі MQTT), що використовує нний брокерський сервер, видавців та передплатників подій. Тут багато клієнтів можуть бути як видавцями, так і передплатниками, і інформація може зберігатися або не зберігатися у черзі для швидкого відновлення. Праворуч знаходиться проект RESTful, де архітектура побудована на HTTP та використовує HTTP-парадигми для зв'язку від клієнта до сервера.

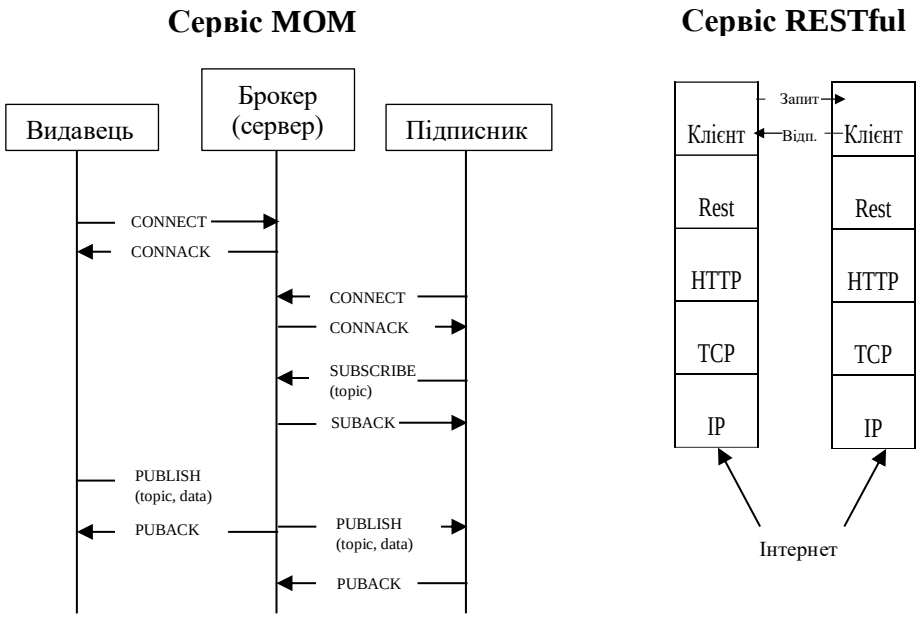


Рис. 3.24. Приклад для порівняння MOM з реалізацією RESTful

3.4.1. Протокол MQTT

MQTT – це MQ Telemetry Transport, який є простим і легким протоколом обміну повідомленнями, призначений для обмежених пристроїв і мереж з низькою пропускною здатністю, з високою затримкою або ненадійністю. Він розроблений на принципах мінімізації пропускної спроможності мережі та вимог до ресурсів пристроїв, намагаючись у той же час забезпечити надійність та певний ступінь впевненості у доставці. Ці принципи також роблять цей протокол ідеальним для появи світу підключених пристроїв типу "машина-машина" (M2M) або "інтернет речей", а також для мобільних додатків, де вкрай важливі пропускна здатність та заряд батареї [266].

Основні особливості протоколу MQTT:

- Асинхронний протокол, що має компактні повідомлення.
- Робота в умовах нестабільного зв'язку на лінії передачі даних.
- Підтримка кількох рівнів якості обслуговування (QoS).
- Легка інтеграція нових пристроїв та адміністрування.
- Працює на прикладному рівні поверх TCP/IP і використовує за замовчуванням порт 1883 (8883 при підключенні через SSL).
- Простий у використанні. Протокол є програмним блоком без зайвої функціональності, що може бути легко вбудований в будь-яку складну систему.

- Зручний для більшості рішень з датчиками. Дає можливість пристроям виходити на зв'язок і публікувати повідомлення, які не були заздалегідь відомі або визначені.
- Низьке навантаження на канал зв'язку.
- Робота в умовах постійної втрати зв'язку або інших проблем на лінії.
- Немає обмежень на формат переданого контенту.

Обмін повідомленнями в протоколі MQTT здійснюється між клієнтом (client), який може бути видавцем або підписником (publisher/subscriber) повідомлень, та брокером (broker) повідомлень (наприклад, Mosquitto MQTT).

Видавець надсилає дані на MQTT-брокер, вказуючи у повідомленні певну тему, топік (topic). Підписники можуть отримувати різні дані від багатьох видавців залежно від підписки відповідних топиків.

Пристрої MQTT використовують певні типи повідомлень для взаємодії з брокером, наведені нижче основні:

Connect – встановити з'єднання з брокером

Disconnect – розірвати з'єднання з брокером

Publish - опублікувати дані в топік на брокері

Subscribe – підписатися на топік на брокері

Unsubscribe - відписатися від топіка

Схема простої взаємодії між передплатником, видавцем та брокером подана на рис. 3.25

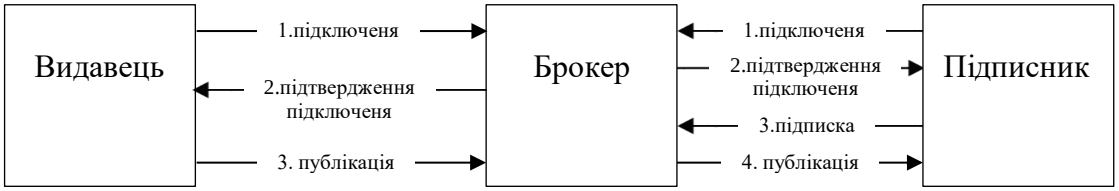


Рис. 3.25. Схема простої взаємодії між передплатником, видавцем та брокером

Брокер MQTT відповідає за з'єднання клієнтів і фільтрацію даних. Такі фільтри забезпечують:

фільтрацію за темами – за задумом, клієнти підписуються на теми та певні гілки тем і не отримують даних більше, ніж хочуть. Кожне опубліковане повідомлення має містити тему і брокер несе відповідальність за повторну передачу цього повідомлення передплатникам або його ігнорування;

фільтрацію за вмістом – брокери мають можливість перевіряти та фільтрувати опубліковані дані. Таким чином, будь-які дані, які не зашифровані, можуть керуватися брокером до того, як зберегти їх чи передати іншим клієнтам;

фільтрація за типом – клієнт, прослуховуючий потік даних, куди він підписаний, може також використовувати свої власні фільтри. Вхідні дані можуть аналізуватися і залежно від цього потік даних обробляється далі або ігнорується.

Структура повідомлень

MQTT повідомлення складається з кількох частин:

Фіксований заголовок (присутній у всіх повідомленнях)

Змінний заголовок (є тільки у певних повідомленнях)

Дані, «навантаження» (є тільки у певних повідомленнях)

На рис. 3.26 показаний фіксований заголовок

Рис. 3.26. Фіксований заголовок повідомлення MQTT

Message Type – це тип повідомлення, наприклад: CONNECT, SUBSCRIBE, PUBLISH та інші.

Flags specific to each MQTT packet – ці 4 біти відведені під допоміжні прапори, наявність та стан яких залежить від типу повідомлення.

Remaining Length - представляє довжину поточного повідомлення (змінний заголовок + дані), може займати від 1 до 4 байти.

Загалом у протоколі MQTT існує 15 типів повідомлень поданих у таблиці 3.7.

Таблиця 3.7. Типи повідомлень у протоколі MQTT

Тип повідомлення	Значення	Напрямок передачі	Опис
Reserved	0000 (0)	немає	Зарезервовано
CONNECT	0001 (1)	К* -> С**	Запит клієнта на підключення до сервера
CONNACK	0010 (2)	К <- С	Підтвердження успішного підключення
PUBLISH	0011 (3)	К <- С, К -	Публікація повідомлення

		> C	
PUBACK	0100 (4)	К <- С, К -> С	Підтвердження публікації
PUBREC	0101 (5)	К <- С, К -> С	Публікацію отримано
PUBREL	0110 (6)	К <- С, К -> С	Дозвіл на видалення повідомлення
PUBCOMP	0111 (7)	К <- С, К -> С	Публікацію завершено
SUBSCRIBE	1000 (8)	К -> С	Запит на підписку
SUBACK	1001 (9)	К <- С	Запит на підписку прийнято
UNSUBSCRIBE	1010 (10)	К -> С	Запит на відписку
UNSUBACK	1011 (11)	К <- С	Запит на відписку прийнято
PINGREQ	1100 (12)	К -> С	PING запит
PINGRESP	1101 (13)	К <- С	PING відповідь
DISCONNECT	1110 (14)	К -> С	Повідомлення про відключення від сервера
Reserved	1111 (15)		Зарезервований

*К – клієнт, **С – сервер

Чотири старші біти першого байта фіксованого заголовка відведені під спеціальні прапори (рис. 3.27)

Рис. 3.27. Прапори протокола MQTT

DUP – прапор дубліката встановлюється, коли клієнт або MQTT брокер здійснює повторне відправлення пакета (використовується у типах PUBLISH, SUBSCRIBE, UNSUBSCRIBE, PUBREL). У разі встановлення прапора змінний заголовок повинен містити Message ID (ідентифікатор повідомлення)

QoS – якість обслуговування (0,1,2)

RETAIN – при публікації даних із встановленим прапором retain, брокер збереже його. При наступній підписці на цей топік брокер негайно надішле повідомлення з цим прапором. Використовується лише у повідомленнях із типом PUBLISH.

У деяких заголовках може міститися *змінний* заголовок.

У ньому містяться такі дані:

- Packet identifier – ідентифікатор пакета, присутній у всіх типах повідомлень, крім: CONNECT, CONNACK, PUBLISH (з QoS <1), PINGREQ, PINGRESP, DISCONNECT.

- Protocol name – назва протоколу (тільки у повідомленнях типу CONNECT).

- Protocol version – версія протоколу (тільки у повідомленнях типу CONNECT).

- Connect flags – прапори, що вказують на поведінку клієнта при підключенні (рис. 3.28).

Bit	7	6	5	4	3	2	1	0
Byte 8	User name	Password	Will Retain	Will QoS		Will Flag	Clean Session	Reserved

Рис. 3.28. Прапори змінного заголовка повідомлення MQTT

User name – за наявності цього прапора у «навантаженні» має бути вказано ім'я користувача (використовується для автентифікації клієнта)

Password – за наявності цього прапора у «навантаженні» має бути вказаний пароль (використовується для автентифікації клієнта)

Will Retain - при установці в 1, брокер зберігає у себе Will Message.

Will QoS – якість обслуговування для Will Message, при встановленому прапорі Will Flag, Will QoS та Will retain є обов'язковими.

Will Flag - при встановленому прапорі після того, як клієнт відключиться від брокера без відправки команди DISCONNECT (у випадках непередбачуваного обриву зв'язку і т.д.), брокер сповістить про це всіх підключених до нього клієнтів через так званий Will Message.

Clean Session – очистити сесію. При встановленому "0" брокер збереже сесію, всі підписки клієнта, а також передасть йому всі повідомлення з QoS1 і QoS2, які були отримані брокером під час відключення клієнта, при його наступному підключенні. Відповідно при встановленій «1», при повторному підключенні клієнту необхідно буде заново підписуватися на топики.

- Session Present – використовується у повідомленні з типом CONNACK. Якщо брокер приймає підключення з Clean Session = 1, він повинен встановити «0» у біт Session Present(SP). Якщо брокер приймає підключення з Clean Session = 0, то значення біта SP залежить від цього, чи зберігав брокер раніше сесію з цим клієнтом (якщо так, то SP

виставляється 1 і навпаки). Тобто цей параметр дозволяє клієнту визначити, чи була збережена брокером попередня сесія.

- Connect Return code - якщо брокер з якихось причин не може прийняти правильно сформований CONNECT пакет від клієнта, то в другому байті CONNACK пакета він повинен встановити відповідне значення з нижченаведеного списку у таблиці 3.8:

Таблиця 3.8. Можливі варіанти відповіді в CONNACK

Значення	Повернене значення	Опис
0	0x00 Connection Accepted	Підключення прийнято
1	0x01 Connection Refused, unacceptable protocol version	Брокер не підтримує версію протоколу клієнта
2	0x02 Connection Refused, identifier rejected	Client ID клієнта, що підключається, немає в списку дозволених
3	0x03 Connection Refused, Server unavailable	З'єднання встановлено, але MQTT сервіс не доступний
4	0x04 Connection Refused, bad user name or password	Неправильний логін або пароль
5	0x05 Connection Refused, no authorized	Доступ до підключення заборонено
6-255		зарезервовано

- Topic Name – назва топіка.

Дані, навантаження

Зміст та формат даних, що передаються в MQTT повідомленнях, визначаються у додатку. Розмір даних може бути обчислений шляхом віднімання Remaining Length довжини змінного заголовка.

Якість обслуговування у протоколі MQTT (QoS)

MQTT підтримує три рівні якості обслуговування (QoS) під час надсилання повідомлень.

QoS-0 At most once. На цьому рівні видавець один раз відправляє повідомлення брокеру і не чекає на підтвердження від нього, тобто відправив і забув.

QoS-1 At least once. Цей рівень гарантує, що повідомлення точно буде доставлено брокеру, але є можливість дублювання повідомлень від видавця. Після отримання дубліката повідомлення, брокер знову надсилає це повідомлення передплатникам, а видавцеві знову надсилає підтвердження про отримання повідомлення. Якщо видавець не отримав повідомлення PUBACK від брокера, він повторно відправляє цей пакет, при цьому в DUP встановлюється «1».

QoS-2 Exactly once. На цьому рівні гарантується доставка повідомлень передплатнику та виключається можливе дублювання надісланих повідомлень.

Видавець надсилає повідомлення брокеру. У цьому повідомленні вказується унікальний пакет ID, QoS=2 і DUP=0. Видавець зберігає повідомлення непідтвердженим, поки не отримає від брокера відповідь PUBREC. Брокер відповідає повідомленням PUBREC, в якому міститься той же Packet ID. Після отримання видавець відправляє PUBREL з тим же Packet ID. До того, як брокер отримає PUBREL, він повинен зберігати копію повідомлення у себе. Після отримання PUBREL він видаляє копію повідомлення та надсилає видавцеві повідомлення PUBCOMP про те, що транзакцію завершено.

Захист передачі даних

Для забезпечення безпеки в MQTT протоколі реалізовано такі методи захисту:

Аутентифікація клієнтів. Пакет CONNECT може містити поля USERNAME і PASSWORD. При реалізації брокера можна використовувати ці поля для автентифікації клієнта.

Контроль доступу клієнтів через Client ID.

Підключення до брокера через TLS/SSL.

3.4.2. Протокол MQTT-SN

Спеціалізована версія для мереж датчиків MQTT називається MQTT-SN (іноді звучить як MQTT-S). Вона дотримується тієї ж філософії MQTT, як і легкий протокол для периферійних пристроїв, але сконструйована спеціально для нюансів бездротової локальної мережі, що є типовою для сенсорних середовищ. Це означає підтримку каналів з низькою пропускною спроможністю, відстеження відмови каналу, короткі повідомлення та апаратне забезпечення з обмеженими ресурсами. MQTT-SN, по суті, настільки прозорий, що може успішно працювати поверх BLE та Zigbee [1].

MQTT-SN не вимагає стек TCP/IP. Його можна використовувати за послідовним з'єднанням (переважний варіант), де простий протокол зв'язку (щоб розрізняти різні пристрої лінії) і накладні витрати дуже малі. Як альтернатива він може використовуватися протокол UDP, який вимагає менше ресурсів, ніж TCP.

Архітектура та топологія MQTT-SN

У топології MQTT-SN є чотири основні компоненти (рис. 3.29):

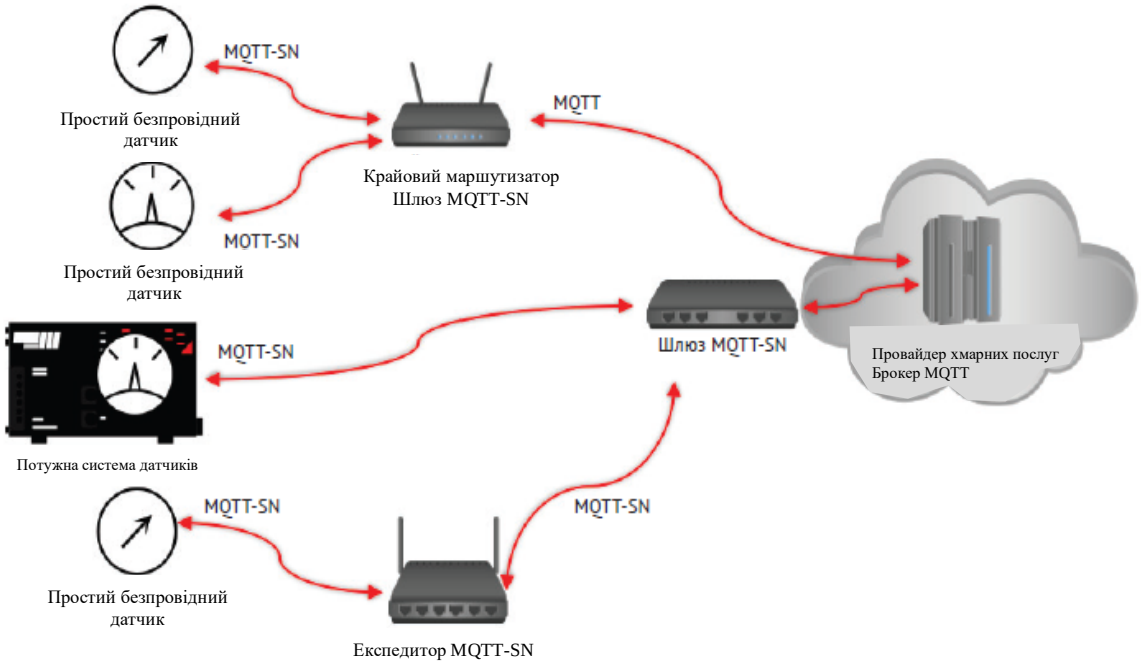


Рис. 3.29. Топологія MQTT-SN

шлюзи – в MQTT-SN шлюз несе відповідальність за перетворення протоколу з MQTT-SN в MQTT і навпаки (хоча можливі й інші переклади). Шлюзи також можуть бути агрегуючими або прозорими;

форвардери – маршрут між датчиком і шлюзом MQTT-SN може проходити по безлічі шляхів і перетинати кілька маршрутизаторів. Вузли між вихідним клієнтом та шлюзом MQTT-SN називаються пересилками, вони просто повторно інкапсулюють кадри MQTT-SN у нові та незмінні кадри MQTT-SN, які відправляються до пункту призначення до тих пір, поки вони не досягнуть правильного шлюзу MQTTSN для перетворення протоколу;

клієнти – клієнти поводяться як і, як і MQTT, можуть підписуватися і публікувати дані;

брокери – брокери поводяться так само, як у MQTT.

Бездротові датчики зв'язуються або зі шлюзами MQTT-SN, які переводять MQTT-SN в MQTT, в інші протокольні форми, або в форвардерів, які просто інкапсулюють кадри MQTT-SN, прийняті в MQTT-SN повідомлення, перенаправлені на шлюз.

Прозорі та збираючі шлюзи

У MQTT-SN шлюзи можуть грати дві різні ролі (рис. 3.30). По-перше, прозорий шлюз керуватиме багатьма незалежними потоками MQTT-SN з сенсорних пристроїв і перетворюватиме кожен потік на повідомлення MQTT.

Агрегаційний/збираючий шлюз об'єднає кілька потоків MQTT-SN у меншу кількість потоків MQTT, відправлених до хмарного брокера

MQTT. Агрегаційний шлюз складніше у створенні, але він зменшить кількість службових даних, що передаються, і кількість одночасних підключень, відкритих на сервері. Для створення топології шлюзу, що агрегує, клієнти повинні публікувати в тій же темі або підписуватися на одну і ту ж ж тему.

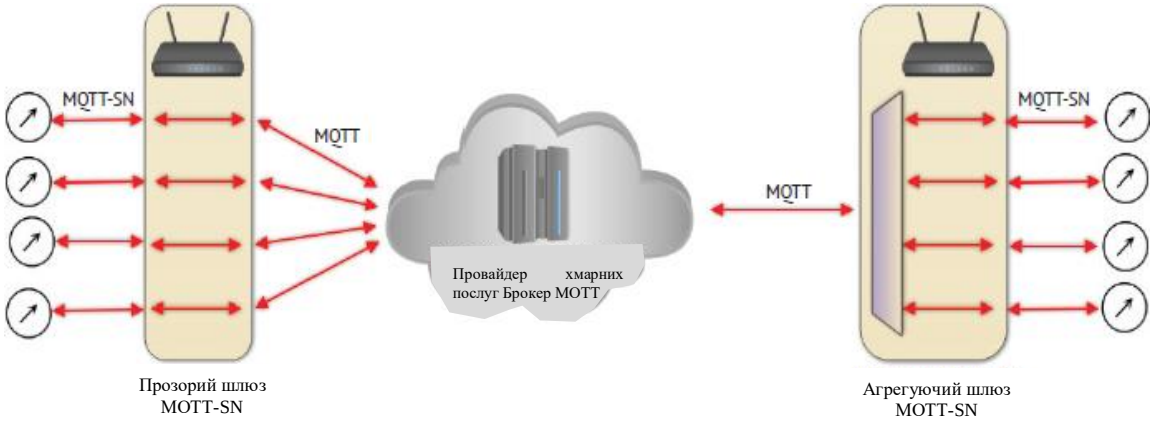


Рис. 3.30. Налаштування шлюзів MQTT-SN

Прозорий шлюз просто виконує перетворення протоколу кожного вхідного потоку MQTT-SN і має взаємно-однозначне відношення між з'єднаннями MQTT-SN і MQTT-з'єднаннями з брокером. Однак шлюз, що агрегує, об'єднає кілька потоків MQTT-SN в одне з'єднання MQTT з сервером.

Відмінності між MQTT та MQTT-SN

Основні відмінності між MQTT-SN та MQTT полягають у наступному:

- У MQTT-SN є три повідомлення CONNECT, а в MQTT – одне. Додаткові два використовуються для передачі теми "Will" та повідомлення "Will";
- MQTT-SN може працювати через спрощене середовище та UDP;
- Імена тем замінюються короткими двобайтовими повідомленнями з ідентифікатором теми. Це необхідно для кращого використання пропускної спроможності у бездротових мережах;
- попередньо визначені ідентифікатори тем і короткі імена тем можуть використовуватися без будь-якої реєстрації. Щоб використовувати цю функцію, клієнт і сервер повинні використовувати один і той самий ідентифікатор теми. Короткі назви досить короткі, щоб бути вбудованими у повідомлення PUBLISH;

- Запропоновано процедуру виявлення, що допомагає клієнтам і дозволяє їм знаходити мережеві адреси серверів і шлюзів. У мережі можуть існувати кілька шлюзів, які можна використовуватиме обміну повідомленнями з клієнтами;
- функція CleanSession отримує властивість Will. Клієнтська передплата може бути збережена, але тепер дані Will також зберігаються;
- У MQTT-SN використовується переглянута процедура keepAlive. Це робиться для підтримки сплячих клієнтів, у яких всі призначені для них повідомлення буферизуються сервером або прикордонним маршрутизатором і передаються під час пробудження.

3.4.3. Обмежений прикладний протокол CoAP

Обмежений прикладний протокол, або він же ***обмежений протокол додатків (CoAP)*** є розробкою IETF (RFC7228). Робоча група IETF Constreined RESTful Environments (CoRE) створила перший проект протоколу у червні 2014 р. Він спеціально призначений як протокол зв'язку для пристроїв з обмеженими можливостями. Основний протокол тепер заснований на RFC7252. Протокол унікальний, оскільки він вперше розроблений для обміну даними типу машина-машина (M2M) між граничними вузлами. Він також підтримує роботу з HTTP за допомогою проксі. Це перетворення HTTP є вбудованим засобом отримання даних через інтернет [1].

CoAP чудово забезпечує аналогічну та легку структуру адресації ресурсів, знайому всім, хто має досвід використання інтернету, але з обмеженими ресурсами та вимогами до пропускної спроможності.

CoAP забезпечує аналогічну функціональність із значно меншими витратами та потребами в потужності.

Крім того, деякі реалізації CoAP виконуються на x64 краще ніж еквіваленти HTTP на аналогічному устаткуванні (табл. 3.9).

Таблиця 3.9. Відмінності протоколів CoAP та HTTP

	Байти в транзакції	Споживана потужність	Час життя батареї
CoAP	154	0.744 mW	151 день
HTTP	1451	1.333 mW	84 дня

Деталі архітектури CoAP

Концепція CoAP заснована на імітації та заміні важких елементів HTTP та використання легкого еквівалента для IoT. Це не заміна HTTP, оскільки в ній відсутні деякі якості; HTTP вимагає потужніших і орієнтованих сервіси систем. Можливості CoAP можна підсумовувати так:

- схожа на HTTP;
- протоколи без встановлення з'єднання;
- безпека за допомогою DTLS, а не TLS, як при нормальній передачі HTTP;
- асинхронний обмін повідомленнями;
- легкий дизайн та низькі вимоги до ресурсів, низькі накладні витрати в плані заголовка;
- підтримка URI та типів вмісту;
- побудований на UDP замість TCP/UDP, як для звичайного HTTP-сеансу;
- HTTP-співставлення без збереження стану, що дозволяє проксисерверу підключатися до HTTP-сеансів.

CoAP має два основні рівні (рис. 3.31):

Стек HTTP	**Стек CoAp**
HTTP	CoAP (Запит/відповідь; Транзакція)
TCP	UDP
IP	IPv6
Канальний рівень	6LoWPAN
Фізичний рівень	IEEE 802.15.4

Рис. 3.31. Порівняння стеків HTTP та CoAP

рівень запиту/відповіді – відповідає за відправлення та отримання запитів на основі RESTful. Запити REST включені до повідомлень CON або NON. Відповідь REST включено у відповідне повідомлення ACK;

рівень транзакцій – на цьому рівні підтримується обмін повідомленнями між кінцевими точками, використовуючи один із чотирьох основних типів повідомлень. Рівень транзакції також підтримує керування багатоадресною розсилкою та перевантаженням.

Контекст, синтаксис та використання CoAP аналогічно HTTP. Адресація у CoAP також у стилі HTTP. Адреса – це структура URI. Як і URI в HTTP, користувач повинен знати адресу заздалегідь, щоб отримати доступ до ресурсу.

На верхньому рівні CoAP використовує такі запити, як GET, PUT, POST і DELETE, як у HTTP. Аналогічно, коди відповідей імітують HTTP, наприклад:

- 2.01 – створено;
- 2.02 – видалено;
- 2.04 – змінено;
- 2.05 – вміст;
- 4.04 – не знайдено (ресурс);
- 4.05 – метод не дозволено.

Типовий URI в CoAP має таку форму:

coap://хост[:порт]/[путь][?запрос]

Система CoAP складається із семи основних учасників:

кінцеві точки – це джерела та адресати повідомлення CoAP. Конкретне визначення кінцевої точки залежить від транспорту, що використовується;

проксі – кінцева точка CoAP, якій клієнти CoAP доручають виконувати запити від свого імені. Зниження навантаження на мережу, доступ до сплячих вузлів та забезпечення певного рівня безпеки – ось деякі функції проксі. Проксі можуть бути явно обрані клієнтом (пряме проксіювання) або можуть використовуватися як сервери insitu (зворотне проксіювання). Як альтернатива проксі може перетворити один запит CoAP на інший запит CoAP або навіть перевести на інший протокол (перехресне проксіювання). Спільною ситуацією є прикордонний маршрутизатор, що проксірує з мережі CoAP до сервісів HTTP для хмарних інтернет-з'єднань;

клієнт: ініціатор запиту. Кінцева точка призначення відповіді;

сервер: кінцева точка призначення запиту. Автор відповіді;

посередник: клієнт, який виступає як і сервер і клієнт по відношенню до вихідного сервера. Проксі є посередником;

сервер походження: сервер, на якому знаходиться даний ресурс;

спостерігачі: клієнт спостерігача може зареєструвати себе з використанням зміненого повідомлення GET. Потім спостерігач підключається до ресурсу, і якщо стан цього ресурсу змінюється, сервер надсилає повідомлення спостерігачеві.

Спостерігачі є унікальними в CoAP і дозволяють пристрою стежити за змінами в конкретному ресурсі. По суті це схоже на модель підписки MQTT, де вузол підписується на подію.

Нижче наведено приклад архітектури CoAP. Будучи легкою системою HTTP, клієнти CoAP можуть взаємодіяти один з одним або сервісами в хмарі, що підтримує CoAP. Як альтернативу проксі можна

використовувати для підключення до служби HTTP у хмарі. Кінцеві точки CoAP можуть встановлювати відносини один з одним навіть на рівні датчиків. Спостерігачі підтримують атрибути, подібні до підписки, для ресурсу, який змінюється, подібно до MQTT. На рис. 3.32 також показані сервери походження, на яких знаходяться ресурси, що спільно використовуються.

Два проксі дозволяють CoAP виконувати переклад на HTTP або надсилання запитів від імені клієнта (рис. 3.32).

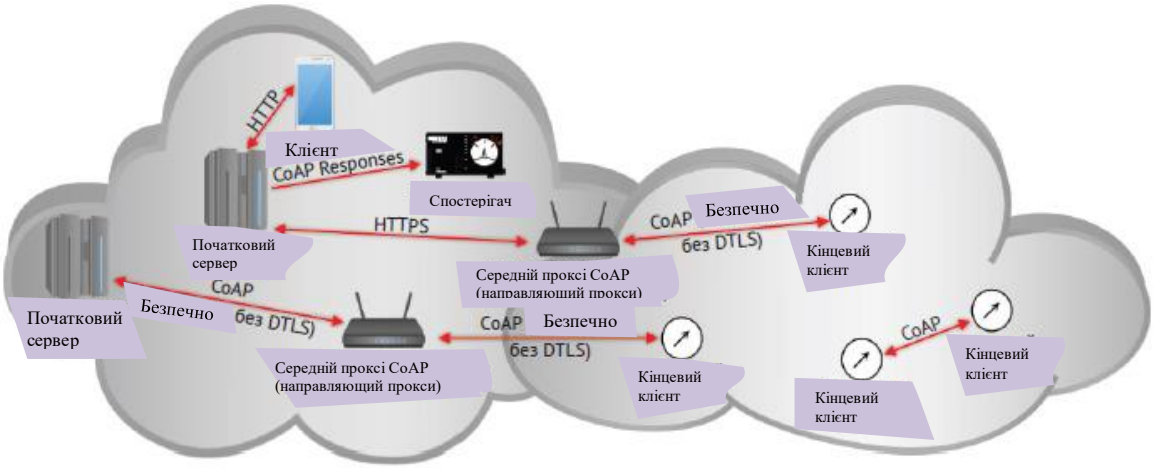


Рис. 3.32. Архітектура CoAP

CoAP використовує порт 5683. Цей порт повинен підтримуватись сервером, який пропонує ресурси, оскільки порт використовується для виявлення ресурсів. Порт 5684 використовується, коли DTLS увімкнено.

Формати повідомлень CoAP

Протоколи, засновані на транспорті UDP, мають на увазі, що з'єднання не може бути, за своєю суттю, надійним. Щоб компенсувати проблеми надійності, CoAP вводить два типи повідомлень, які відрізняються вимогою підтвердження доставки, або ні. Додатковою особливістю цього підходу є те, що повідомлення можуть бути асинхронними.

Загалом, у CoAP всього чотири повідомлення:

- **підтверджений (CON)** – потрібен ACK. Якщо повідомлення CON надіслано, ACK має бути прийнято протягом довільного інтервалу часу між ACK_TIMEOUT та (ACK_TIMEOUT * ACK_RANDOM_FACTOR). Якщо ACK не отримано, відправник передає повідомлення CON знову і знову з інтервалами, що експоненційно зростають, до тих пір, поки не отримає ACK або RST. Це, власне, форма контролю навантаження CoAP. Максимальна кількість спроб встановлена за допомогою MAX_RETRANSMIT. Це

механізм забезпечення відмовостійкості для компенсації відсутності такої в UDP;

- непідтверджений (NON) – не потрібно ACK. По суті, це повідомлення «спалити і забути» або транслювати широкомовно;

- підтвердження (ACK) – підтверджує повідомлення CON. Повідомлення ACK може повторюватися разом з іншими даними;

- скид (RST) – вказує, що отримано повідомлення CON, але контекст відсутній. Повідомлення RST може повторюватися разом з іншими даними.

CoAP – це дизайн RESTful з використанням повідомлень запит/відповідь, скомпонованих у повідомленнях CoAP. Це дозволяє підвищити ефективність та збереження смуги пропускання, як показано на рис. 3.33.

Рис. 3.33. Повідомлення CoAP NON та CON

На рисунку показано три приклади транзакцій запиту/відповіді CoAP, що не підтверджуються і підтверджуються. Вони визначені та описані наступним чином:

непідтверджений запит/відповідь (ліворуч) – повідомлення, передане між клієнтами A та B, з використанням типової конструкції HTTP GET. Через деякий час B здійснює деякі маніпуляції з даними та повертає температуру 20 °C;

підтверджений запит/відповідь (посередині) – включений ідентифікатор повідомлення, унікальний ідентифікатор для кожного повідомлення. Токен є значення, яке має бути правильним протягом усього періоду обміну;

підтверджений запит/відповідь (праворуч) – тут повідомлення може бути підтверджено. Обидва клієнти A і B чекатимуть ACK після кожного обміну повідомленнями. Щоб оптимізувати зв'язок, клієнт B може вибрати скомпонувати ACK з даними, що повертаються, як показано в крайньому правому кутку.

Процес повторної передачі показано на рис. 3.34.

Щоб врахувати відсутність відмовостійкості в UDP, CoAP використовує механізм тайм-ауту при спілкуванні з повідомленнями, що підтверджуються. Якщо закінчиться час очікування, або надсилаючи повідомлення CON, або отримавши ACK, відправник буде повторно надсилати повідомлення. Відправник відповідає за керування таймаутом та повторну передачу до максимальної кількості повторних передач. Зверніть увагу, що повторна передача ACK повторно використовує той же Message_ID

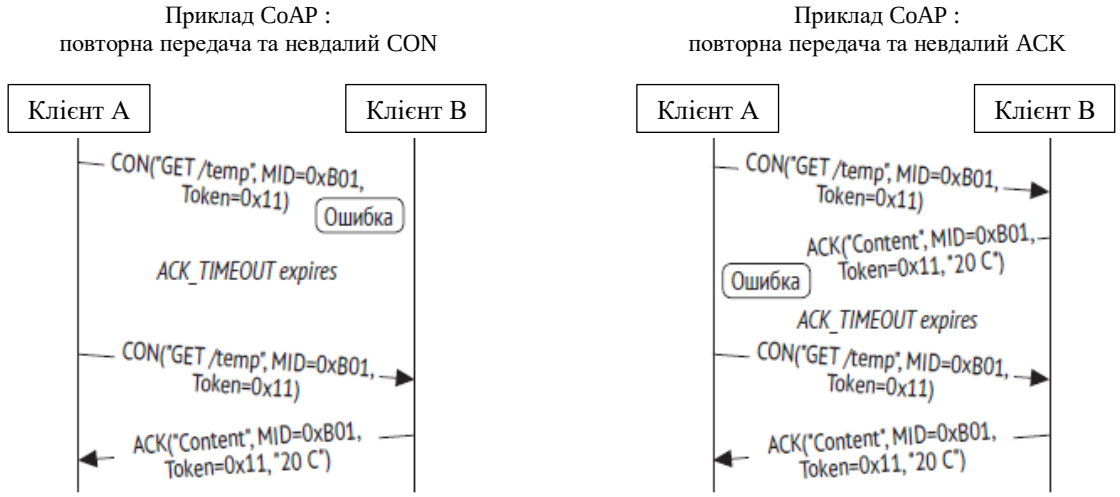


Рис. 3.34. Механізм повторної передачі CoAP

Хоча для іншої архітектури обміну повідомленнями потрібен центральний сервер передачі повідомлень між клієнтами, CoAP дозволяє передавати повідомлення між будь-якими клієнтами CoAP, включаючи датчики і сервери. CoAP включає просту модель кешування. Кешування контролюється через коди відповідей у заголовку повідомлення. Маска номера параметра визначає, чи є ключем кеша. Параметр Max_Age використовується для керування часом життя елемента кешу та забезпечення свіжість даних. Тобто Max_Age встановлює максимальний час, протягом якого може бути кешована відповідь, перш ніж вона повинна бути оновлена. Max_Age за замовчуванням має значення 60 секунд і може бути розширений до 136,1 року. Проксі грають роль кешуванні; наприклад, сплячий граничний датчик може використовувати проксі-сервер для кешування даних та економії енергії.

Заголовок повідомлення CoAP створено для забезпечення максимальної ефективності та збереження смуги пропускання. Заголовок має довжину чотири байти з типовим повідомленням запиту,

що містить тільки заголовки довжини від 10 до 20 байт. Це, як правило, у 10 разів менше, ніж заголовок у HTTP.

Структура складається з ідентифікаторів типу повідомлення (T), які мають бути включені до кожного заголовка разом із відповідним унікальним ідентифікатором повідомлення. Поле Code використовується для сигналізації про помилки або успішні стани по каналах. Після заголовка всі інші поля є необов'язковими та включають токени змінної довжини, опції та корисне навантаження (рис. 3.35).

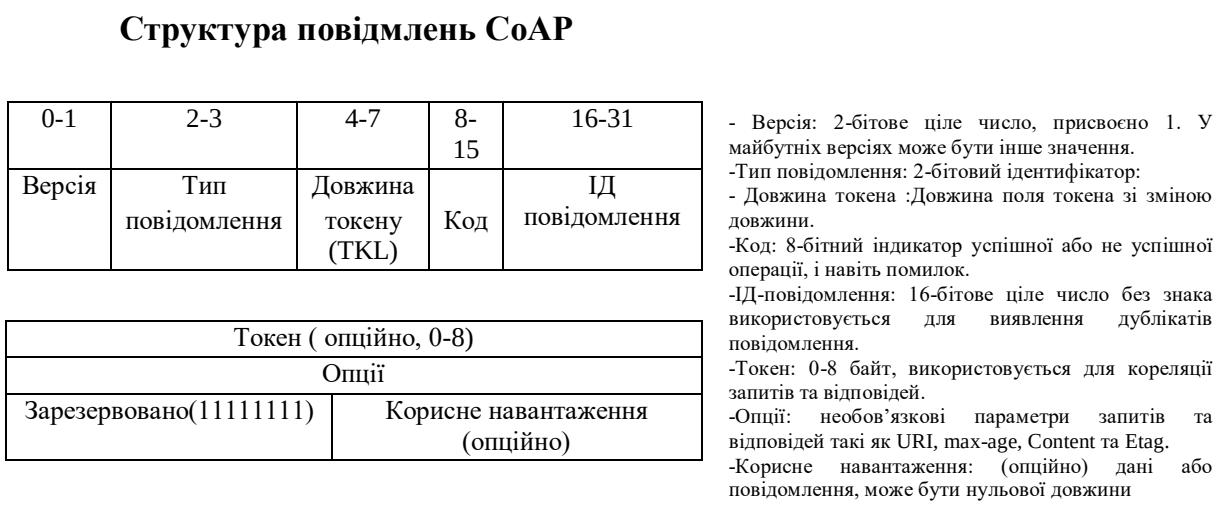


Рис. 3.35. Структура повідомлення CoAP

UDP також може призвести до появи повідомлень, що повторюються, як для передач CON, так і для NON. Якщо ідентичні Message_ID передаються одержувачу в межах запропонованого EXCHANGE_LIFETIME, вважається дублікат. Це може статися, як показано на попередніх малюнках, коли немає або скинуто ACK, і клієнт повторно передає повідомлення з тим же Message_ID. У специфікації CoAP зазначено, що одержувач ACK має отримати кожне повторне повідомлення, але повинен обробити лише один запит або відповідь. Це правило може бути спрощене, якщо повідомлення CON передає запит, який є ідемпотентним.

Як згадувалося, CoAP допускає роль спостерігача у системі. Це унікально, оскільки дозволяє CoAP поводитися аналогічно MQTT. Процес спостереження дозволяє клієнту реєструватися для спостереження, і сервер сповіщатиме клієнта щоразу, коли контрольований ресурс змінює свій стан. Тривалість спостереження може бути визначена під час реєстрації. Крім того, відносини спостереження закінчуються, коли клієнт, що ініціює, відправляє RST або інше повідомлення GET (рис. 3.36).

Як згадувалося раніше, у стандарті CoAP немає вбудованої автентифікації або шифрування, користувач повинен покладатися на DTLS для забезпечення такого рівня безпеки.

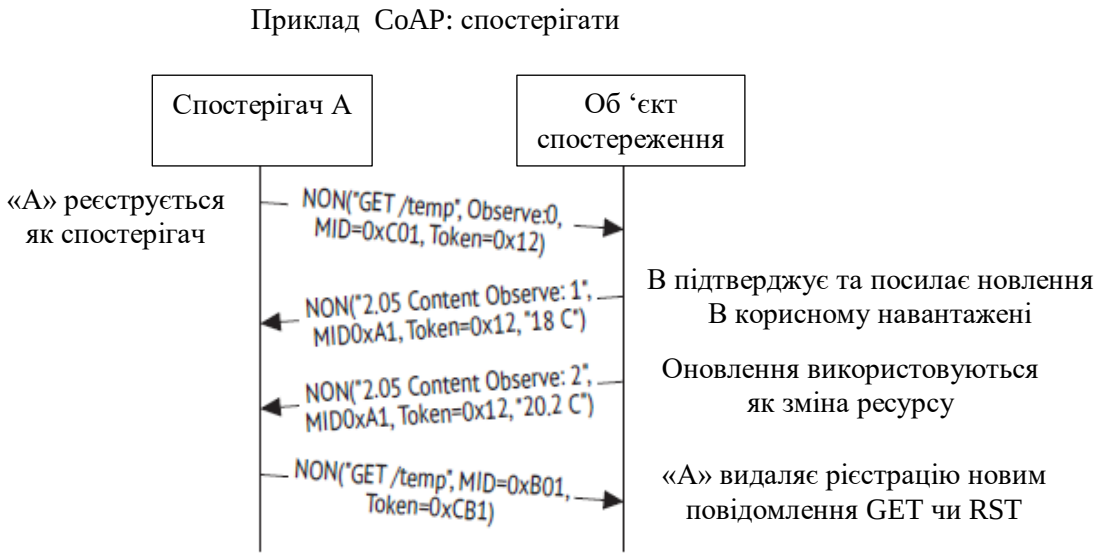


Рис. 3.36. Процес реєстрації та оновлення спостерігачів CoAP

Якщо використовується DTLS, то приклад URI буде наступним:

//небезпечно

coap://example.net:1234/~temperature/value.xml

// безпечно

coaps://example.net:1234/~temperature/value.xml

CoAP також пропонує механізми виявлення ресурсів. Просто відправлення запит GET на /.well-known/core буде розкривати список відомих ресурсів на пристрої. Крім того, рядки запиту можуть бути використані для застосування до запиту певних фільтрів.

3.4.4. Огляд протоколів інтернету речей

Існує багато інших протоколів обміну повідомленнями, які використовуються або пропонуються для розгортання IoT та M2M. У цьому підрозділі коротко розглянемо основні з них у алфавітному порядку [26].

Asterisk. Asterisk це **с**тек різних протоколів (SIP, H.323, MGCP і ін.) Для моніторингу і управління системою комп'ютерної телефонії Asterisk шляхом відправки команд CLI і обробки відповідей. Серед функцій Asterisk можна виділити голосову пошту, конференц-зв'язок, інтерактивне голосове меню і т. д. Asterisk використовується для

управління і контролю якості обслуговування віртуальної інфраструктури, бездротових, голосових мереж.

AMQP. AMQP означає Advanced Message Queuing Protocol. Це стабільний та перевірений протокол MOM, що використовується такими великими джерелами даних, як JPMorgan Chase, для обробки понад 1 мільярд повідомлень на день та Ocean Observatory Initiative для збору більше 8 терабайт океанографічних даних щодня.

Протокол добре зарекомендував себе у банківській та кредитній галузі, але також має місце у IoT. Крім того, AMQP стандартизований ISO та IEM як ISO/IEC 19464: 2014.

Протокол AMQP знаходиться у верхній частині стека TCP і використовує порт 5672 для зв'язку. Дані серіалізуються в AMQP, що означає, що повідомлення надсилаються в одиничних кадрах. Кадри передаються по віртуальних каналах, ідентифікованих унікальним channel_id. Кадри складаються із заголовків, channel_ids, інформації корисного навантаження та завершальної частини. Канал, однак, може асоціюватись лише з одним хостом. Повідомленням присвоюється унікальний глобальний ідентифікатор.

AMQP – система зв'язку, орієнтована повідомлення і контроль потоків. Це протокол рівня провідників та має низькорівневий інтерфейс. Провідниковий протокол звертається до API одразу над фізичним рівнем мережі. API-інтерфейс на рівні каналу дозволяє різним службам обміну повідомленнями, таким як .NET (NMS) та Java (JMS), взаємодіяти один з одним. Аналогічно AMQP намагається відокремити видавців від підписників. На відміну від MQTT, він має механізми для балансування навантаження та формалізації черги. Широко використовується протокол, заснований на AMQP, є RabbitMQ. RabbitMQ є брокером повідомлень AMQP, написаним на Erlang. Крім того, є кілька клієнтів AMQP, таких як клієнт RabbitMQ, написаний на Java, C#, Javascript і Erlang, а також Apache Qpid, написаний на Python, C++, C#, Java і Ruby.

Один або кілька віртуальних хостів з власними просторами імен, обмінами та чергами повідомлень будуть знаходитись на центральному сервері (серверах). Виробники та споживачі підписуються на послуги обміну. Служба обміну отримує повідомлення від видавця та надсилає дані у відповідну чергу. Це відношення називається прив'язкою, і прив'язка може бути прямою до однієї черги, або до кількох черг (як у широкомовній передачі). Як альтернатива прив'язка може асоціювати один обмін з однією чергою з використанням ключа маршрутизації; це формально називається прямим обміном. Іншим типом обміну є обмін теми.

BACnet (Building Automation and Control Networks). Відкритий протокол автоматизації і управління інженерними мережами. Підтримує специфікації BACnet IP і BACnet MS / TP для читання / запису властивостей об'єктів, доступу до сервісів пристроїв і обробки сповіщень. Протокол BACnet є міжнародним рідним стандартом ISO 16484-5. Використовується для домашньої і промислової автоматизації, управління процесами, тестування і вимірювань, міжмашинної взаємодії (M2M).

CAP (Common Alerting Protocol). Протокол загального оповіщення, призначений для обміну повідомлень CAP. Використовується для прийому і передачі сигналів про аварійні або позаштатні ситуації.

CoAP (Constrained Application Protocol). Веб-протокол передачі даних для використання в обмежених вузлах і мережах Інтернету речей (детально розглядався раніше).

CORBA (Common Object Request Broker Architecture). Протокол для здійснення інтеграції ізольованих систем, який дає можливість програмам, написаним на різних мовах програмування, що працюють в різних вузлах мережі, взаємодіяти один з одним так само просто, ніби вони перебували б в адресному просторі одного процесу. Протокол необхідний для здійснення дзвінків CORBA через IP мережу зі специфікацією вхідних параметрів і обробкою даних відповіді.

CWMP (CPE WAN Management Protocol). Протокол управління і моніторингу абонентського обладнання (CPE) відповідно до специфікації TR-0695. Протокол призначений для автоматичної конфігурації та динамічної підготовки сервісів до роботи, управління версіями програмного забезпечення, моніторингу станів і продуктивності, діагностики пристроїв і мережі в цілому.

DDS (Service Distribution Service) - реалізує шаблон публікації-підписки для відправлення та прийому даних, подій та команд серед кінцевих вузлів. Вузли-видавці створюють інформацію, «topic» (теми, розділи: температура, місцеположення, тиск) і публікують шаблони. Вузлам, що зацікавлені в даних розділах, DDS прозоро доставляє створені шаблони. В якості транспортного протоколу застосовується UDP. Також DDS дозволяє управляти параметрами QoS (якість облування).

DHCP (Dynamic Host Configuration Protocol). Протокол динамічного налаштування вузла, що дозволяє пристрою автоматично встановлювати отримувати IP-адресу і інші параметри, необхідні для роботи в мережі TCP/IP [5]. У IoT служить для моніторингу працездатності DHCP-сервера.

DLMS/COSEM (Distribution Line Message Specification/COmpanion Specification for Energy Metering). Протокол, який базується на концепціях моделі OSI, регламентує обмін даними між приладами обліку (лічильниками) та системами збору даних, в основі якого лежить клієнт-серверна архітектура. Основними специфікаціями в цьому стандарті є DLMS і COSEM. Використовується для отримання поточних показань приладів обліку та історії. Віддалений збір даних економить постачальникам комунальних послуг витрати, пов'язані зі збором показників лічильників. Ще однією перевагою є те, що виставлені рахунки будуть відповідати реальним показникам споживання, а не ґрунтуватися на переданій оцінці.

DNP3 (Distributed Network Protocol). Протокол передачі даних між об'єктами мережі IoT: читання/запис, вибір даних і управління процесами: пряме управління, управління подіями і т. д. Для безпечної аутентифікації використовується розширення Secure Authentication.

DNS (Domain Name System). Протокол для отримання IP-адреси по імені пристрою [6]. В рамках IoT використовується для моніторингу працездатності DNS-сервера.

Ethernet/IP. Відкритий промисловий протокол, який підтримує обмін повідомленнями (обмін повідомленнями введення/виведення в реальному часі). Стандарт EtherNet/IP забезпечує об'єднання в єдиний інформаційний простір всіх компонентів систем автоматизації IoT – від рівнів засобів введення/виведення, контролерного обладнання, серверів до рівня систем управління підприємством.

EVA-DTS. Протокол передачі даних для торгових автоматів. Стандарт визначає структуру загальних елементів даних і описує засоби передачі даних. Використовується також для моніторингу торгових автоматів (локальних файлів і папок), збору доступних метрик, статистики та помилок EVA-DTS, перевірки контрольних сум, завантаження вмісту файлів в ядро системи.

FTP (File Transfer Protocol). Протокол прикладного рівня стека TCP/IP, призначений для передачі файлів в мережі [7]. В IoT використовується для моніторингу атрибутів віддалених файлів, контролю працездатності FTP-сервера (моніторинг доступності, завантаження процесора, використання дискового простору і пам'яті, стану процесів, а також нестандартних метрик для серверів, що працюють на різних платформах і операційних системах).

GPS/GLONASS і M2M Data. Система для отримання довільних звітів від будь-яких супутникових датчиків і інших пристроїв M2M через протоколи TCP або UDP. Обробка команд здійснюється на основі бізнес-правил. В IoT дана система лежить в основі управління

транспортом і служить для визначення і відстежування місцеположення, збору, зберігання, обробки і візуалізації різних телеметричних даних. З її допомогою можна також здійснювати моніторинг поточного стану автомобіля (швидкість, паливо/рівень заряду батарей, поточну витрату, рівень масла, і ін.), експлуатацію транспортних засобів (віддалене вимикання двигуна, відправка повідомлень водієві або оператору і т. д.), моніторинг профілактичних і ремонтних робіт, настройку обміну даними з системою управління запасами в режимі реального часу.

HTTP/HTTPS (HyperText Transfer Protocol). Протокол прикладного рівня для передачі даних [8]. Основою HTTP є клієнт-серверна архітектура. Для IoT більш безпечна реалізація є тільки HTTP клієнта на пристрої, тобто пристрій може ініціювати з'єднання, а не отримувати його. Завданнями протоколу HTTP/HTTPS є надання доступу до внутрішніх веб-серверів контролерів/модулів, які перебувають в мережах, захищених брандмауерами або NAT, а також завантажувати вміст веб-сторінок в ядро системи, моніторинг працездатності веб-сервера.

ICMP (Internet Control Message Protocol). Мережевий протокол, що використовується для оповіщення про помилки на пристроях в мережі. Будь-який пристрій має можливість відправляти, отримувати або обробляти повідомлення ICMP. У більшості випадків ICMP не використовується між кінцевими користувачами, а використовується адміністратором мережі для усунення неполадок або перевірки втрат пакетів на маршрутах. Але існують і призначені для користувача утиліти, такі як моніторинг доступності (ping), трасування мережевих маршрутів (traceroute).

IMAP (Internet Message Access Protocol). Протокол прикладного рівня моделі TCP/IP, призначений для доступу до електронної пошти [10]. IMAP дозволяє не тільки приймати повідомлення, але і керувати електронною поштою прямо на поштовому сервері, тобто при перегляді листи не завантажуються на пристрій, а залишаються на сервері. Також протокол необхідний для моніторингу працездатності IMAP-сервера.

IPMI (Intelligent Platform Management Interface). IPMI представляє собою набір інтерфейсів і протоколів для управління і контролю можливостей програмного забезпечення і операційної системи незалежно від використання пристрою. Системні адміністратори можуть використовувати IPMI повідомлення для моніторингу стану серверів і пристроїв (наприклад, значення температури, напруги, стан вентиляторів, джерел живлення і т.д.)

JMS (Java Message Service). Стандарт для обміну повідомленнями, широко використовується для інтеграції серверних

додатків, таких як бази даних, аналітичних систем і автоматизації бізнес-процесів. JMS використовується в основному з Java-додатками. JMS широко використовується в центрах обробки даних, що дозволяє легко інтегрувати в серверні додатки. З недоліків для IoT можна відзначити високі вимоги до живлення, обробки і пам'яті пристроїв.

JMX (Java Management Extensions). JMX визначає стандарт для написання JMX-об'єктів, так званих MBean'ов. Будь JMX-клієнт (віддалений або локальний додаток) має можливість читати, записувати, викликати методи і отримувати доступ до атрибутів цим MBean'ом за допомогою контейнера, в якому вони містяться. Також JMX дозволяє запитувати конфігураційні установки серверів і змінювати їх під час виконання програми. Крім цього JMX здійснює моніторинг, формує повідомлення про події, таймер і виконує динамічне завантаження класів з XML-файлів.

KNX. Протокол для взаємодії з пристроями KNX через IP. Система KNX служить для домашньої і промислової автоматизації (управління освітленням, кліматом, забезпечення безпеки та ін.). Всі пристрої мережі (в термінах KNX - датчики, актуатори і системні пристрої) підключаються по провідній шині, через яку і обмінюються інформацією з використанням протоколу KNX. Обмін інформацією обов'язково супроводжується підтвердженням на кожне повідомлення, тому дана технологія не підходить мереж критичних до затримок.

LDAP (Lightweight Directory Access Protocol). Протокол прикладного рівня для доступу до каталогів, а саме до системи управління базами даних. Є спрощеною версією протоколу DAP. LDAP використовується для формування запитів на читання, пошук, редагування, встановлення і розриву зв'язку, загрузки результатів запитів в ядро системи. Система LDAP містить в собі готові схеми зберігання даних, наприклад, структуру облікових записів, довідника адрес, прав і привілеїв. Таким чином, в каталогах LDAP містяться дані, необхідні для аутентифікації користувача (ім'я користувача, пароль), дані про права користувача, призначені для користувача настройки, дані про групи людей, наприклад, списки розсилки повідомлень.

LON/LonTalk. Протокол для обміну даними, призначений для моніторингу і управління мережевими пристроями, взаємодіючими через різні середовища комунікації такі, як кручена пара, лінії електроживлення, оптоволокно, і бездротове радіочастотне середовище. Протокол використовується для задач автоматизації різних функцій в промисловому управлінні, домашній автоматизації, моніторингу транспортних засобів, а також в системах автоматизації будівель таких,

як системи управління освітленням і системи опалення, вентиляції, кондиціонування, системи інтелектуального будинку.

MDB (Multi Drop Bus). Протокол для взаємодії пристроїв, підключених до комп'ютерної шини MDB. У будь-якій точці шини можна визначити, який пристрій посилає інформацію. Ведені (ведомий-ведущий) пристрої фільтрують данні в шині, які їм необхідно отримати. MDB шини використовуються в торгових автоматах, де контролери обмінюються даними з компонентами торгового автомата, наприклад, приймачами грошових коштів. Можлива робота в двох режимах: режим веденого (моніторинг транзакцій) і режим ведучого (обробка транзакцій, ініційованих сервером). Також MDB використовується для моніторингу вхідних даних по послідовному порту або TCP/UDP з'єднанню.

МЕК 60870-5-104. Протокол телемеханіки, призначений для передачі даних в центри управління. Передача здійснюється після установки TCP з'єднання. Використовується в рішеннях домашньої і промислової автоматизації (системи кондиціонування, освітлення, відеоспостереження, отримання даних від лічильників, вимірювальних перетворювачів), контролю і управління центрами обробки даних.

Meter-Bus. Протокол для взаємодії пристроїв по шині M-Bus. Технології M-bus переважно застосовується в автоматизованих системах для зняття показань з приладів обліку електричної енергії (електролічильники), теплової енергії (теплолічильники), витратомірів води та газу. Дані передаються на комп'ютерну станцію (сервер) безпосередньо або через концентратори шини M-Bus, а також підсилювачі-повторювачі сигналу.

Modbus. Відкритий комунікаційний протокол для обміну даними між мережевими пристроями, заснований на архітектурі ведучий-ведений (master-slave) Обмін даними являє собою транзакції, що складаються із запитів і відповідей. Перевагами протоколу Modbus є відкритість і масовість використання (безліч виробників випускають різні типи пристроїв, датчиків, що підтримують даний протокол). Задачами протоколу Modbus є контроль і управління мережами, читання і запис даних в регістри зберігання, доступ до файлів, діагностика стану пристроїв.

MQTT. Мережевий протокол для обміну повідомленнями в мережах з низькою пропускною здатністю між пристроями, що реалізують модель ведучий-ведений (детально розглядався раніше).

NetFlow. Відкритий протокол, розроблений компанією Cisco для моніторингу трафіку в мережі, і підтримується не тільки обладнанням даної компанії, але і пристроями від інших виробників. Головними завданнями протоколу Netflow є декомпозиція і глибокий аналіз

мережевого трафіка [11]. Netflow надає можливість аналізу мережевого трафіку на рівні сеансів, роблячи запис про кожну транзакції TCP/IP. Протокол Netflow містить в собі три компоненти: сенсор, колектор і аналізатор. Сенсор представляє собою пристрій, що фіксує дані про конкретний сеанс зв'язку, які проходять через нього. Іншими словами, сенсор фіксує потоки, що проходять через нього. Під потоками маються на увазі всі пакети, що мають однакові поля адреси, порту, коду повідомлень ICMP, версії протоколу IP, мережевого інтерфейсу. Колектор необхідний для збору інформації від сенсорів і розміщенні її в певних базах даних. Аналізатор в свою чергу здійснює обробку даних і їх візуалізацію.

NMEA 0183 (National Marine Electronics Association). Протокол для передачі повідомлень між пристроями, зокрема транспортними засобами, оснащеними прийомопередавачами GPS/ГЛОНАСС. Всі повідомлення передаються в текстовому вигляді, і кожне повідомлення закінчується значенням контрольної суми, для перевірки цілісності на приймальній стороні. Протокол NMEA 0183 використовується в основному для відстеження місцезнаходження пристроїв, часу місцезнаходження, може містити дані швидкості і напрямку руху, кількості використовуваних супутників для визначення координат, відношенні сигнал/шум кожного видимого супутника. У протоколі NMEA 0183 є обмеження для максимальної довжини повідомлення, вона не повинна перевищувати 80 символів.

ODBC (Open Database Connectivity). Програмний інтерфейс, заснований на SQL, для доступу до баз даних. Призначений для уніфікації взаємодії між додатком і різними джерелами даних. Постачальники даних розробляють драйвера для взаємодії зі стандартними функціями ODBC API з урахуванням свого продукту, що дозволяє прикладним програмістам розробляти програми з можливістю доступу до даних, не вивчаючи особливості роботи з базами даних. Приклад використання – Integration Manager. Програма, що дозволяє об'єднати кілька складних баз даних в одну структуру з можливістю спрощеного збору, подання та передачі інформації.

OPC (OLE for Process Control). Стандарт інтерфейсів для спільної роботи засобів автоматизації, що функціонують на різних апаратних платформах, в різних промислових мережах і вироблених різними фірмами. Стандарт працює з таким операційними системами як Windows, Linux і Mac OS. Основною частиною стандарту є специфікація OPC DA для протоколу взаємодії між клієнтом і апаратною частиною пристрою (контролерами, модулями введення/виведення і т. д.) в режимі реального часу. Існує чотири режими роботи сервера OPC DA:

синхронний, асинхронний, режим підписки і режим поновлення даних. Сервер OPC DA активно застосовується в рішеннях промислової і домашньої автоматизації.

OPC UA (OPC Unified Architecture). Стек протоколів OPC UA є частиною розглянутого вище стандарту OPC. На відміну від протоколу OPC DA, OPC UA встановлює взаємодію між сервером і клієнтом, що не залежить від програмної і апаратної частини пристроїв і від типу мережі. Основою архітектури OPC UA є SOA (архітектура, орієнтована на послуги). Під послугою розуміється набір функцій, прописаних в конкретному ПО. Даний набір може передаватися від сервера клієнту і навпаки. повідомлення передаються у вигляді XML тексту або бінарного файлу. Кейси застосування OPC UA аналогічні OPC DA.

POP3 (Post Office Protocol Version 3). Протокол для прийому повідомлень електронної пошти [12]. При перегляді пошти з використанням протоколу POP3 всі електронні листи зберігаються на пристрій користувача та автоматично видаляються з сервера. Всі подальші дії з листами будуть виводитися саме на пристрої. Перевагами використання даного протоколу є можливість отримання доступу до поштової скриньки навіть при відсутності Інтернету.

RADIUS (Remote Authentication Dial In User Service). Протокол служби дистанційної аутентифікації користувачів по комутованих лініях. Протокол RADIUS має клієнт-серверну архітектуру і представляє собою службу без встановлення з'єднання. Сервер RADIUS може підтримувати безліч методів аутентифікації користувача. Для перевірки справжності імені користувача та пароля, наданих сервера, можуть використовуватися протоколи PPP, PAP, CHAP, вхід UNIX та інші механізми аутентифікації.

SIP (Session Initiation Protocol). Протокол сигнальної інформації, призначений для управління сеансами мультимедійного зв'язку. Найбільш розповсюдженим варіантом застосування протоколу SIP є здійснення IP-телефонії (передача голосу, відео, миттєвих повідомлень по IP мережі). Стандарт протоколу SIP описаний в RFC 3261.

SMB/CIFS (Server Message Block). Протокол з клієнт-серверною архітектурою, для віддаленого доступу до мережевих ресурсів (файлів, принтерів та ін.). Common Internet File System - перша версія протоколу. Актуальна версія SMB 2.0 реалізована Microsoft і використовується в Microsoft Windows Network. Клієнти з'єднуються з сервером, використовуючи протокол NetBIOS через TCP/IP, NetBEUI або IPX / SPX. Після установки з'єднання, клієнти можуть посилати SMB-команди сервера, який дає їм доступ читання та запису до файлів, і дозволяє виконувати весь перелік дій, які можна виконувати з файловою

системою. Однак в разі використання SMB ці дії відбуваються через мережу. Протокол використовується для отримання доступу і моніторингу файлів і папок за технологією Microsoft Windows Network (SBM/CIFS).

SMI-S (Storage Management Initiative Specification). Стандарт управління дисковими сховищами. SMI-S є ANSI і ISO стандартом. Основна ідея стандарту - уніфікація управління дисковими сховищами через веб-запити. Більше 800 різних апаратних і 75 програмних рішень підтримують даний стандарт. System Center Virtual Machine Manager 2012 дозволяє підключити SMI-S сховища, так щоб адміністратор мав можливість з консолі VMM отримувати інформацію про LUN, групи RAID, вільне місце у сховищі і так далі.

SMPP (Short Message Peer-to-Peer). Відкритий стандарт в телекомунікаційній галузі, розроблений спеціально для забезпечення гнучкого інтерфейсу передачі коротких повідомлень (ESME) між зовнішніми пристроями або додатками, маршрутизаторами і центрами повідомлень (SMSC). Через свою універсальність і підтримку SMS-протоколів без GSM, SMPP є найбільш широко використовуваним протоколом для короткого обміну повідомленнями. SMPP реалізований на Java в проект jSMPP, на Python в проекті python-SMPP і на PHP в PHP-SMPP.

SMTP. Протокол, призначений тільки для відправки повідомлень на поштовий сервер [13]. В рамках IoT протокол SMTP використовується для відправки динамічно згенерованих повідомлень на вимогу, у відповідь на подію або відповідно до розкладу.

SNMP (Simple Network Management Protocol). Протокол для управління пристроями в IP-мережах на основі архітектури TCP/UDP. Всі керовані комутатори або маршрутизатори забезпечують інформацію про стан своїх портів/інтерфейсів в реальному часі за протоколом SNMP. Ця інформація доступна і використовується вбудованими засобами обробки даних (тривогами, звітами, діаграмами та ін.). Протокол має три версії, з яких SNMPv1 і SNMPv2 вважаються застарілими, а SNMPv3 - повний Інтернет стандарт з максимальним рівнем готовності для RFC. На практиці підтримуються всі три версії.

SOAP (Simple Object Access Protocol). Протокол технологій веб-служб, призначений для обміну довільними XML повідомленнями і виклику процедур. Остання версія 1.2, по якій SOAP є розширенням протоколу XML-RPC. SOAP може використовуватися з будь-яким протоколом прикладного рівня, однак його взаємодія з кожним із цих протоколів має свої особливості, які повинні бути визначені окремо. Найчастіше SOAP використовується з HTTP. Прикладом використання

можуть служити запити на сервер інтернет-магазину, що містять id товару, для отримання відповіді з докладним описом.

SQL (Structured query language). Структурована мова запитів для роботи з базами даних різних розмірів. Основним завданням SQL є формування і відправка динамічно згенерованих запитів SELECT/UPDATE/INSERT/DELETE і подальша завантаження результатів запитів в ядро системи.

SSH (Secure Shell). Протокол прикладного рівня, що забезпечує безпечне з'єднання і передачу даних між пристроями. Це досягається завдяки використанню шифрування трафіку. Також можливе створення SSH тунелю між користувачами. В цьому випадку незашифровані дані шифруються на одній стороні тунелю і розшифровуються на інший. В загальному випадку протокол SSH використовується для віддаленого доступу до пристроїв, виконання скриптів і додатків на віддалених комп'ютерах.

STOMP. STOMP означає простий (або потоковий) протокол проміжного рівня, орієнтований текстові повідомлення. Це текстовий протокол, розроблений Codehaus до роботи з орієнтованим повідомлення проміжним ПЗ. Брокер, розроблений однією мовою програмування, може отримувати повідомлення від клієнта, написаного іншою мовою. Протокол схожий на HTTP і працює з TCP. STOMP складається з заголовка кадру та тіла кадру. Він відрізняється від багатьох протоколів, оскільки він не стосується питань підписки чи черг. Він просто використовує семантику, подібну до HTTP, таку як SEND з рядком мети. Брокер повинен проаналізувати спілкування та зіставити йому тему чи чергу для клієнта. Споживач даних підписуватиметься на пункти призначення, надані брокером.

У STOMP є клієнти, написані на Python (stomp.py), TCL (tStomp) та Erlang (stomp.erl). Деякі сервери мають рідну підтримку STOMP, наприклад RabbitMQ (через плагін), і деякі сервери були розроблені специфічними мовами (Ruby, Perl або OCaml).

Syslog (System log). Стандарт і мережевий протокол відправки і реєстрації повідомлень про події (логів), що відбуваються в комп'ютерних мережах, основаних на протоколі IP. Механізм Syslog залишається незмінним з незначними варіаціями: джерела формують прості текстові повідомлення невеликого розміру (до 1024 байт) про події, що відбуваються в них та передають їх на оброблення серверу «syslog server», використовуючи один з мережевих протоколів UDP або TCP. Формування повідомлень про події та їх передача відбувається за певними правилами, так званим протоколом Syslog. Як правило, повідомлення відсилається у відкритому вигляді, але можливо

шифрування повідомлень і відправка їх по SSL/TLS. Джерела повідомлень і сервер Syslog можуть розміщуватись на різних машинах, що дозволяє організувати збір і зберігання повідомлень від безлічі географічно рознесених різнорідних джерел в єдиному сховищі, що дає доступ відразу до всіх пристроїв і комп'ютерів в мережі.

Telnet (Terminal network). Протокол прикладного рівня, призначений для здійснення віддаленого доступу до пристроїв. Принцип роботи аналогічний протоколу SSH з відмінністю лише в тому, що дані передаються в незазашифрованому вигляді. Завданнями протоколу Telnet є виконання скриптів і додатків на віддалених комп'ютерах, обробка даних, що надходять від пристрою, і приведення їх до стандартного вигляду. Telnet надає повний доступ до функціональних можливостей пристрою і може використовуватися для віддаленого управління робототехнічними системами в IoT.

VMware SOAP API. Програмне забезпечення фірми VMware, що надає інтерфейс прикладного програмування (API) Lab Manager system. Використовуючи безпечний API, можна підключитися до сервера Lab Manager для виконання або автоматизації різних операцій. Lab Manager SOAP API використовує XML-технології, включаючи SOAP в якості протоколу зв'язку, і опис веб-служб (WSDL) в якості мови опису інтерфейсу. Використовуючи переважно середовище розробки з підтримкою Web, ви можете створити клієнта Web-сервісу додатків, що використовують стандартні протоколи веб-сервісів, для програмного виконання завдань: запит інформації про віртуальну машину і конфігурації, виконання дії на машинах і конфігураціях, захоплення, перевірка, клонування, видалення і розгортання конфігурацій, створення URL-адреси конфігурації LiveLink, який ви можете надіслати електронною поштою іншим членам команди.

WMI (Windows Management Instrumentation). Розширена і адаптована під Windows реалізація стандарту WBEM, прийнятого багатьма компаніями, в основі якого лежить ідея створення універсального інтерфейсу моніторингу і управління різними системами і компонентами розподіленого інформаційного середовища підприємства з використанням об'єктно-орієнтованої ідеології і протоколів HTML і XML. Використовується для моніторингу властивостей об'єктів, виконання WQL запитів і методів об'єктів, обробки подій.

Контрольні питання до розділу 3

1. Назвіть особливості стандарту IEEE 802.15.4.
2. Назвіть особливості стандарту Bluetooth (IEEE 802.15.1).
3. Назвіть особливості стандарту Z-Wave
4. Чим відрізняється стандарт 6LoWPAN від звичайного IP?
5. Чим відрізняється стандарт Thread від звичайного IP?
6. Назвіть особливості LoRa та LoRaWAN.
7. Назвіть особливості Sigfox.
8. Порівняйте між собою протоколи MQTT та MQTT-SN.
9. Поясніть особливості обмеженого прикладного протоколу CoAP.

4. ХМАРНІ, ТУМАННІ ТЕХНОЛОГІЇ ТА АНАЛІЗ ДАНИХ В IoT

Кількість пристроїв IoT безупинно росте з кожним днем, і вони генерують величезний обсяг даних. Тому виникає потреба в зручній архітектурі системи, що здатна обробляти, зберігати і передавати ці дані. Зараз для цих цілей використовують хмарні сервіси. «Хмари» здатні закрити більшість запитів IoT. Наприклад, забезпечити моніторинг служб, швидку обробку будь-яких обсягів даних, що генеруються пристроями, а також їх візуалізацію. Однак використовувати хмарні сервіси при вирішенні завдань управління у реальному часі досить проблематично із-за суттєвих і погано прогнозуємих затримок сигналів. Тому все більш популярною стає парадигма туманних обчислень (Fog), що здатна доповнити хмарні рішення, масштабувати і оптимізувавши інфраструктуру IoT. Туманні обчислення забезпечують швидкий відгук на запити і мінімальну затримку при обробці даних. Тобто Fog саме доповнює «хмари», розширює його можливості.

4.1. Хмарні технології в IoT

Вважається, що ідея хмарних обчислень з'явилася ще в 1960 році, коли Джон Маккарті висловив припущення, що коли-небудь комп'ютерні обчислення будуть проводитися за допомогою «загальнонародних утиліт». Є відомості, що перший опис хмари виник в компанії Compaq в середині 1990-х років, де технологічні футуристи передбачали обчислювальну модель, яка переводила обчислення в мережу і на хости. Однак, ідеологія хмарних обчислень отримала популярність з 2007 року завдяки швидкому розвитку каналів зв'язку і стрімко зростаючим потребам користувачів [27].

Під хмарними обчисленнями (від англ. Cloud computing, також використовується термін «хмарна (розсіяна) обробка даних») зазвичай розуміється надання користувачу комп'ютерних ресурсів і потужностей у вигляді інтернет-сервісу. Таким чином, обчислювальні ресурси надаються користувачеві в «чистому» вигляді, і користувач може не знати, які комп'ютери обробляють його запити, під керуванням якої операційної системи це відбувається і т.д.

Під *хмарними технологіями* будемо розуміти модель системи процедур, що забезпечують цілеспрямовану зміну матеріальних і ідеальних (віртуальних) об'єктів.

Саме поняття «хмара» відноситься до інфраструктури обчислювальних служб, які зазвичай надаються відповідно до запиту користувача. Набір ресурсів (обчислень, мереж, сховищ і пов'язаних з ними програмних сервісів) може динамічно масштабуватися в бік збільшення або зменшення в залежності від середнього навантаження і якості обслуговування. Як матеріальні об’єкти, хмари, представляють собою тисячі серверів, що розміщені у великих центрах обробки даних (ЦОД), які надають клієнтам послуги, орієнтовані на зовнішнього споживача, і варіанти оплати за використання цих послуг. ЦОД створюють ілюзію єдиного хмарного ресурсу, в той час як насправді може бути використано багато географічно розподілених ресурсів. Це надає користувачеві відчуття незалежності від місця розташування. Ресурси є еластичними, що означає можливість масштабування під вимоги користувача. Сервіси, які працюють в хмарі, відрізняються від традиційного програмного забезпечення своєю конструкцією та реалізацією. Хмарні додатки можуть розроблятися і розвиватися швидше і меншою мірою залежати від мінливості середовища. Таким чином, розгортання хмар відбувається дуже швидко.

4.1.1. Види хмар та хмарна архітектура

У хмарному середовищі розрізняють три різні моделі топології розгортання хмар [27]: приватна хмара (private cloud), хмара загального користування (public cloud) та гібридна хмара (hybrid cloud). Незалежно від моделі, фреймворки хмар повинні забезпечувати динамічну масштабованість, швидкість розробки і розгортання, а також появу в локальному місці незалежно від його близькості.

Приватна хмара. У приватній хмарі інфраструктура надана одній організації або корпорації, що може включати декілька споживачів. Інфраструктура розміщується у приміщеннях організації, а її спільне використання та розпорядження ресурсами є загальними. Немає концепції спільного використання ресурсів або об'єднання поза межами власної інфраструктури власника.

Приватна хмара вважається безпечною, та дозволяє проводити перевірку якості, для гарантії, що інформація обробляється виключно системами, керованими клієнтом. Однак, щоб вважатися хмарою, повинні існувати деякі аспекти хмарних сервісів, такі як віртуалізація і балансування навантаження. Приватна хмара може бути локальною або може бути розгорнутою на обладнанні, що надається третьою стороною виключно для її використання.

Публічна хмара – інфраструктура, що надається на вимогу для багатьох клієнтів і додатків. Інфраструктура являє собою набір ресурсів, які будь-яка людина може використовувати в будь-який час в рамках своїх угод про рівень обслуговування. Перевага тут у тому, що явна шкала хмарних центрів обробки даних дозволяє забезпечити безпрецедентну масштабованість для багатьох клієнтів, які обмежені тільки тим, яку частину послуг вони хочуть придбати.

Гібридна хмара. Гібридна архітектурна модель являє собою поєднання приватних і громадських хмар. Такими комбінаціями можуть бути використання одночасно декількох публічних хмар, або комбінація громадської та приватної хмарної інфраструктури. Організації використовують гібридну модель, якщо є дані, які потребують унікального підходу, а як інтерфейс можна використовувати хмару. Іншим варіантом використання є дотримання угоди з хмарними провайдерами про компенсацію умов недостатньої масштабованості власних ресурсів. У цьому випадку публічна хмара буде використовуватися як балансувальник навантаження до тих пір, поки кількість даних і їх використання не повернуться в обмежений простір приватної хмари. Цей варіант використання називається хмарним вибухом і відноситься до використання хмар в якості умовних ресурсів.

Сучасні корпоративні системи, як правило, використовують гібридну архітектуру для забезпечення безпеки критично важливих додатків і даних на місцевості та використовують публічну хмару для простоти підключення і швидкості розгортання.

4.1.2. Моделі хмарних сервісів

Все, що стосується Cloud computing (CC), зазвичай прийнято називати aaS - «as a Service», тобто «як сервіс», або «у вигляді сервісу». Хмарні провайдери зазвичай підтримують цілий ряд продуктів «*Все як сервіс*» (XaaS). На даний час концепція передбачає надання наступних типів послуг своїм користувачам [27]:

Storage-as-a-Service («зберігання як сервіс»). Найпростіший з CC-сервісів, що представляє собою дисковий простір на вимогу. Послуга Storage-as-a-Service дає можливість зберігати дані в зовнішньому сховищі (в «хмарі»). Для користувача воно буде виглядати, як додатковий логічний диск або папка. Сервіс є базовим для інших, оскільки входить до складу практично кожного з них.

Database-as-a-Service («база даних як сервіс»). Послуга більше для адміністрування, бо надає можливість працювати з базами даних, подібно так, як СУБД було встановлено на локальному ресурсі. В цьому

випадку значно легше розділяти проекти між різними виконавцями та заощадити на комп'ютерному обладнанні та ліцензіях, необхідних для грамотного використання СУБД у великій чи середній організації.

Information-as-a-Service («інформація як сервіс»). Дає можливість віддалено використовувати будь-які види інформації, яка може змінюватися щохвилини або навіть щомиті.

Process-as-a-Service («управління процесом як сервіс»). Віддалений ресурс, який може зв'язати воєдино кілька ресурсів (таких як послуги або дані, що містяться в межах однієї «хмари» або інших доступних «хмар»), для створення єдиного бізнес-процесу.

Application-as-a-Service («додаток як сервіс»). Також називається, Software-as-a-Service («ПЗ як сервіс»). Позиціонується як «програмне забезпечення на вимогу», яке розгорнуто на віддалених серверах і кожен користувач може отримувати до нього доступ за допомогою Інтернету, причому всі питання оновлення та ліцензій на дане забезпечення регулюється постачальником даної послуги. Оплата, в даному випадку, відбувається за фактичне використання останнього. Як приклад можна навести Google Docs, Google Calendar і т.п. онлайн-програми.

Platform-as-a-Service («платформа як сервіс»). Користувачеві надається комп'ютерна платформа з встановленою операційною системою і певним програмним забезпеченням.

Integration-as-a-Service («інтеграція як сервіс») - це можливість отримувати з «хмари» повний інтеграційний пакет, включаючи програмні інтерфейси між додатками і управління їх алгоритмами. Сюди входять відомі послуги та функції пакетів централізації, оптимізації та інтеграції корпоративних додатків (EAI), але вони надаються як «хмарний» сервіс.

Security-as-a-Service («безпека як сервіс»), даний вид послуги надає можливість користувачам швидко розгортати продукти, що вимагають безпечного використання веб-технологій, електронного листування, локальної мережі. Користувачі даного сервісу мають змогу економити на розгортанні та підтримці своєї власної системи безпеки.

Management/Governace-as-a-Service («адміністрування та управління як сервіс»), надає можливість керувати і задавати параметри роботи одного або багатьох «хмарних» сервісів. Це в основному такі параметри, як топологія, використання ресурсів, віртуалізація.

Infrastructure-as-a-Service («інфраструктура як сервіс»). Користувачеві надається комп'ютерна інфраструктура, зазвичай віртуальні платформи (комп'ютери), пов'язані в мережу, які він самостійно налаштовує під власні цілі.

Testing-as-a-Service («тестування як сервіс») – дає можливість тестування локальних або «хмарних» систем з використанням тестового ПЗ з «хмари» (при цьому жодного устаткування або забезпечення на підприємстві, не потрібно).

Для наочності, узагальнимо сервіси архітектури «хмара», в одну схему на якій наведено класифікацію сервісів, за типом послуг (рис.4.1).

Рис. 4.1. «Хмарні» сервіси

Тобто, послуга програмного забезпечення з оплатою через використання. Сервіс включає в себе службу мережі (NaaS),

програмне забезпечення як послугу (SaaS),

платформу як послугу (PaaS)

інфраструктуру як послугу (IaaS). Кожна модель надає все більше і більше хмарних сервісів від постачальників. Ці сервісні пропозиції - додана вартість хмарних обчислень. Як мінімум, ці послуги повинні компенсувати капітальні витрати, з якими зіштовхується клієнт для придбання і обслуговування такого обладнання центру обробки даних, і врахувати це як експлуатаційні витрати.

На рис. 4.2 показані відмінності в управлінні хмарними моделями, які пропонує компанія Microsoft для свого хмарного середовища Azure, що будуть описані в подальшому [3].

NaaS включає такі сервіси, як SDP і SDN. IaaS підштовхує апаратні системи і сховище до хмари. PaaS включає в себе інфраструктуру, але також управляє операційною системою і часом виконання системи або

контейнерами у хмарі. Нарешті, SaaS підштовхує всі сервіси, в інфраструктуру і сервіси до хмарного провайдера.

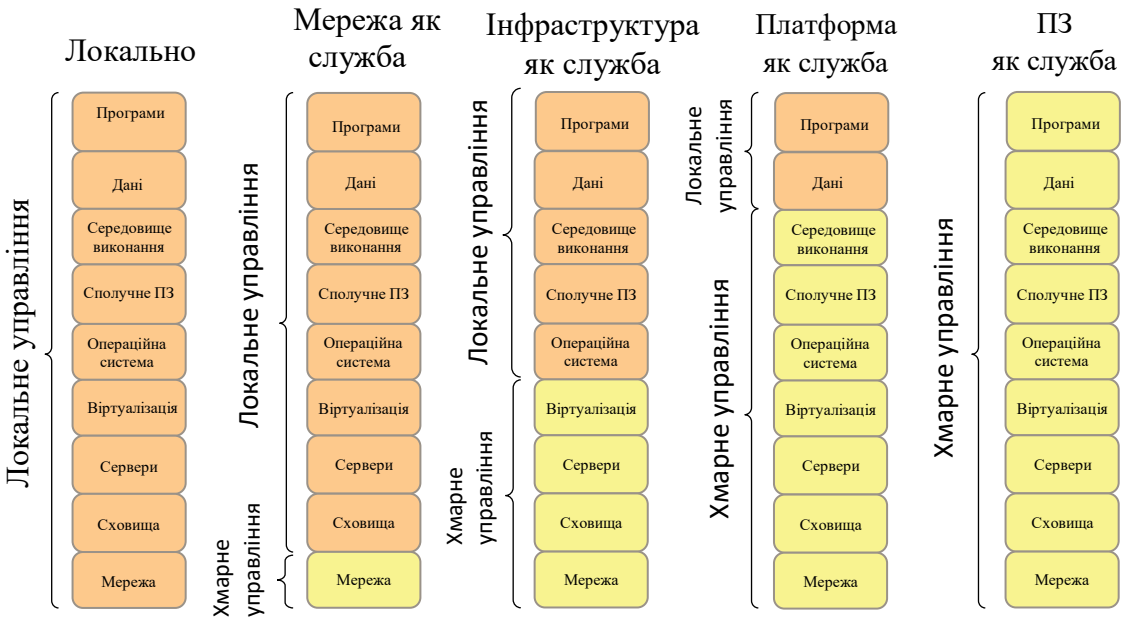


Рис. 4.2. Моделі хмарної архітектури для Microsoft хмарного середовища Azure.

Власний об'єкт – це той, де управління всіма службами, інфраструктурою і сховищем здійснюється власником.

NaaS. Для NaaS характерні такі сервіси, як Мережева взаємодія, визначене ПЗ (SDN) і Програмно-визначені периметри (SDP). Ці продукти є керованими хмарами і організованими механізмами для забезпечення оверлейних мереж і безпеки підприємств. Замість того, щоб створювати глобальну інфраструктуру і виділяти капітал для підтримки корпоративних комунікацій, при створенні віртуальної мережі може використовуватися хмарний підхід. Це дозволяє мережі оптимально масштабувати ресурси в бік збільшення або зменшення в залежності від потреб, а нові мережеві якості можуть бути придбані і розгорнуті швидко.

SaaS. SaaS є основою хмарних обчислень. У провайдера зазвичай є пропоновані додатки або послуги, які пропонуються кінцевим користувачам за допомогою таких клієнтів, як мобільні пристрої, тонкі клієнти або фреймворки в інших хмарах. З точки зору користувача, віртуальний SaaS фактично працює на клієнті користувача. Ця абстракція програмного забезпечення дозволила галузі домогтися значного зростання в хмарному сервісі. Сервіси SaaS працюють для таких пристроїв, як Google Apps, Salesforce і Microsoft Offce 365.

PaaS. PaaS використовує базове устаткування і програмні засоби нижнього рівня, що надаються хмарою. В такому випадку кінцевий користувач просто використовує апаратне забезпечення центру обробки даних, операційну систему, проміжне ПЗ і різні бази даних постачальника для розміщення свого приватного додатка або сервісів. Проміжне ПЗ може складатися з систем баз даних. При побудові багатьох галузей промисловості було використано обладнання хмарних постачальників, наприклад для Swedbank, Trek Bicycles і Toshiba. Прикладами публічних постачальників PaaS є IBM Bluemix, Google App Engine і Microsoft Azure [12].

Різниця між PaaS і IaaS полягає в тому, що ви отримуєте переваги масштабованості і OPEX (скор. від англ. operating expenses, операційні витрати) – повсякденні витрати компанії для ведення бізнесу, виробництва товарів і послуг.

Сума операційних витрат і капітальних вкладень (англ. CAPEX) складають витрати компанії, які не включаються в пряму собівартість продуктів або послуг, які пропонує ринку дана компанія. Наприклад, покупка копіювального апарату відноситься до капітальних вкладень, а покупка паперу, тонера, оплата споживаної ним електроенергії, ремонту та обслуговування цього пристрою відносяться до операційних витрат. Стосовно до бізнесу операційні витрати включають в себе, зокрема, оплату оренди приміщень для офісу, комунальних платежів компанії, витрати на рекламу, ліцензійні та страхові платежі, витрати на відрядження та транспортні витрати, оплату сторонніх адвокатів і аудиторів, зарплату персоналу і т.д. Операційні витрати (повсякденні витрати компанії на організацію продажів, адміністрування т.д.) протиставляються прямим витратам - витратам компанії на безпосереднє виробництво продуктів і послуг. Іншими словами, прямі витрати – це сума грошей, які компанія витрачає на перетворення сировини або комплектуючих у готову продукцію. У звіті про фінансові результати операційні витрати вказуються в прив'язці до періоду часу, протягом якого вони були понесені – місяць, квартал або рік. Операційні витрати також пов’язані з хмарною інфраструктурою, але у вас також буде перевірене проміжне ПЗ і операційні системи від провайдера. Це такі системи, як Docker, де програмне забезпечення розгортається в контейнерах. Якщо ваш додаток розгортається в межах обмежень, що надається постачальником інфраструктури, ви можете очікувати більш швидкий вихід на ринок, оскільки більшість компонентів, ОС і проміжного програмного забезпечення гарантовано будуть доступні.

IaaS. IaaS була початковою концепцією хмарних сервісів. У цій моделі постачальник створює масштабовані апаратні служби в хмарі і

надає модифікацію програмних фреймворків для створення клієнтських віртуальних машин. Це забезпечує максимальну гнучкість при розгортанні, але вимагає більших зусиль з боку клієнта.

4.1.3. Хмарна архітектура OpenStack

OpenStack - це сервер Apache 2.0 з відкритим вихідним кодом, який використовується для створення хмарних платформ. Це IaaS розробляється спільнотою розробників з 2010 р OpenStack Foundation управляє програмним забезпеченням і підтримує більше 500 компаній, включаючи Intel, IBM, Red Hat і Ericsson. Ми будемо використовувати OpenStack в якості еталонної архітектури для інших постачальників хмарних обчислень, оскільки велика частина компонентів і термінологія також використовуються в комерційних хмарах [1].

OpenStack починався як спільний проект NASA і Rackspace в 2010 р Архітектура має всі основні компоненти інших хмарних систем, включаючи обчислення і балансування навантаження; компоненти зберігання, включаючи резервне копіювання і відновлення; мережеві компоненти, інформаційні панелі, системи безпеки та ідентифікації, пакети даних і аналітики, інструменти розгортання, монітори, лічильники і додатки. Це ті компоненти, які буде використовувати архітектор при виборі хмарного сервісу.

З точки зору архітектури, OpenStack являє собою змішані шари компонентів. Основна форма хмари OpenStack показана на рис. 4.3.

Кожен сервіс має певну функцію і унікальне ім'я (наприклад, Nova). Система працює, в цілому, надаючи масштабовані функціональні можливості хмарного класу масштабу корпорації.

Всі комунікації в компонентах OpenStack виконуються через **протокол розширеної черги повідомлень (AMQP)**, зокрема, RabbitMQ або Qpid.

Повідомлення можуть бути або неблокуючими, або блокуючими в залежності від того, як було відправлено повідомлення. Повідомлення буде відправлено як об'єкт JSON в RabbitMQ, і одержувачі отримають свої повідомлення в одному сервісі. Це метод зв'язку (**Remote Procedure Call - RPC**) між основними підсистемами. *Перевага хмарного середовища полягає в тому*, що проблеми клієнта і сервера повністю незалежні один від одного, і це дозволяє серверам динамічно масштабуватися в бік збільшення або зменшення. Повідомлення не передаються, а направляються, що знижує трафік до мінімуму. Нагадаємо, що AMQP - це стандартний протокол обміну повідомленнями, який використовується в просторі IoT.

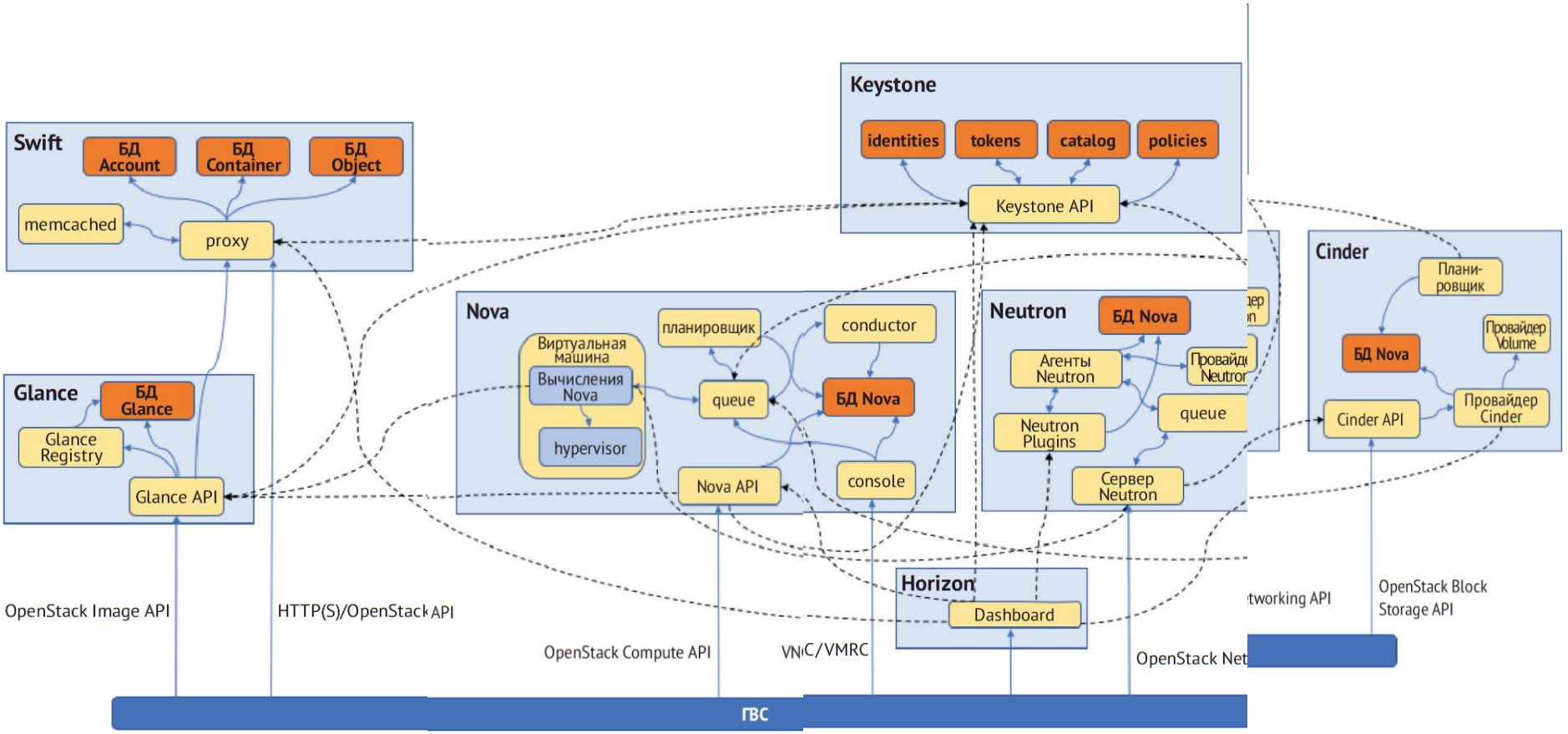


Рис. 4.3 – Високорівнева архітектурна діаграма OpenStack [1]

Keystone - управління ідентифікацією та обслуговуванням

Keystone - це служба управління ідентифікаторами хмари OpenStack. Менеджер ідентифікації встановлює облікові дані користувача і, авторизацію входу. Це, по суті, відправна точка або точка входу в хмару.

Цей ресурс буде підтримувати центральний каталог користувачів і їх прав доступу. Це найвищий рівень безпеки, що забезпечує незалежність і безпеку для користувача середовищ. Keystone може взаємодіяти з такими сервісами, як LDAP на корпоративному рівні. Keystone також підтримує базу даних токенів і надає часові маркери користувачам аналогічно тому, як **Amazon Web Services (AWS)** встановлює облікові дані. Реєстр служб використовується для запиту продуктів або послуг, доступних користувачеві програмно.

Glance - сервіс зображень

Glance - це серцевина управління віртуальними машинами для OpenStack. Більшість хмарних сервісів забезпечить певний рівень віртуалізації і матиме аналоговий ресурс, подібний Glance.API служби зображень це служба RESTful, що дозволяє клієнту розробляти шаблони VM, виявляти доступні віртуальні машини, клонувати зображення на інші сервери, реєструвати віртуальні машини і навіть безперешкодно переміщати працюючі віртуальні машини на різні фізичні сервери без перерви в роботі. Glance викликає Swift (сховище об'єктів) для з витягання або зберігання різних зображень. Glance підтримує різні стилі віртуальних образів:

raw - неструктуровані зображення;

vhd - VMWare, Xen, OracleVirtualBox;

vmdk - загальний формат диска;

vdi - зображення емулятора QEMU;

iso - зображення на оптичному диску (CD-ROM);

aki / ari / ami - зображення Amazon.

Віртуальна машина складається з усього вмісту образу жорсткого диска, включаючи гостьові операційні системи, середовища виконання, додатки та служби.

Обчислення Nova

Це основа служби управління обчислювальними ресурсами OpenStack. Її мета - визначити і врахувати обчислювальні ресурси на основі попиту. Вона також несе відповідальність за управління системним гіпервізором і віртуальними машинами. Nova може працювати з декількома віртуальними машинами, наприклад з VMware або Xen, або може управляти контейнерами.

Масштабування на вимогу є невід'ємною частиною будь-якої хмарної пропозиції.

Nova заснована на API веб-служби RESTful для спрощення управління.

База даних Nova необхідна для підтримки поточного стану всіх об'єктів в кластері. Наприклад, кілька станів різних серверів в кластері можуть бути наступними:

ACTIVE - сервер активно працює;

BUILD - сервер в стані збірки і поки не закінчений;

DELETED - сервер був вилучений;

MIGRATING - сервер переноситься на інший хост.

Nova покладається на планувальник, щоб визначити, яке завдання виконати і де її виконати. Планувальник може випадково асоціювати спорідненість або використовувати фільтри, щоб вибрати набір хостів, які найкращим чином відповідають деяким набором параметрів. Кінцевим продуктом фільтра буде впорядкований список хост-серверів для використання від кращого до гіршого (несумісні хости будуть вилучені зі списку).

OpenStack має багатий набір фільтрів, що дозволяють налаштовувати розподіл серверів і сервісів. Це дозволяє дуже чітко контролювати забезпечення і масштабування сервера. Це класичний і дуже важливий аспект хмарного дизайну. Такі фільтри включають, але не обмежуються такими характеристиками, як: розмір оперативної пам'яті; ємність і тип диска; рівні IOPS; використання процесора; групова спорідненість; спорідненість з CIDR.

Swift - зберігання об'єктів

Swift надає резервну систему зберігання для центру обробки даних OpenStack. Swift дозволяє масштабувати кластери шляхом додавання нових серверів. Сховище об'єктів буде містити такі речі, як облікові записи та контейнери. Віртуальна машина користувача може зберігатися або кешуватися в Swift. Обчислювальний вузол Nova може викликати безпосередньо Swift і завантажувати зображення при першому запуску.

Neutron - мережеві сервіси

Neutron - це управління мережею OpenStack і служба VLAN. Вся мережа може настроюватися і надає такі послуги, як: доменні служби імен; DHCP - протокол динамічної конфігурації хостів; функції шлюзу; управління VLAN; з'єднання на другому рівні моделі OSI; SDN; протоколи з покриттям і тунелюванням; VPN; NAT (SNAT і DNAT); системи виявлення вторгнень; балансування навантаження; брандмауери.

Cinder - блочне сховище

Cinder забезпечує OpenStack постійними службами зберігання блоків, необхідними для хмари. Він виступає в ролі *сховища як служби* для використання з базами даних, динамічними файловими системами і

в інших випадках, де важливим є захист від витоків даних. Це важливо і для потокових сценаріїв IoT. Як і інші компоненти OpenStack, система зберігання така ж динамічна і масштабується в міру необхідності. Архітектура побудована на принципах високої доступності і відкритих стандартах.

Функціональність, що надається Cinder, включає:

створення, видалення і прив'язку пристроїв зберігання до примірників Nova;

сумісність з декількома сховищами (HP 3PAR, EMC, IBM, Ceph, CloudByte, Scality);

підтримку декількох інтерфейсів (Fibre Channel, NFS, Shared SAS, IBM GPFS, iSCSI);

резервне копіювання і витягання образів дисків;

збереження зображень в певні моменти часу;

альтернативне сховище для зображень VM.

Horizon

Horizon - це панель інструментів OpenStack. Це спрощений вид в OpenStack для клієнта. Він забезпечує веб-подання різних компонентів, які включають OpenStack (Nova, Cinder, Neutron і інші). Horizon являє собою зображення призначеного для користувача інтерфейсу хмарної системи як альтернативний засіб поверх API. Horizon розширюємий, тому третя сторона може додавати свої віджети або інструменти в панель інструментів. Можна, додати новий компонент білінгу, і потім для клієнтів може бути створений відповідний елемент панелі Horizon.

Більшість систем IoT, які використовують хмарні обчислення, матимуть деяку форму панелі моніторингу з аналогічними функціями.

Heat - оркестрація (опція)

Heat може запускати кілька складових хмарних додатків і управляти хмарною інфраструктурою на основі шаблонів в екземплярі OpenStack. Heat інтегрується з технологіями телеметрії для автоматичної настройки системи відповідно з навантаженням. Шаблони в Heat намагаються відповідати форматам AWS CloudFormation, а відносини між ресурсами можуть бути вказані аналогічним чином (наприклад, даний том підключений до даного сервера).

Ceilometer - телеметрія (опція)

OpenStack надає додатковий сервіс під назвою Ceilometer, який може використовуватися для збору даних телеметрії і обліку ресурсів, використовуваних кожною службою. Вимірювання використовується для збору інформації про використання і перетворення її в рахунки клієнта. Ceilometer також надає інструменти оцінки і виставлення рахунків. Значення виставленої вартості конвертується в еквівалентну валюту, а білінг використовується для початку процесу оплати.

Ceilometer контролює і вимірює різні події, такі як запуск служби, додавання тому і зупинка примірника. Метрики збираються по використанню ЦПУ, кількості ядер, використання пам'яті і переміщенню даних. Все це збирається і зберігається в базі даних MongoDB.

4.1.4. Обмеження хмарних архітектур для IoT

Постачальник хмарних сервісів знаходиться за межами граничного пристрою IoT і керує глобальною мережею. Однією з особливостей архітектури IoT є те, що пристрої PAN і WPAN можуть не відповідати протоколу IP. Протоколи, такі як Bluetooth Low Energy і Zigbee, не засновані на IP, тоді як всі в глобальних мережах, включаючи хмари, засновані на IP. Таким чином, роль прикордонного шлюзу полягає у виконанні перекладу з одного протоколу в інший (рис. 4.4).

Рис. 4.4. Ефекти затримок в хмарі. Реакція в реальному часі має вирішальне значення в багатьох додатках IoT і змушує пересувати обробку ближче до кінцевого пристрою [1]

Ефект затримки

Іншим ефектом є *час очікування* і *час відгуку* для подій. У міру наближення до датчика ви входите в область, де діють жорсткі вимоги виконання в реальному часі. Ці системи, як правило, являють собою глибоко вбудовані системи або мікроконтролери з малою затримкою,

призначені для реальних подій. Наприклад, відеокамера чутлива до частоти кадрів (як правило, 30 або 60 кадрів в секунду) і повинна виконувати ряд послідовних завдань в конвеєрі потоку даних (позбавлення від мозаїки, позначення, баланс білого і гамма-регулювання, відображення гами, масштабування і стиснення). Обсяг даних, що проходять через конвеєр відеозображення (відео 1080p з використанням 8 біт на канал зі швидкістю 60 кадрів в секунду) складає приблизно 1,5 ГБ/с. Кожен кадр повинен проходити через цей конвеєр в режимі реального часу, тому більшість процесорів сигналів відеозображення використовують для цих перетворень контролери. Якщо ми перемістимося вгору по стеку, шлюз буде мати кращий час відгуку і зазвичай реагує за мілісекунди з однією цифрою. Коефіцієнт стробування в часі відгуку - це *латентність* WPAN і навантаження на шлюз. Більшість WPAN, таких як BLE, є змінними і залежать від кількості пристроїв BLE під шлюзом, інтервалів сканування, інтервалів реклами і т.д. Інтервали з'єднання BLE можуть досягати 7,5 мс, але можуть варіюватися в залежно від того, як клієнт налаштовує інтервали реклами, щоб звести до мінімуму споживання енергії. Сигнали Wi-Fi зазвичай мають затримку 1,5 мс. Затримка такого рівня вимагає фізичного інтерфейсу з PAN. Не можна очікувати, що при передачі необроблених пакетів BLE в хмару, буде швидкодія майже в реальному часі.

Компонент обробки в хмарі додає ще одну ступінь затримки в WAN. Маршрут між шлюзом і провайдером хмарних обчислень може пролягати декількома шляхами на основі розташування центрів обробки даних і шлюзу. Хмарні провайдери зазвичай надають набір регіональних центрів обробки даних для нормалізації трафіку.

У дослідженні “Вимірювання широкосмугової Америки” (опубліковане наприкінці 2018 року) повідомляється про типові затримки підключення до Інтернету для поширених форм широкосмугових послуг США:

- Волоконно-оптичний: 12-20 мс.
- Кабельний Інтернет: 15-34 мс.
- DSL: 25-80 мс.
- Супутниковий Інтернет: 594-612 мс.

Більш точний аналіз часу затримки для хмар і часу відгуку можна отримати за допомогою CL Audit. Існують і інші інструменти для аналізу латентності, такі як Fathom і SmokePing. Ці сайти досліджують, моніторять і зберігають затримку TCP, HTTP і SQL при роботі з базою даних в AWS і Microsoft Azure на щоденній основі в багатьох регіонах світу. Це забезпечує найкращу видимість загального впливу затримки, яку можна очікувати від хмарного рішення.

Як правило, затримки у хмарі становитимуть десятки, якщо не сотні мілісекунд, без урахування будь-яких накладних витрат на обробку даних, що надходять. І це повинно бути взято до уваги для різних рівнів відповіді при побудові хмарної архітектури для IoT. Архітектури близьких пристроїв допускають відповіді до 10 мс, а також вони повторювані і детерміновані. Хмарні рішення можуть мати мінливий час відгуку, а також час затримки на порядок більше, ніж при використанні граничних пристроїв. Архітектор повинен враховувати, де розгорнути частину рішення на основі розгляду цих двох ефектів. Постачальників хмарних обчислень також слід вибирати на основі моделей розгортання центрів обробки даних. Якщо рішення IoT розгортається у всьому світі або, можливо, буде розширюватися для охоплення декількох регіонів, хмарний сервіс повинен мати центри обробки даних, розташовані в географічно близьких областях, щоб допомогти в зменшенні часу відгуку.

Таким чином, метою IoT є отримання даних від датчика, їх аналіз та прийняття правильних рішень. У разі коли відбувається масштабування до тисяч, мільйонів і потенційно мілліардів датчиків, які передають і передають дані без зупинок, необхідно впроваджувати передові інструменти для отримання, зберігання, передачі, аналізу та прогнозування значень з цього моря даних. Хмарні обчислення - один з елементів, що дозволяють використовувати цю послугу у вигляді кластерів, що масштабується обладнання, та програмного забезпечення. Туманні обчислення роблять хмарну обробку ближче до краю для вирішення проблем із затримкою, безпекою та витратами на зв'язок.

4.2. Туманні технології в IoT

Туманні обчислення — це модель, яка забезпечує обчислення та зберігання даних між кінцевими пристроями та традиційними центрами хмарних обчислень. Туманні обчислення — концепція за якою частина даних оброблєються в локальних мережах, а не виключно в дата-ценрі. Будь-який пристрій, що має обчислювальні здібності, сховище та мережу підключення, може бути вузлом туману. Приклади пристроїв включають промислові контролери, комутатори, маршрутизатори, вбудовані сервери та камери відеоспостереження [1].

У 2011 році виникла потреба у розширенні хмарних обчислень за допомогою туманних обчислень, щоб впоратися з величезною кількістю пристроїв IoT та великими обсягами даних для реальних програм із низькою затримкою.

19 листопада 2015 року Cisco Systems, ARM Holdings, Dell, Intel, Microsoft та Princeton University створили консорціум OpenFog для просування інтересів та розвитку в туманних обчислень.

Основні харектеристики туманних обчислень:

Проінформаваність про місцеположення пристрою.

Низька затримка. Оскільки туманні обчислення знаходиться ближче до кінцевих пристроїв, вони забезпечують меншу затримку при обробці даних кінцевих пристроїв.

Мобільність. Для багатьох програм Fog важливо спілкуватися безпосередньо з мобільними пристроями, і підтримках відповідних протоколів (наприклад, LISP).[1]

Взаємодія в режимі реального часу. Програми для туманних обчислень забезпечують взаємодію між вузлами у режимі реального часу, а не пакетної обробки, що використовується в хмарних обчисленнях.

Сумісність. Вузли туманних обчислень можуть взаємодіяти та працювати з різними доменами та постачальниками послуг.

Переваги туманних обчислень

Контроль конфіденційності. Завдяки туманним обчисленням ви можете краще контролювати рівень конфіденційності. Ви можете обробляти та аналізувати чутливі дані локально, а не надсилати їх до централізованої хмари для аналізу.

Підвищення продуктивності. Туманні обчислення допомагають підвищити продуктивність і збільшити швидкість бізнес-процесів. Вони можуть дозволити пошук лише тих даних, які потребують негайної взаємодії людини, а не всіх даних, скорочуючи час і зусилля, необхідні для пошуку потенційних проблем.

Безпека даних. Туманні обчислення дозволяють під'єднати до мережі кілька пристроїв. Замість одного централізованого місця, яке може стати вразливим, діяльність відбувається між різними локальними кінцевими точками, що полегшує ідентифікацію загроз, таких як заражені файли, потенційні хаки або зловмисне програмне забезпечення.

Недоліки туманних обчислень

Складність. Система, яка включає в себе безліч пристроїв, які зберігають і аналізують власні дані, які можуть бути розташовані в будь-якому місці, в будь-який час, додає складності мережі, яка колись надсилала всі свої дані до централізованого місця.

Ризики. Збільшення кількості з'єднань означає збільшення ризиків. Завдяки численним пристроям і численним користувачам, ризик виходу зіпсованих або заражених файлів, додатків або інформації в основний потік даних компанії різко зростає.

Локальні дані. Використовуючи туманні обчислення, великий об'єм даних зберігається на самих пристроях. Ці пристрої часто знаходяться за межами фізичного місцезнаходження офісу, і багато компаній або менеджерів підприємств вважають, що ця конфігурація збільшує ризик порушення даних

4.2.1. Особливості туманних обчислень

Туманні обчислення - це еволюційне розширення хмарних обчислень на краю. У цьому питанні описується різниця між обчисленнями Fog і Edge і представлені різні топології і архітектурні посилання для Fog Computing [1].

Філософія Hadoop для туманних обчислень

Туманні обчислення розвинули успіх Hadoop і MapReduce, і для того, що б краще зрозуміти важливість Fog Computing, варто розглянути, як працює Hadoop. **MapReduce** - це метод зіставлення, а **Hadoop** - це платформа з відкритим вихідним кодом, заснована на алгоритмі MapReduce.

MapReduce включає три етапи: відображення, перетасування і зменшення. На фазі відображення обчислювальні функції працюють з локальними даними. Крок перетасовки перерозподіляє дані в міру необхідності. Це критичний крок, коли система намагається поєднати всі залежні дані в одному вузлі. Останнім кроком є фаза зменшення, при якій обробка відбувається паралельно на всіх вузлах.

Загальний висновок тут полягає в тому, що MapReduce намагається наблизити обробку туди, де знаходяться дані, і не переміщати дані туди, де знаходяться процесори. Ця схема ефективно усуває перевантаження комунікацій і природне вузьке місце в системах з надзвичайно великими структурованими або неструктурованими наборами даних. Ця парадигма також може бути застосована і до IoT. У просторі IoT дані (можливо, дуже великий обсяг даних) створюються в реальному часі в вигляді потоку даних. Це дані великого обсягу в разі IoT. Це не статичні дані, такі як база даних або кластерне сховище Google, а нескінченний потік даних з усіх куточків світу. Туманні конструкції - природний спосіб вирішити цю нову проблему з великими даними.

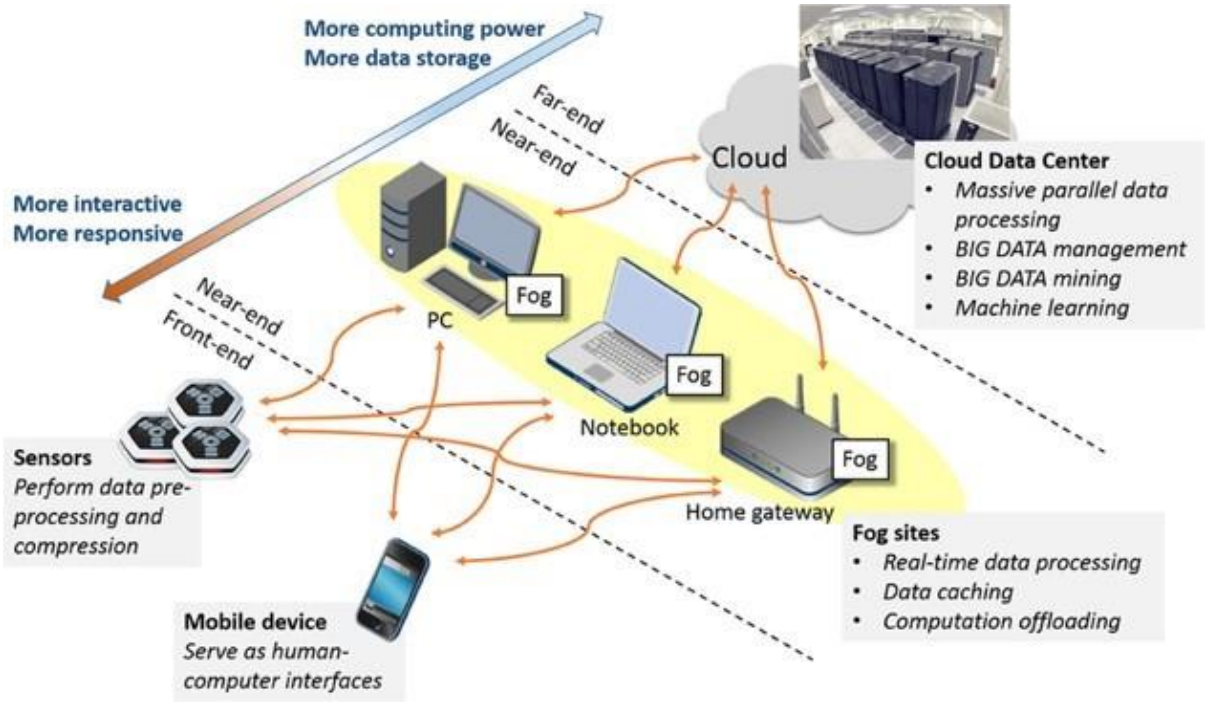


Рис. 4.5. Порівняння туманних, граничних і хмарних обчислень

Порівняння туманних, граничних і хмарних обчислень

Ми вже визначили граничні обчислення як обробку, що переміщається близько до того місця, де генеруються дані. У разі IoT граничним пристроєм може бути сам датчик з невеликим мікроконтролером або вбудованою системою, здатною до WAN-зв'язку. В інших випадках межа буде шлюзом в архітектурі з особливо обмеженими кінцевими точками, що змушують шлюз зависати. Гранична обробка також зазвичай згадується в контексті «машина-машина», де існує щільна кореляція між краєм (клієнтом) і сервером, розташованим в іншому місці.

Граничні обчислення існують, як зазначено, для усунення проблем із затримкою і непотрібним завантаженням смуги пропускання, а також для додавання таких сервісів, як денатурація і безпека, близьким до джерела даних. У граничного пристрою може бути зв'язок з хмарним сервісом ціною затримки і несучої; він не бере активної участі в хмарній інфраструктурі.

Хмарні обчислення трохи відрізняються від парадигми граничних обчислень. У хмарних обчисленнях в першу чергу розділяється API інфраструктури та стандарти зв'язку з іншими туманними вузлами і/або хмарною службою. Туманні вузли є розширеннями хмари, тоді як граничні пристрої можуть використовувати або не використовувати хмару.

Іншим ключовим принципом туманних обчислень є те, що туман може існувати в ієрархічних шарах. Туманні обчислення також можуть балансувати навантаження і керувати даними зі сходу на захід і з

півночі на південь, щоб допомогти в балансуванні ресурсів. З визначення хмари і його сервісів, які були описані в попередньому розділі, можна подумати про ці туманних вузлах як просто про ще одні (хоча і менш потужні) інфраструктури в гібридній хмарі.

4.2.2. Архітектура OpenFog RA

Архітектура туманного фреймворка, такого як хмарний фреймворк, необхідна для розуміння міжмережевої взаємодії і контрактів за даними між різними рівнями. Тут ми розглянемо OpenFog Consortium Reference Architecture [1].

Консорціум OpenFog є некомерційною галузевої групою, зайнятою визначенням стандартів сумісності з туманними обчисленнями. Хоча вони не є органом по стандартизації, але впливають на напрямок діяльності інших організацій за допомогою взаємодії і впливу в галузі. Довідкова архітектура OpenFog - це модель, яка допомагає архітекторам і бізнес-лідерам в створенні устаткування, програмного забезпечення і придбання інфраструктури для туманних обчислень. OpenFog розуміє переваги хмарних рішень і бажання розширити цей рівень обчислень, зберігання, мереж і масштабованості до граничного рівня, не жертвуючи затримками і пропускною спроможністю.

OpenFog RA керується набором основних принципів, які називаються в термінології OpenFog опорами, або стовпами (рис. 4.6).

Рис. 4.6. Опори (стовпи) OpenFog RA

Вони являють собою ключові атрибути, які система повинна включати, щоб відповідати визначенню OpenFog горизонтальної системної архітектури, яка забезпечує розподіл функцій обчислення, зберігання, управління та мережі ближче до джерела даних (користувачам, речам та ін.) вздовж безперервного континууму від хмари до речам. У наступних розділах коротко описується кожен стовп архітектури.

Безпека. Багато IoT-додатків, що підтримуються OpenFog RA, є критичними з точки зору збереження конфіденційності, важливості завдань і навіть життєзабезпечення. Таким чином, будь-яка компрометація безпеки в туманній мережі може мати серйозні наслідки.

Безпека в OpenFog RA не може бути одним рішенням, що підходить для всієї архітектури. Швидше вона описує всі механізми, які можуть бути використані, щоб зробити туманний вузол безпечним, - від кремнію до коду.

Реалізація безпеки має багато різних описів та атрибутів, таких як конфіденційність, анонімність, цілісність, довірливість, атестація, верифікація та вимірювання.

Стовп безпеки OpenFog RA починається з чіткого визначення базових будівельних блоків. Усі туманні вузли повинні використовувати апаратно-базовану незмінну основу довіри. Ця апаратна основа довіри є перевіреним надійним апаратним компонентом, який отримує керування під час увімкнення живлення. Потім ланцюжок довіри розширюється інші апаратні, апаратно-програмні і програмні компоненти.

Внаслідок близькості до кордону вузли туманної мережі часто є першими точками керування доступом та шифруванням. Це означає, що вони повинні забезпечити контекстуальну цілісність, ізоляцію та управління агрегуванням чутливих даних, перш ніж ті покинуть кордон.

Масштабованість. Можливість масштабованості враховує динамічні, технічні та бізнес-потреби, пов'язані з розгортанням туманних обчислень. Ієрархічні властивості та локалізація вузлів на логічних краях мереж надають додаткові можливості для масштабування. Так, індивідуальні вузли можуть масштабуватись внутрішньо за допомогою додавання апаратних або програмних компонентів. Туманна мережа може збільшувати або зменшувати масштаб шляхом додавання або вилучення вузлів або одному рівні ієрархії, або на суміжних рівнях. Системи зберігання, мережна зв'язність та аналітичні послуги можуть масштабуватися разом із туманною інфраструктурою.

Масштабування може торкатися багатьох аспектів функціонування туманної мережі. Наприклад, продуктивність, ємність, надійність, безпека, апаратні та програмні засоби.

Масштабованість дозволяє забезпечити базову підтримку для задоволення бізнес-вимог і дозволяє використовувати модель «плати в міру зростання», що суттєво з погляду економії коштів на початковому етапі розгортання.

Відкритість. Відкритість є суттєвою для успіху широкого розгортання екосистеми туманних обчислень для IoT та додатків. Патентовані рішення або продукти від одного виробника можуть призвести до обмеження диверсифікації, що негативно вплине на вартість, якість та інноваційність.

Відкритість як фундаментальний принцип дозволяє туманним вузлам існувати всюди в мережі. Вона дозволяє об'єднувати ресурси у вигляді виявлення. Це означає, що нові програмно-визначені вузли можуть бути динамічно створені для вирішення бізнес-завдань.

Автономність. Операційна автономність дозволяє доставити необхідну функціональність у разі відмови зовнішніх сервісів і має бути забезпечена на всіх рівнях ієрархії. Автономність на межі мережі означає, що потрібна функціональність забезпечується локальними пристроями. Прийняття рішень буде виконуватися на всіх рівнях ієрархії, і вже не потрібно, щоб рішення приймалися лише у хмарі. OpenFog підтримує автономність широкого ряду функцій і за своєю природою не спирається на централізовану сутність для функціонування. Автономність забезпечується в таких областях, як виявлення, оркестрування та управління, безпека та функціонування.

Програмуємість. Ця властивість забезпечує високо адаптивне розгортання, у якому повторна постановка завдання туманний вузол чи шар (туманний кластер) задля забезпечення оперативної динаміки може бути повністю автоматизована. Повторна постановка задачі може бути виконана за допомогою вбудованих у туманний кластер інтерфейсів програмованості або за допомогою таких у туманних кластерах з більш високим порядком ієрархії за допомогою внутрішніх та міжвузлових комунікацій. Еталонна архітектура OpenFog підтримує різні сценарії розгортання з використанням стандартів, технологій, API, фреймворків і контейнерів часу виконання, тому рішення для конкретного домену складається з компонентів, запропонованих в еталонній архітектурі.

Надійність, доступність, обслуговування (RAS). RAS реалізується у всіх успішних системних архітектурах і, таким чином, набуває великого значення і в архітектурі OpenFog. Апаратне забезпечення, програмне забезпечення та мережна ієрархія є трьома основними

областями системи RAS. Надійне розгортання дозволить продовжувати постачати розроблені функції як за нормальних, так і за несприятливих умов експлуатації. Надійність апаратних засобів, ПЗ та мережевих пристроїв утворюють базис для доступності та обслуговуваності.

RAS особливо важлива для різних розгортань OpenFog у суворих умовах довкілля. Ось чому це важлива особливість архітектури OpenFog та чому аспекти RAS можна знайти у всіх реалізаціях OpenFog.

Маневреність. Маневреність є ключовим елементом, який дозволяє швидко приймати правильні бізнес-рішення на основі туманних обчислень. Прогнозовані обсяги даних, що генеруються IoT, означають, що малоймовірно, що люди самостійно зможуть їх зрозуміти, отримати знання та прийняти правильні рішення про те, як найкраще використати свої IoT-активи на користь свого бізнесу. В архітектурі OpenFog принцип маневреності полягає в тому, щоб цінні дані, що генеруються в IoT, могли бути швидко перетворені на ефективні ідеї, які сприяють швидким рішенням та додатковим рівням автоматизації для підтримки бізнес-інтересів. Швидкі тактичні рішення у відповідь повинні прийматися настільки близько до кордону, наскільки це можливо. Більш стратегічні, системні рішення та управління політиками робляться на вищих рівнях ієрархії.

Ієрархічність. Обчислювальні ресурси OpenFog можна як логічну ієрархію, що базується на функціональних вимогах IoT-систем. Залежно від масштабу та природи області проблем, які мають бути вирішені, ієрархія може складатися з мережі з'єднаних інтелектуальних секційних систем, розподілених за фізичними чи логічними рівнями, або може бути реалізована як єдина фізична система, масштаб якої визначається складністю проблеми. Очікується, що більшість реальних систем будуть базуватися на комбінації туманних та хмарних обчислень, тоді як меншість систем може бути або туманною, або хмарною.

OpenFog Reference Architecture – це багатошаровий підхід від граничних датчиків і виконавчих механізмів внизу, до додатків нагорі. Архітектура має деяку схожість з типовою хмарною архітектурою, такою як OpenStack, але розширює її, оскільки вона більш схожа на PaaS, ніж на IaaS. В цьому випадку OpenFog надає повний стек і, як правило, апаратно незалежний або, по крайній мірі, абстрагує платформу від іншої частини системи (рис. 4.7).

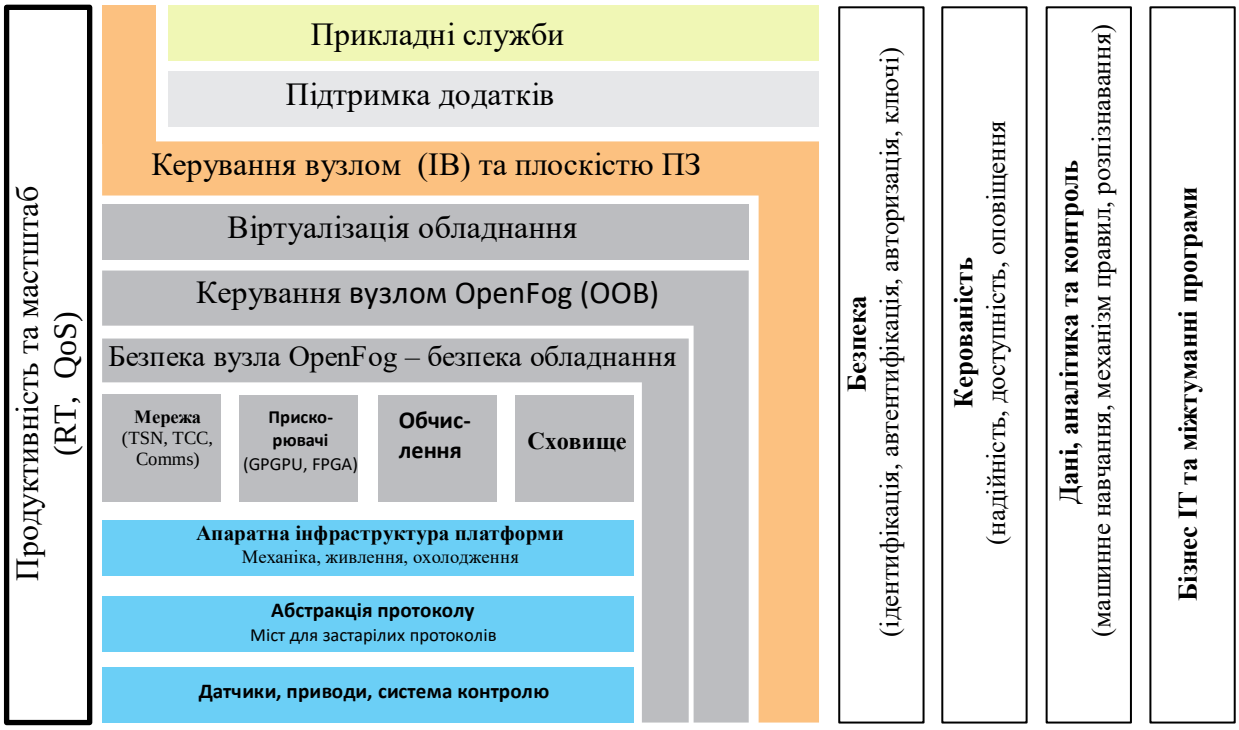


Рис. 4.7. Архітектура OpenFog RA

Служби додатків

Роль сервісного рівня полягає в наданні «віконного скла» і спеціальних послуг, необхідних для виконання завдання. Це включає в себе надання з'єднувачів для інших служб, зміст пакетів для аналітики даних, надання користувальницького інтерфейсу при необхідності і надання основних послуг.

З'єднувачі на прикладному рівні з'єднують служби з рівнем підтримки. Рівень абстракції протоколу забезпечує шлях для спілкування з'єднувача безпосередньо з датчиком. Кожна послуга повинна розглядатися як мікросервіс в контейнері. Консорціум OpenFog виступає за метод розгортання контейнерів в якості основного методу розгортання програмного забезпечення на кордоні. Це має сенс, коли ми розглядаємо граничні пристрої як розширення хмари. Приклад розгортання контейнера може виглядати наступним чином. Кожен циліндр являє собою окремий контейнер, який можна розгорнути і керувати ним окремо. Потім кожен сервіс надає API для взаємодії контейнерів і шарів (рис. 4.8).

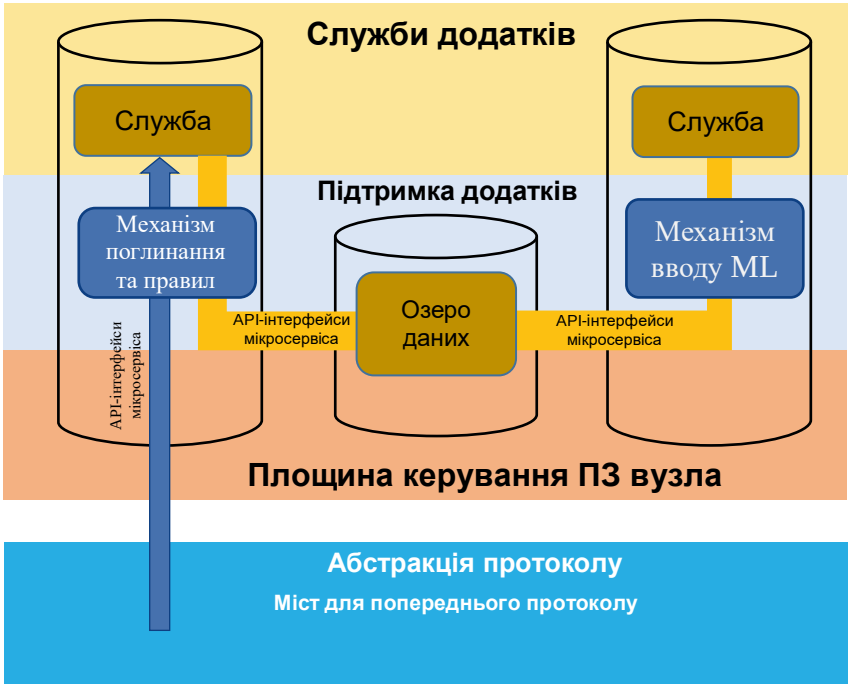


Рис. 4.8. Приклад програми OpenFog. Тут можуть бути розгорнуті кілька контейнерів, кожен з яких надає різні послуги та функції підтримки

Підтримка додатків

Це компонент інфраструктури, який допоможе зібрати остаточне рішення для клієнтів. Цей рівень залежить від того, як він був розгорнутий (наприклад, як контейнер). Підтримка здійснюється в багатьох формах, в тому числі:

управління додатком (ідентифікація образу, перевірка способу, установка образу, аутентифікація);

засоби журналювання;

реєстрація компонентів і сервісів;

движки виконання (контейнери, віртуальні машини);

мови виконання (Node.js, Java, Python);

сервери додатків (Tomcat);

шини повідомлень (RabbitMQ);

бази даних та архіви (SQL, NoSQL, Cassandra);

аналітичні фреймворки (Spark);

служби безпеки;

веб-сервери (Apache);

інструменти аналітики (Spark, Drool).

OpenFog пропонує, щоб ці служби були упаковані у вигляді контейнерів, як показано на рис. 4.8. Еталонна архітектура не є суворим керівництвом, і архітектор повинен вибрати правильний рівень підтримки, яка може бути включена на обмеженому граничному

пристрої. Наприклад, обробка та ресурси можуть допускати лише прості правила і забороняти що-небудь на зразок потокового процесора, не кажучи вже про рекурентні нейронні мережі.

Управління вузлом і базове ПО

Це відноситься до управління **In-Band (IB)** і визначає, як туманний вузол спілкується з іншими вузлами в своїй області. Вузли також управляються через цей інтерфейс для оновлень, статусу і розгортання. Базове ПО може включати в себе операційну систему вузла, користувацькі драйвери і прошивку, протоколи зв'язку і управління, управління файловою системою, програмне забезпечення для віртуалізації і контейнеризацію для мікросервісів.

Цей рівень стека програмного забезпечення стосується майже будь-якого іншого шару в OpenFog Reference Architecture. Типовими особливостями базового ПО є:

виявлення сервісу - дозволяє ситуативні моделі, довірчі відносини між хмарами;

виявлення вузла - дозволяє додавати туманні вузли і приєднувати їх до кластерів, подібних хмарним кластерам;

управління станом - дозволяє різні обчислювальні моделі для обчислень із збереженням стану і без стану для багатьох вузлів;

управління Pub/Sub - дозволяє переносити дані, а не витягувати їх. Крім того, дозволяє використовувати рівні абстракції при розробці програмного забезпечення.

Еталонна архітектура OpenFog або, в будь-якому випадку, як і будь-яка туманна архітектура, повинна забезпечувати рівні розгортання. Тобто, туманна архітектура не просто обмежена хмарою, підключеною до туманного шлюзу, з'єднаному з декількома датчиками. Насправді існує безліч топологій, що залежать від масштабу, пропускної здатності, навантаження і економічних факторів, які можуть бути розроблені. Еталонна архітектура повинна підходити для безлічі топологій, так само, як реальна хмара може динамічно масштабуватися і балансувати навантаження на підставі попиту.

Віртуалізація обладнання

Як і звичайні хмарні системи, OpenFog визначає апаратне забезпечення як рівень віртуалізації. Додатки не повинні мати прив'язки до певного набору обладнання. Тут система повинна балансувати навантаження через туман і переміщатися по ресурсам або додавати ресурси в міру необхідності. Всі апаратні компоненти віртуалізовані на цьому рівні, включаючи обчислення, мережу і сховище.

Безпека вузла OpenFog

Консорціум визначає цей рівень як частину безпеки обладнання стека. Туманні вузли вищого рівня повинні мати можливість

контролювати туманні вузли нижчого рівня як частину ієрархії в топології. Однорангові вузли повинні мати можливість спостерігати за своїми сусідами на сході-заході. Цей шар також виконує такі задачі:

криптування;

монітори спостереження і фізичної безпеки;

інспекція та моніторинг пакетів (зі сходу на захід і з півночі на південь).

Мережа

Це перший компонент рівня апаратної системи. Модуль являє собою модуль зв'язку схід-захід і північ-південь. Мережевий рівень обізнаний про туманну топологію і маршрутизації. Він виконує фізичну функцію маршрутизації до інших вузлів. Це велика відмінність від традиційної хмарної мережі, яка віртуалізує всі внутрішні інтерфейси. Тут мережу має сенс і географічна присутність в розгортанні IoT. Наприклад, батьківський вузол, керуючий чотирма іншими дочірніми вузлами, всі з яких підключені до камер, може нести відповідальність за об'єднання відеоданих з чотирьох джерел і об'єднання (злиття) вмісту зображення разом для створення поля зору на 360°. Для цього він повинен знати, який дочірній вузол відноситься до того чи іншого напрямку, і він не може робити це довільно або випадково.

Вимоги до мережевого компоненту:

- стійкість, якщо лінія зв'язку відмовляє. По суті, може знадобитися зрозуміти, як перебудувати мережу, щоб зберегти потік даних;

- мережевий рівень також є місцем для передачі даних і перепакування від не IP-датчиків до IP-протоколів. Прикладами цього є Bluetooth, Z-Wave і провідні датчики;

- обробка відмов;

- зв'язування з багатьма мережами (Wi-Fi, провідні, 5G);

- забезпечення типової мережевої інфраструктури, необхідної при розгортанні підприємства (безпека, маршрутизація і т.д.).

Прискорювачі [1]

Іншим аспектом OpenFog, чим він відрізняється від інших хмарних схем, є поняття послуг прискорювача. Прискорювачі тепер звичайні у вигляді GPGPU і навіть FPGA для надання послуг для обробки зображень, машинного навчання, комп'ютерного зору і сприйняття, обробки сигналів і шифрування/дешифрування. OpenFog передбачає, що туманні вузли можуть забезпечуватися ресурсами і розподілені по мірі необхідності. В ієрархії можна змусити ферму вузлів другого або третього рівня забезпечити додаткові обчислювальні засоби в міру необхідності динамічно. Ми можемо навіть змусити працювати інші форми прискорення в тумані, наприклад:

вузли, призначені для масового зберігання в разі, якщо необхідно створити велике озеро даних;

вузли, які включають в себе альтернативні лінії зв'язку, такі як супутникова радіостанція в катастрофічному випадку, коли не працює вся наземна зв'язок.

Обчислення

Обчислювальна частина стека аналогічна обчислювальної функціональності рівня Nova в OpenStack. Основні функції:

виконання завдання;

моніторинг і забезпечення ресурсів;

балансування навантаження;

запити можливостей.

Зберігання

Середовище зберігання архітектури підтримує інтерфейс низького рівня до туманного сховища. Типи зберігання, з якими ми стикалися раніше, такі як озера даних або пам'ять робочого простору, можуть знадобитися на кордоні для жорсткого аналізу в реальному часі. Шар зберігання також буде керувати всіма традиційними типами пристроїв зберігання, такими як:

масиви віртуальної пам'яті;

замінні диски;

fash-накопичувачі;

RAID;

шифрування даних.

Інфраструктура апаратної платформи

Рівень інфраструктури - це не стільки фактичний рівень між програмним і апаратним забезпеченням, скільки фізична і механічна структура туманного вузла. Оскільки туманні пристрої будуть знаходитися в суворих і віддалених місцях, вони повинні бути міцними, стійкими, а також автономними. OpenFog визначає випадки, які необхідно враховувати при розгортанні туманних обчислень, в тому числі:

розміри, потужність і вагові характеристики;

системи охолодження;

механічне закріплення і утримання;

механіки обслуговування;

стійкість до фізичного впливу і звітність.

Абстракція протоколу

Рівень абстракції протоколу пов'язує самі нижні елементи системи IoT (датчики) з іншими верствами туманного вузла, іншими туманними вузлами і хмарою. OpenFog підтримує модель абстракції для ідентифікації та зв'язку з сенсорним пристроєм через шар абстракції

протоколу. Абстрагуючи інтерфейс до датчиків і периферійних пристроїв, гетерогенну суміш датчиків можна розгорнути на одному туманному вузлі, наприклад, аналогові пристрої, які проходять через цифроаналогові перетворювачі, а також цифрові датчики. Навіть інтерфейси до датчиків можуть бути індивідуалізовані, наприклад, Bluetooth для температурних пристроїв в транспортних засобах може підключатися до інших датчиків двигуна, датчикам сполучення SPI на різних автомобільних електроприладах і датчикам GPIO до різних датчиків відкриття дверей і випадку крадіжки. Абстрагуючи інтерфейс, верхні шари стека програмного забезпечення можуть звертатися до таких розрізнених пристроїв за допомогою стандартизованого підходу.

Датчики, приводи і системи управління

Це нижній рівень стека IoT: датчики і граничні пристрої. Ці пристрої можуть бути розумними, німими, дротовими, бездротовими, розташовуватися ближче, далі і т. д. Однак об'єднання в тому, що вони якимось чином повідомляються з туманним вузлом, а той несе відповідальність за забезпечення, захист і управління кожним датчиком.

4.2.3. Туманні топології

Туманні топології можуть існувати в багатьох формах, і архітектору необхідно враховувати кілька аспектів при розробці туманної системи «від краю до краю». Зокрема, при розробці топології вступають в силу обмеження, такі як вартість, навантаження на процесор, інтерфейс виробника і передача між сходом-заходом. Туманна мережа може бути простою, як граничний маршрутизатор з підтримкою туману, що з'єднує датчики з хмарним сервісом. Вона також може ускладнюватися в багаторівневу туманну ієрархію туману з різним ступенем здатності обробки в кожному шарі і ролями на кожному рівні, одночасно перерозподіляючи навантаження обробки, коли і де це необхідно (зі сходу на захід і з півночі на південь) [1].

Визначальні фактори моделей засновані на наступному:

- **зменшення обсягу даних** - наприклад, система збирає неструктуровані відеодані з тисяч датчиків або камер, агрегує дані і шукає конкретні події в режимі реального часу. Якщо це так, то скорочення набору даних буде значним, так як тисячі камер будуть щодня виробляти сотні ГБ даних, а туманним вузлам потрібно буде перевести великі обсяги даних в прості форми: «так», «ні», «небезпека», «токени безпеки події»;
- **кількість граничних пристроїв** - якщо система IoT являє із себе всього лише один датчик, то набір даних малий і може не виправдовувати використання граничного туманного вузла.

Однак, якщо число датчиків зростає або, в гіршому випадку, зростання кількості датчиків непередбачуване і динамічне, то туманна топологія може зажадати масштабування вгору або вниз динамічно. Ми розглядаємо як приклад стадіон з використанням Bluetooth-маяка. Оскільки аудиторія зростає на певних майданчиках, система повинна мати можливість масштабуватися нелінійно. В інших випадках стадіон може займати лише невелику площу, і йому потрібні тільки обмежені ресурси обробки та підключення;

- **можливості туманного вузла** - в залежності від топології і вартості деякі вузли можуть краще підходити для підключення до систем WPAN, тоді як інші вузли в ієрархії можуть мати додаткові можливості по обробці для машинного навчання, розпізнавання образів або роботи із зображеннями. Прикладом можуть бути граничні туманні вузли, які управляють безпечної сіткою Zigbee і мають спеціальне обладнання для аварійних ситуацій або безпеки WPAN. Вище цього рівня буде існувати вузол обробки туману, який буде мати додаткову оперативну пам'ять і GPGPU для підтримки потоків необроблених даних з шлюзів WPAN;
- **надійність системи** - архітекторові може знадобитися розгляд форм відмови в моделі IoT. Якщо один крайній туманний вузол дав збій, інший міг би зайняти його місце для виконання якої-небудь дії або сервісу. Цей випадок важливий в життєво-критичних випадках або при роботі в реальному часі. Таким же чином додаткові туманні вузли можуть підключатися на вимогу; надлишкові вузли можуть знадобитися в ситуаціях забезпечення відмовостійкості. У разі відсутності додаткових надлишкових вузлів деяка обробка може спільно здійснюватися сусідніми вузлами за рахунок використання системних ресурсів і затримки, але система буде продовжувати функціонувати. Крайовий варіант використання - це те, де сусідні вузли діють як сторожові пси один для одного. В разі збою туманного вузла або збою зв'язку з вузлом сторожовий таймер сигналізує про подію збою для хмари і може виконувати локальні критичні дії. Хорошим прикладом є випадок, коли туманний вузол дає збої при контролі трафіку на шосе; сусідній вузол може побачити відмову точки, попередити хмару про подію і подати сигнал на електронне табло, що розміщене на шосе, щоб зменшити швидкість.

Найпростішим рішенням для туману є блок обробки на краю (шлюз, «тонкі клієнти», маршрутизатор), розташовані поруч з масивом датчиків. Тут туманний вузол може використовуватися в якості шлюзу

для мережі або mesh-мережі WPAN і обмінюватися даними з хостом (рис. 4.9).

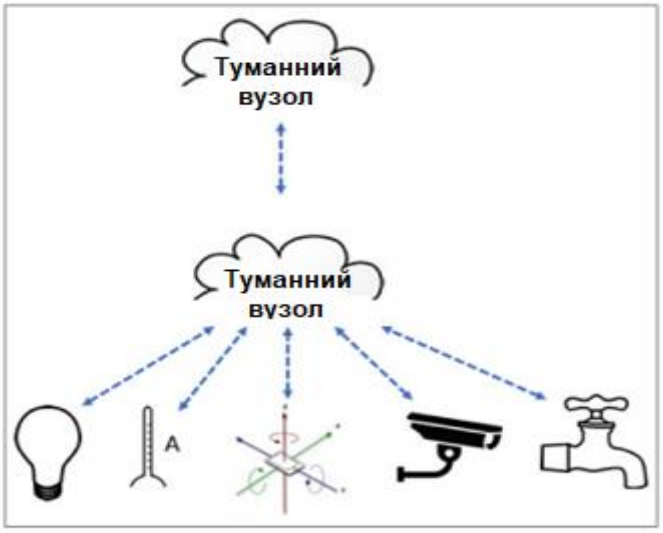


Рис. 4.9. Проста туманна топологія. Гранично-туманний пристрій управляє масивом датчиків і може взаємодіяти з іншим туманним вузлом на основі M2M [1]

Наступна базова туманна топологія включає хмару як батька по туманній мережі. Туманний вузол в цьому випадку буде збирати дані, захищати край і виконувати обробку, необхідну для зв'язку з хмарою. Цю модель відокремлює від граничних обчислень те, що сервісні і програмні рівні туманного вузла поділяють відносини з хмарним фреймворком (рис. 4.10).

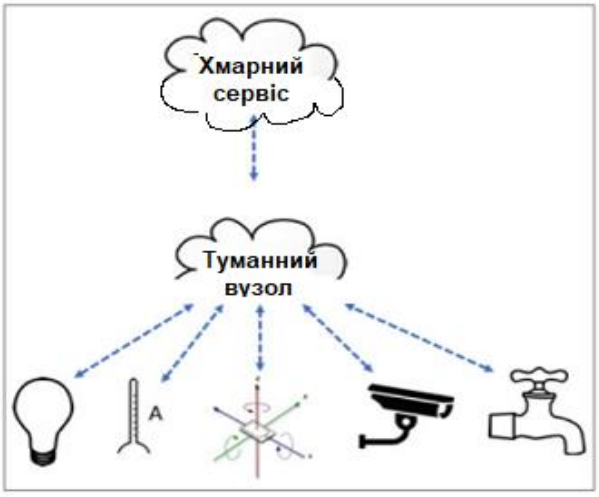


Рис. 4.10. Підключення «туману» до хмарної топології. Тут туманний вузол встановлює відносини з постачальником хмари [1]

Наступна модель використовує кілька туманних вузлів, відповідальних за послуги і обробку на краю, і кожен з них підключається до набору датчиків.

Батьківська хмара забезпечує кожен туманний вузол, так ніби б він був єдиним вузлом. Кожен вузол має унікальний ідентифікатор, щоб забезпечити унікальний набір сервісів на основі розташування. Наприклад, кожен туманний вузол може перебувати в іншому місці для роздрібної мережі. Туманні вузли також можуть пов'язувати і передавати дані зі сходу на захід між граничними вузлами. Прикладом можуть служити умови холодного зберігання, коли необхідно підтримувати і управляти вентиляторами і морозильниками, щоб запобігти псуванню продуктів живлення. У роздрібного продавця може бути кілька вентиляторів в різних місцях, все управляються єдиною хмарною службою, але працюють вони з туманними вузлами на краю (рис. 4.11).

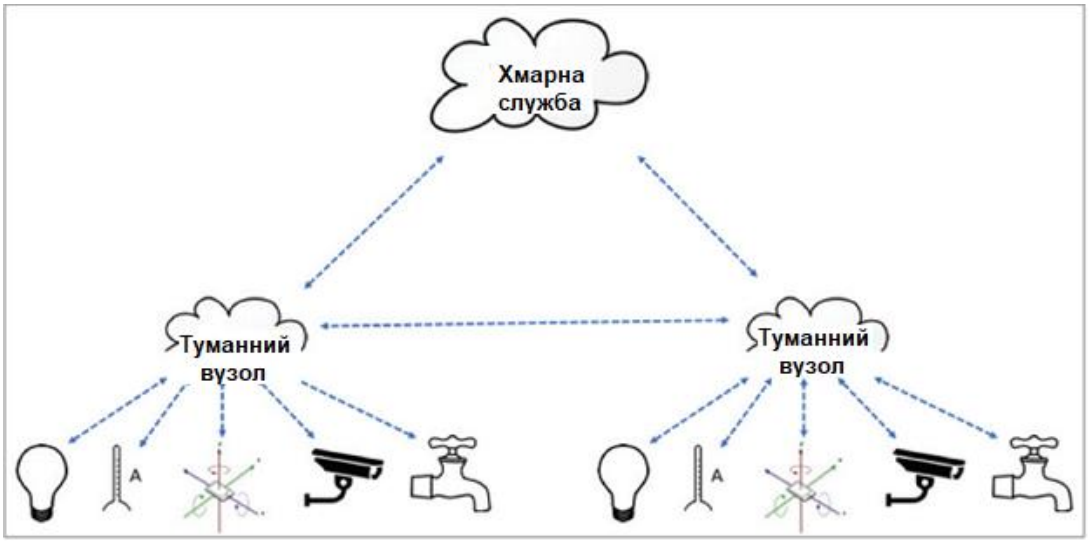


Рис. 4.11. Кілька туманних вузлів з однією основним хмарою [1]

Наступна модель розширює другу топологію, додаючи можливість надійно і конфіденційно спілкуватися з декількома постачальниками хмар від декількох туманних вузлів. У цій моделі можуть бути розгорнуті кілька батьківських хмар. Наприклад, в розумних містах можуть існувати численні географічні райони, і вони можуть охоплюватися різними муніципалітетами. Кожен муніципалітет може вибрати одного хмарного провайдера, порівнявши його з іншими, але всі муніципалітети управляються з використанням одного затвердженого та бюджетного виробника камер і датчиків. У цьому випадку виробник камер і датчиків матиме один екземпляр хмари, що співіснує з декількома муніципалітетами.

Туманні вузли повинні мати можливість доставляти дані декільком постачальникам хмарних обчислень (рис. 4.12).

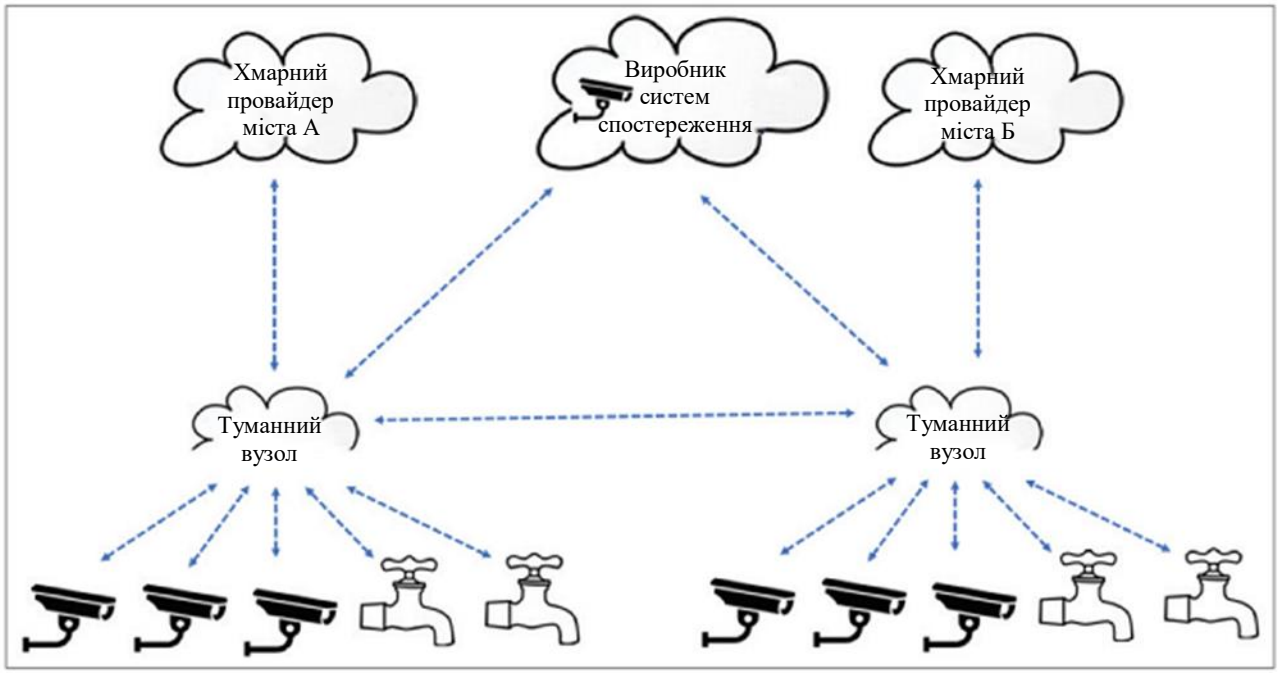


Рис. 4.12. Кілька туманних вузлів з кількома хмарними провайдерами. Хмари можуть являти собою суміш громадських і приватних хмар [1]

Туманні вузли також не потребують індивідуального з'єднання, що зв'язує датчики з хмарами «один до одного». Туманні вузли можуть бути поміщені в стек, багаторівневими або навіть законсервованими до тих пір, поки не знадобляться. Ієрархія рівнів туманних вузлів один над одним може здаватися суперечливою, якщо ми намагаємося зменшити затримки, але, як згадувалося раніше, вузли можуть бути спеціалізованими. Наприклад, вузли, розташовані ближче до датчиків, можуть надавати послуги в режимі реального часу або мати обмеження за вартістю, що вимагає від них мінімального обсягу даних для зберігання і обчислення. Рівні над ними можуть надавати обчислювальні ресурси, необхідні для агрегованого зберігання, машинного навчання або розпізнавання зображень з використанням додаткових пристроїв, що запам'ятовують або GPGPU-процесорів. Наступний приклад ілюструє використання на прикладі освітлення міста.

Тут кілька камер фіксують рух пішоходів і трафік; туманні вузли, які мають найтісніший контакт з камерами, виконують агрегацію і витяг ознак і передають ці дані вгору до наступного рівня. Батьківський туманний вузол отримує дані і виконує необхідне розпізнавання зображень за допомогою алгоритму глибокого навчання. Якщо буде спостерігатися цікава подія (наприклад, пішохідна прогулянка вночі вздовж шляху), подію буде відправлено в хмару. Компонент хмари

реєструє подію і сигналізує множині вуличних ліхтарів в околиці пішохода, щоб вони збільшили освітленість. Ця картина буде тривати до тих пір, поки туманні вузли бачать рух пішохода. Кінцевою метою є загальна економія енергії, щоб кожна вулична лампочка не працювала на повну інтенсивність весь час (рис. 4.13)

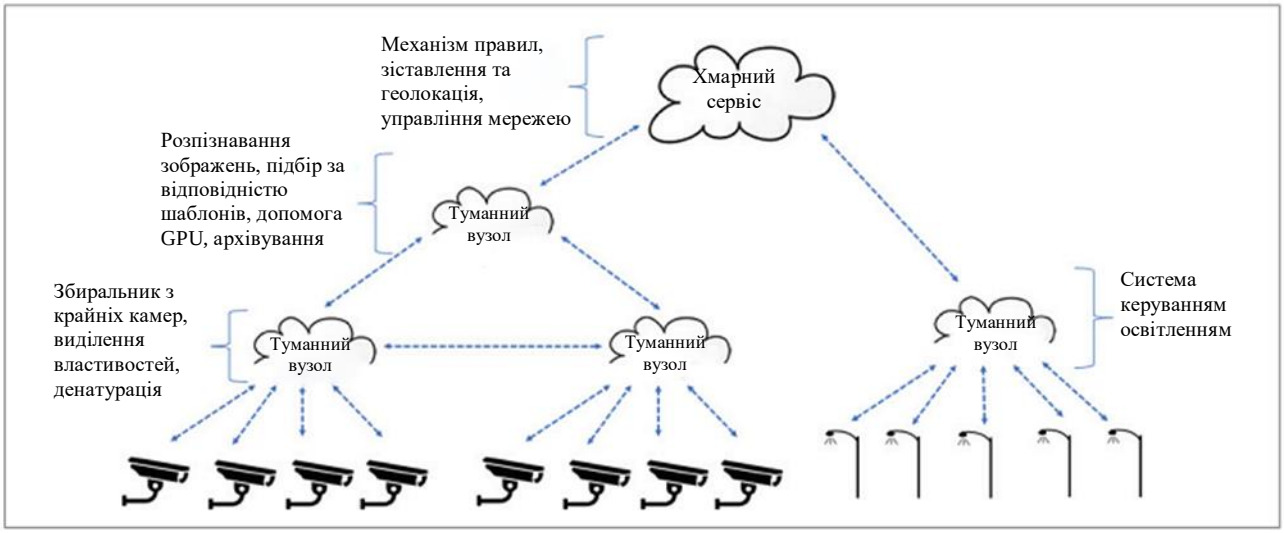


Рис. 4.13. Багаторівнева туманна топологія. Туманні вузли складаються в стек в ієрархії рівнів, щоб забезпечити додаткові послуги або абстракції [1]

Таким чином, туманні обчислення роблять хмарну обробку ближче до краю для вирішення проблем із затримкою, безпекою та витратами на зв'язок. Хмарні та туманні технології працюють разом, щоб забезпечити роботу аналітичних пакетів у вигляді движків правил для складних агентів обробки подій. Вибір моделі хмарних провайдерів, фреймворків, туманних вузлів і модулів аналітики є важливим завданням, і безліч матеріалів присвячено глибокому розгляду семантики програмування і створення цих сервісів. Архітектор IoT повинен зрозуміти топологію і кінцеву мету системи, щоб побудувати структуру, яка задовольняє сьогоднішні потреби і масштабується в майбутньому.

У наступному підрозділі ми обговоримо аналітику даних IoT. Хмара, безумовно, може містити ряд аналітичних функцій, однак необхідно чітко розуміти, що певний аналіз повинен виконуватися на краю, ближчому до джерела даних (датчика), або, якщо це має сенс, в хмарі (використовуючи довгострокові історичні дані).

4.3. Аналіз даних в IoT

Сьогодні світ перетворився на величезний цифровий простір. Ми управляємо, ділимося та фактично зберігаємо всі аспекти нашого життя онлайн [28].

Дані зі всіх наших пристроїв - комп'ютерів, планшетів та смартфонів - постійно збираються та передаються в мережу, та насправді це лише початок процесу. Незабаром вся інформація буде потрапляти онлайн навіть з таких пристроїв, як годинники, телевізори, датчики в розумних будинках, авто, обладнання на виробництві та з безлічі інших девайсів. Крім того, ми самостійно продукуємо гігабайти інформації, коли спілкуємося з друзями в соцмережах, робимо покупки онлайн, користуємося пошуком.

Цікавий факт: якщо зібрати всю інформацію, яку накопичило людство з початку часів включно до 2000-го року, то виявиться, що її менше, ніж ми продукуємо зараз протягом лише одної хвилини. Цей феномен повністю змінює розуміння світу та нашого місця в ньому.

Однак, основна цінність IoT-систем полягає не в архівації мільйонів подій, згенерованих датчиками, а в їх інтерпретації та прийнятті відповідних рішень. Світ, у якому мільярди сутностей з'єднуються і взаємодіють один з одним та з хмарними платформами, захоплює дух, але нас насамперед цікавлять дані, їх зміст, те, що в них не потрапило, та закономірності, які з них можна вивести. Це та область Інтернету речей, яка відноситься до науки про дані та їх аналіз, є, напевно, найбільш цінною для клієнтів.

4.3.1. Типи даних для аналізу в Інтернеті речей

В Інтернеті речей циркулюють різні типи даних. Для вилучення з них корисної інформації необхідно провести їх обробку та аналіз. У свою чергу, кожен тип даних потребує свої інструменти і методи обробки та аналізу. Галузь науки, що вивчає методи та алгоритми отримання інформації з даних, отримала назву «Data Science ».

Основні категорії даних, що зустрічаються в Інтернеті речей для обробки і аналізу, можна звести в такі типи:

- Структуровані.
- Неструктуровані.
- На природній мові.
- Машинні.
- Графові.
- Аудіо, відео та графіка.
- Потокові.

Всі ці типи даних являють інтерес, і їх варто розглянути докладніше.

Структуровані дані залежать від моделі даних і зберігаються в фіксованому полі всередині запису. Відповідно, структуровані дані часто зручно зберігати в таблицях, в базах даних або файлах Excel (рис. 4.14). SQL (Structured Query Language, мова структурованих запитів) є основним засобом управління і поводження з запитами до даних, що зберігаються в базах даних. Також іноді зустрічаються структуровані дані, які досить важко зберегти в традиційній реляційній базі даних (один із прикладів - ієрархічні дані, наприклад генеалогічне дерево).

Втім, світ не складається з структурованих даних; просто це подання зручно для людини і машин. Найчастіше реальні дані зберігаються в неструктурованому вигляді.

	Indicator ID	Dimension List	Timeframe	Numeric Value	Missing Value Flag	Confidence Inte
2	214390830	Total (Age-adjusted)	2008	74.6%		73.8%
3	214390833	Aged 18-44 years	2008	59.4%		58.0%
4	214390831	Aged 18-24 years	2008	37.4%		34.6%
5	214390832	Aged 25-44 years	2008	66.9%		65.5%
6	214390836	Aged 45-64 years	2008	88.6%		87.7%
7	214390834	Aged 45-54 years	2008	86.3%		85.1%
8	214390835	Aged 55-64 years	2008	91.5%		90.4%
9	214390840	Aged 65 years and over	2008	94.6%		93.8%
10	214390837	Aged 65-74 years	2008	93.6%		92.4%
11	214390838	Aged 75-84 years	2008	95.6%		94.4%
12	214390839	Aged 85 years and over	2008	96.0%		94.0%
13	214390841	Male (Age-adjusted)	2008	72.2%		71.1%
14	214390842	Female (Age-adjusted)	2008	76.8%		75.9%
15	214390843	White only (Age-adjusted)	2008	73.8%		72.9%
16	214390844	Black or African American only (Age-adjusted)	2008	77.0%		75.0%
17	214390845	American Indian or Alaska Native only (Age-adjusted)	2008	66.5%		57.1%
18	214390846	Asian only (Age-adjusted)	2008	80.5%		77.7%
19	214390847	Native Hawaiian or Other Pacific Islander only (Age-adjusted)	2008	DSU		
20	214390848	2 or more races (Age-adjusted)	2008	75.6%		69.6%

Рис. 4.14. Приклад структурованих даних

Неструктуровані дані важко підігнати під конкретну модель даних, тому що їх зміст залежить від контексту або має змінний характер. Один із прикладів неструктурованих даних - звичайні повідомлення електронної пошти (рис. 4.15). Хоча повідомлення містить структуровані елементи (відправник, заголовок, тіло), одні і ті ж завдання можуть вирішуватися безліччю різних способів, наприклад, існує незліченна кількість варіантів згадування конкретної людини в повідомленнях. Проблема додатково ускладнюється існуванням тисяч мов і діалектів.

Повідомлення електронної пошти, написане людиною (на зразок показаного на рис. 4.15), також є ідеальним прикладом даних природною мовою.

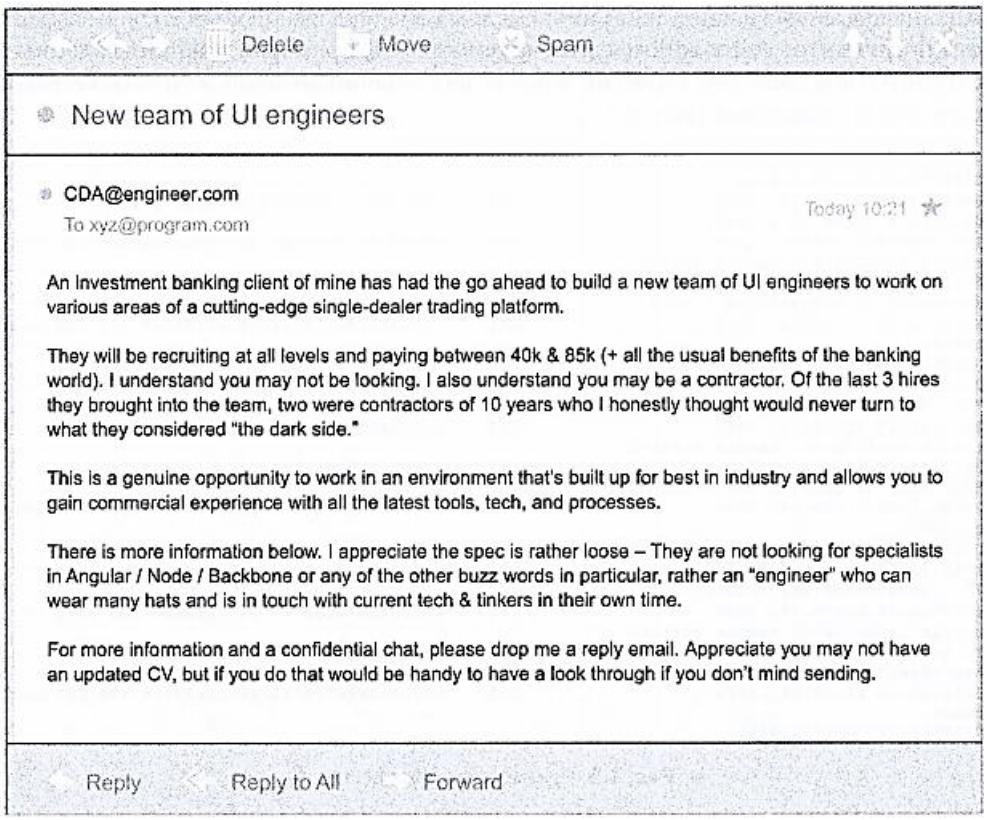


Рис. 4.15. Приклад неструктурованих даних:
поштове повідомлення

Дані на природній мові становлять особливий різновид неструктурованих даних; обробка таких даних достатньо складна, тому що вона вимагає знання як лінгвістики, так і спеціальних методів Data Science .

Спільнота обробки даних на природній мові домоглося успіху в області розпізнавання сутностей, розпізнавання тематичних областей, узагальнення, завершення тексту і аналізу емоційного забарвлення, але моделі, адаптовані для однієї предметної області, погано узагальнюються для інших областей. Навіть самі сучасні методи не зможуть розшифрувати сенс довільного фрагмента тексту. І цей факт навряд чи когось здивує: у людей також виникають проблеми зі сприйняттям природної мови. Він неоднозначний за своєю природою. Сама концепція сенсу виглядає спірно. Двоє людей слухають одну розмову та винесуть зовсім різний смисл з неї. Адже сенс може відрізнятися навіть від настрою того хто говорить.

Машинні дані. До машинних даних відноситься інформація, що автоматично генерується комп'ютером, процесом, додатком або пристроєм без втручання людини і передається разом з іншими даними. Машинні дані стають одним з основних джерел інформації, і ситуація навряд чи зміниться.

Аналіз машинних даних через їх величезні обсяги і швидкості сильно залежить від інструментів з високою масштабованість. До прикладів машинних даних відносяться журнали веб-серверів, записи деталізації дзвінків, журнали мережевих подій і телеметрії (рис. 4.16.)

```
CSIPERF:TXCOMMIT;313236
2014-11-28 11:36:13, Info                  CSI    00000153 Creating NT transaction (seq
69), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000154 Created NT transaction (seq 69)
result 0x00000000, handle @0x4e54
2014-11-28 11:36:13, Info                  CSI    00000155@2014/11/28:10:36:13.471
Beginning NT transaction commit...
2014-11-28 11:36:13, Info                  CSI    00000156@2014/11/28:10:36:13.705 CSI perf
trace:
CSIPERF:TXCOMMIT;273983
2014-11-28 11:36:13, Info                  CSI    00000157 Creating NT transaction (seq
70), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000158 Created NT transaction (seq 70)
result 0x00000000, handle @0x4e5c
2014-11-28 11:36:13, Info                  CSI    00000159@2014/11/28:10:36:13.764
Beginning NT transaction commit...
2014-11-28 11:36:14, Info                  CSI    0000015a@2014/11/28:10:36:14.094 CSI perf
trace:
CSIPERF:TXCOMMIT;386259
```

Рис. 4.16. Приклад машинних даних

Машинні дані на рис. 4.16 добре вкладаються в структуру класичної бази даних. Це не кращий формат для даних з високим ступенем зв'язності або «мережевих» даних, в яких досить значиму роль грають відносини між сутностями.

Графові, або мережеві, дані. Термін «графові дані» може збити з пантелику тому що будь-які дані можуть бути представлені у вигляді графа. Під «графом» в даному випадку мається на увазі поняття графа з математичної теорії графів – математична структура для моделювання попарних відносин між об'єктами. Якщо коротко, в графових, або мережевих, даних особлива увага приділяється зв'язкам або суміжності об'єктів. Графові структури даних використовують вузли, ребра і властивості для представлення і збереження графічних даних. Графові дані природним чином підходять для подання соціальних мереж, а їх структура дозволяє обчислювати такі специфічні метрики, як вплив учасників і найкоротший шлях між двома людьми.

Приклади графових даних зустрічаються на багатьох веб-сайтах соціальних мереж (рис. 4.17). Наприклад, в LinkedIn можна побачити, кого ви знаєте в тій чи іншій компанії. Ваш список, хто вам пише в Твіттері також є прикладом графових даних. Сила і міць пов'язаних даних проявляється при аналізі декількох графів, що перекриваються, побудованих на одних і тих же вузлах. Наприклад, уявіть, що ребра позначають «друзів» на Фейсбук. А тепер візьмемо інший граф з тими ж людьми, але він зв'язує колег по бізнесу через LinkedIn, і третій граф,

заснований на інтересі до фільмів на Netflix. Накладення цих трьох графів дозволить отримати відповіді на багато цікавих питань.

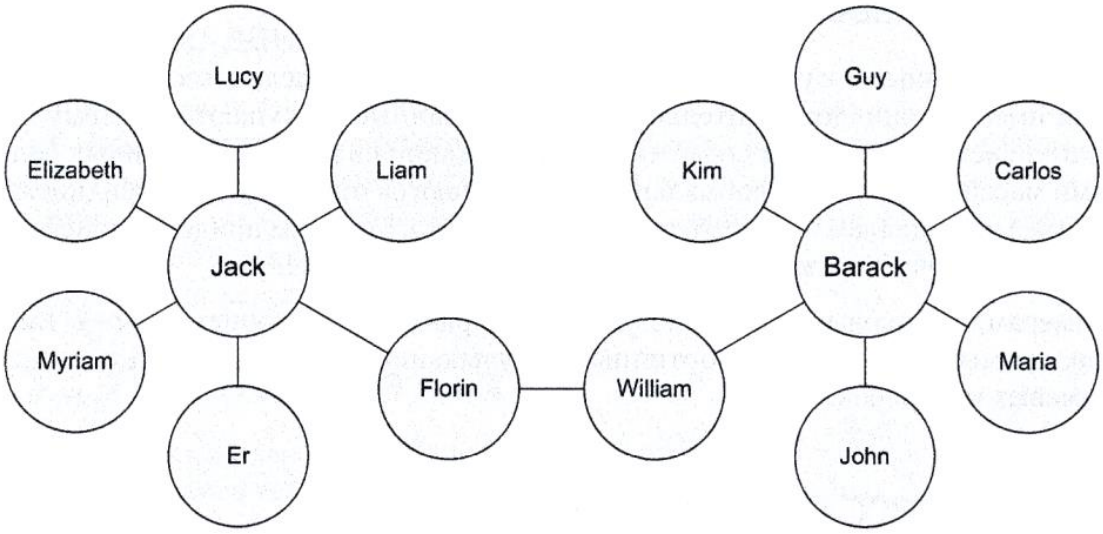


Рис. 4.16. Приклад графових данних

Для зберігання графових даних використовуються графові бази даних, а для побудови запитів до них – такі спеціалізовані мови запитів, як SPARQL. Робота з графовими даними створює специфічні проблеми, причому для комп’ютера це завдання стає ще складніше.

Аудіо, відео та графіка – типи даних, що ставлять непрості завдання перед спеціалістом з аналізу. Завдання, тривіальні з точки зору людини (наприклад, розпізнавання об'єкта на картинці), виявляються складними для комп'ютера. Ще у 2014 році компанія MLBAM (Major League Baseball Advanced Media) оголосила, що обсяг записуваних відеоматеріалів для одного бейсбольного матчу буде збільшений приблизно до 7 Тбайт з метою проведення оперативного аналізу. Високошвидкісні камери на стадіонах записують руху м'яча і спортсменів для того, наприклад, щоб обчислювати в реальному часі траєкторію руху захисника.

Нещодавно компанії DeepMind вдалося створити алгоритм, який здатний навчатися грати у відеоігри. Алгоритм отримує на вході зміст екрану і навчається інтерпретувати ці дані в складному процесі глибокого навчання. Це гарне досягнення, і компанія Google придбала DeepMind для розробки штучного інтелекту. Алгоритм навчання отримує дані, які генеруються комп'ютерною грою, тобто потокові дані.

Потокові дані можуть приймати майже будь-яку з перерахованих форм, однак у них є одна додаткова властивість. Дані надходять в систему при виникненні деяких подій, а не завантажуються в сховище даних великими масивами. І хоча формально вони не є окремим різновидом даних, виділяємо їх в особливу категорію, тому що

доведеться пристосувати свій робочий процес для роботи з потокової інформацією.

Прикладами потокових даних можуть бути розділ «Що відбувається?» в Твіттері, прямі трансляції спортивних і музичних заходів та дані біржових котирувань.

Іноді дані необхідно інтерпретувати та аналізувати у режимі реального часу у вигляді потоку, а іноді вони архівуються з подальшим поглибленим аналізом у хмарі. Це вже етап *споживання даних*. У таких випадках інформацію доводиться зіставляти з іншими джерелами на льоту, а іноді вона просто записується і скидається в так зване *озеро даних* (англ. data lake), на зразок Hadoop.

Далі йде етап переміщення. Такі системи обміну повідомленнями, як Kafka, направляють дані в потоковий або пакетний процесор (або обидва одночасно). Дані обробляються як безперервний потік. Цей процес зазвичай має певні обмеження та високу продуктивність, оскільки інформація обробляється у пам'яті. У зв'язку з цим обробка даних має відбуватися не повільніше, ніж їхнє надходження. І якщо у хмарі можна розраховувати на роботу в режимі, близькому до реального часу, промислові пристрої або безпілотні автомобілі не дають жорстких гарантій щодо продуктивності їх обчислювачів.

З іншого боку, пакетна обробка добре підходить для великих обсягів даних, особливо у порівнянні показників датчиків з раніше збереженою інформацією.

Наступний етап може включати прогнозування і повернення відповіді. Дані можуть бути виведені на панель керування, записані в журнал або повернені назад до прикордонного пристрою, який може вжити певних заходів для вирішення деяких проблем.

Тут ми коротко розглянемо різні моделі аналізу даних: від обробки складних подій до машинного навчання.

4.3.2. Простий аналіз даних в Інтернеті речей

Аналіз даних займається пошуком подій, як правило, у потоках інформації. Існує кілька видів подій і ролей, які мають підтримувати пристрої потокового аналізу в режимі реального часу. Нижче наведено загальні категорії аналітичних функцій, що базуються на доповіді Сріната Перери та Сріскандараї Сухотаяна «Шаблони рішень для потокового аналізу в режимі реального часу», представленому на 9-й Міжнародній конференції з розподілених подійних систем (DEBS '15) у Нью-Йорку [1]:

Попередня обробка - відкидання малоцікавих подій, денатурування, вилучення властивостей, сегментація, переведення даних у більш

відповідний формат (хоча озера даних воліють уникати негайного перетворення), додавання таких атрибутів, як ярлики (озерам даних ярлики не потрібні).

Оповіщення – перевірка даних; якщо вони не відповідають якимось граничним умовам, надсилається попередження. Елементарний приклад: температура перевищує певний ліміт, встановлений у датчику.

Вікна подій – створюється рухомий діапазон (вікно), у якому діють власні правила. Вікно може бути обмежене за часом (наприклад, протягом однієї години) або довжиною (наприклад, у межах 2000 показань датчика). Вікна можуть бути рухомими (наприклад, перевірка останніх 10 подій датчика та повернення результатів при кожній новій події) та пакетними (коли подія генерується лише після завершення вікна). Ця функція добре підходить для створення правил та подій, що займаються підрахунком. Наприклад, ви можете дізнатися кількість стрибків температури за останню годину і зробити висновок про те, що один із датчиків має дефект.

З'єднання - об'єднують кілька потоків даних в один. Можна навести приклад у сфері логістики. Припустимо, компанія розсилає посилки та відстежує їх за допомогою спеціальних маяків, а її численні вантажівки, літаки та об'єкти нерухомості шлють потоки геолокаційних даних. Таким чином, маємо два потоки: один для посилки, а інший для заданої вантажівки. Коли вантажівка підбирає посилку, ці два потоки поєднуються.

Помилки - працюючи з мільйонами датчиків, ви неодмінно зіткнетеся з втратою даних, їх спотворенням або неправильним порядком прямування. Це актуально для Інтернету речей з його численними асинхронними та незалежними потоками. Наприклад, інформація може загубитися в мережі, якщо автомобіль заїде в підземний гараж. Цей аналітичний шаблон зіставляє дані у межах одного потоку, намагаючись знайти подібні помилки.

Бази даних – аналітичний пакет має взаємодіяти з якимось сховищем даних. Наприклад, якщо дані надходять з низки датчиків, таких як ярлики Bluetooth, що сигналізують про розкрадання або зникнення пристрою, вам потрібно буде зіставляти їх з базою даних ідентифікаторів ярликів, що розшукуються.

Тимчасові події та шаблони – ця функція найчастіше застосовується у поєднанні з вікнами подій, згаданими вище. Тут цільовим шаблоном є набори чи послідовності подій. Це такий кінцевий автомат. Уявіть, що ми відстежуємо стан пристрою за його температурою, вібраціями та шумом, який він видає. Послідовність тимчасових подій може мати такий вигляд:

1) визначити, чи температура перевищує 100 °C;

2) визначити, чи перевищує рівень вібрацій 1 м/с;

3) визначити, чи пристрій видає звук гучністю 110 дБ;

4) надіслати оповіщення, якщо всі ці події відбуваються у такій послідовності.

Відстеження – подія генерується за фактом наявності або відсутності будь-яких даних у заданому місці або у певний момент часу. Найпростішим прикладом є геолокація для службових автомобілів, коли компанії необхідно знати їхнє точне розташування і коли вони востаннє відвідували те чи інше місце. Цю функцію можна застосовувати в сільському господарстві, комунальному вивезенні сміття, прибиранні снігу, для відстеження пасажиропотоків, пацієнтів, цінних активів, багажу тощо.

Тенденції – цей шаблон особливо корисний у прогностичному обслуговуванні. Він передбачає створення правила виявлення подій на основі наборів даних, пов'язаних за часом. Тимчасові події мають схожий принцип роботи, але натомість у них фігурує поняття послідовності. У цій моделі час є одним із вимірів процесу. Актуальна історія подій, пов'язаних з часом, може використовуватися у сільському господарстві. Наприклад, до голови корови можна прикріпити датчик, який відстежуватиме її переміщення та температуру. Щоб подивитися, чи корова рухалася протягом дня, можна скласти послідовність подій. Відсутність руху може вказувати на те, що тварина захворіла або померла.

Пакетні запити – пакетна обробка зазвичай виявляється більш комплексною і глибокою, ніж обчислення в реальному часі. Добре спроектована потокова платформа може розділяти аналіз на кілька частин та передавати їх системі пакетної обробки. Ми ще повернемося до цієї теми у обговоренні lambda-функцій.

Глибокий аналіз – під час обробки у режимі реального часу рішення про виникнення подій приймаються на льоту. Чи ця подія повинна згенерувати повідомлення – це питання, відповідь на яке може вимагати додаткової обробки, яка виконується пізніше. Такий підхід працює, оскільки події подібного роду мають бути рідкісними; і, поки нові події створюються в режимі реального часу, старі передаються в движок глибокої обробки. Як приклад можна навести систему відеоспостереження. Уявіть, що платформа розумного міста б'є на сполох, якщо зникла дитина, і запускає просту модель розпізнавання та класифікації зображень для потокових движків реального часу. Модель здатна виявити номерний знак автомобіля, де знаходиться дитина, або навіть логотип на дитячій футболці. Насамперед потрібно буде захопити зображення з номерними знаками та логотипами на одязі перехожих та передати їх у хмару. Із мільйонів отриманих зразків система аналізу

спробує розпізнати потрібний номер чи логотип. Це перший етап. Потім, щоб виключити помилкові спрацьовування, розпізнаний кадр (разом із сусідніми кадрами) буде спрямований на більш просунуту систему аналізу, яка використовує докладніші алгоритми розпізнавання об'єктів (злиття зображень, збільшення роздільної здатності, машинне навчання).

Моделі та навчання – модель першого рівня, описана вище, насправді може бути механізмом логічного висновку системи машинного навчання. Подібні системи будуються на моделях, що навчаються і можуть використовуватися на льоту під час аналізу у режимі реального часу.

Обмін сигналами – дії часто доводиться повертати назад до прикордонного компонента або датчика. Типовим прикладом є автоматизація та безпека фабрики. Якщо температура пристрою перевищить певний ліміт, цю подію потрібно не лише записати в журнал, але й надіслати команду прикордонному компоненту, щоб уповільнити пристрій. Механізм комунікацій у системі має бути двоспрямованим.

Управління – нарешті, ми маємо можливість керувати усіма цими засобами аналізу. Для роботи з системою повинні бути передбачені такі функції як запуск, зупинка, створення звітів, запис до журналу та налагодження.

Тепер ми зосередимося на побудові хмарної аналітичної архітектури, здатної споживати непередбачувані та безперервні потоки даних та інтерпретувати їх у режимі, максимально близькому до реального часу.

Верхній рівень хмарної архітектури

На рис. 4.17 показаний типовий шлях проходження даних від датчика до панелі управління. Дані проходять через кілька проміжних ланок (вузли WPAN, широкосмуговий канал, хмарне сховище у вигляді озера даних тощо). При виборі архітектури для побудови аналітичного рішення хмарного слід враховувати ефект масштабування. Рішення, прийняті на ранніх етапах проектування, можуть пригодитися для десятка IoT-вузлів та одного хмарного кластера, але коли кількість кінцевих IoT-пристроїв зросте до тисяч, а система почне охоплювати кілька географічних зон, масштабування може виявитися неефективним.

Аналітичні функції хмарної системи (прогнозування-відповідь) можуть набувати кількох форм:

система правил – визначає дію та повертає результат;

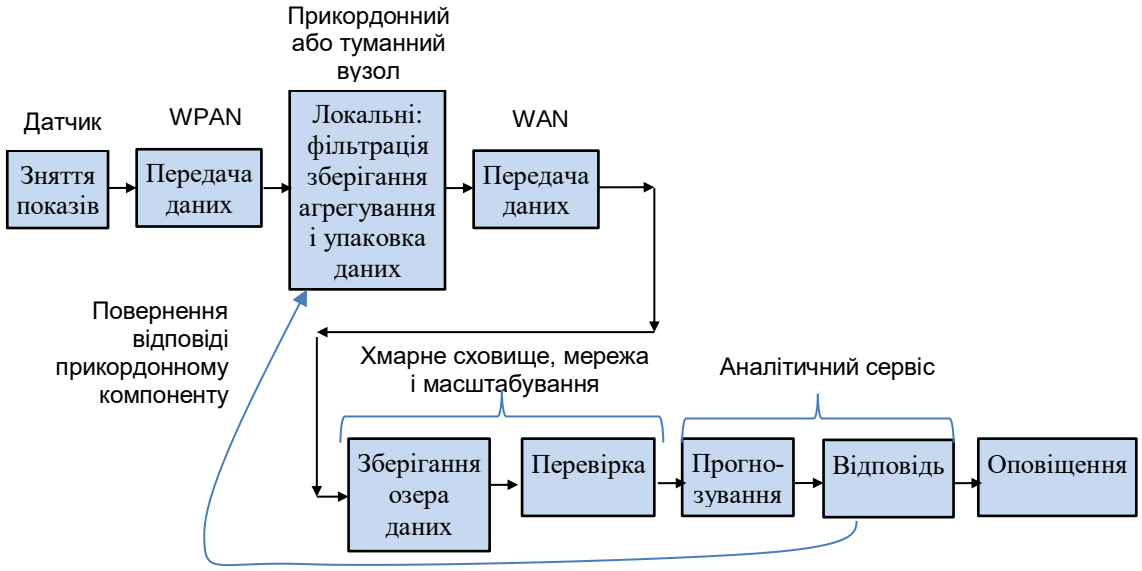


Рис. 4.17. Типова схема передачі даних від датчика до хмари

поточна обробка – тут події, такі як показання датчиків, передаються потоковому процесору. Шлях обробки є граф з операторами як вузли; при цьому дані переходять від одного оператора до іншого. Кожен вузол містить код для певного етапу обробки та посилання на наступний вузол у графі. Такий граф можна реплікувати та виконувати паралельно у кластері, тому його можна масштабувати на сотні комп'ютерів;

обробка складних подій - ця частина системи заснована на мові запитів високого рівня SQL та займається обробкою подій. Вона заточена під низькі затримки;

Lambda-архітектура – ця модель намагається збалансувати пропускну спроможність системи та рівень затримок, виконуючи пакетну обробку та паралельні обчислення з величезними наборами даних.

Система правил [1]

Система правил - це програмний компонент, який виконує якісь дії у відповідь на події. Наприклад, якщо вологість у кімнаті перевищить 50%, власнику надсилається SMS. Цей механізм називають системою управління бізнес-правилами (англ. Business Rule Management System, або BRMS).

Деякі системи правил мають стан. Це означає, що вони можуть зберігати історію подій і виконувати різні дії залежно від їхнього порядку, кількості чи характеру їх надходження. Системи, які не мають стану, перевіряють лише поточну подію (рис. 4.18).

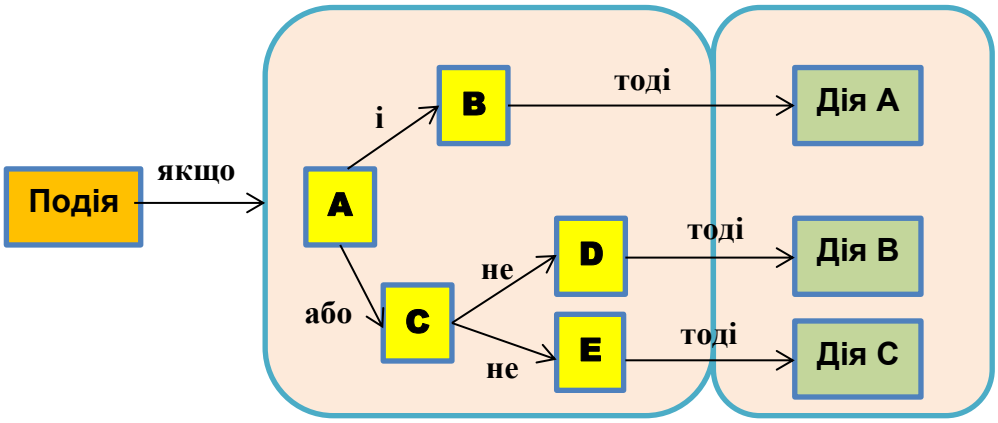


Рис. 4.18. Приклад простої системи правил

У нашому прикладі роль системи правил виконуватиме Drools. Drools – це BRMS, розроблена компанією Red Hat і що розповсюджується під ліцензією Apache 2.0. Її промислова версія називається JBoss Enterprise. Всі свої об'єкти Drools зберігає у робочій пам'яті; це такий набір подій, надісланих датчиками, які слід порівняти задоволення заданого правила.

Drools підтримує два види зчеплення: пряме та зворотне. *Зчеплення* – це спосіб логічного висновку, запозичений з теорії ігор.

При прямому зчепленні дані приймаються до тих пір, поки не буде сформовано ланцюжок правил. Остання може бути послідовністю операторів if/then, як на рис. 4.18. Пряме зчеплення постійно шукає дані, що задовольняють одному з маршрутів if/then, щоб завершити дію. Зворотне зчеплення рухається у протилежному напрямку: від дії до даних, а не навпаки. Нижче наведено псевдокод, що демонструє просту систему правил:

```
Smoke Sensor = Smoke Detected
Heat Sensor = Heat Detected

if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == Heat_Detected) then Fire
if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor == Heat_Detected)
      thenFurnace_On
if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == !Heat_Detected) then
      Smoking
if (Fire) then Alarm
if (Furnace_On) then Log_Temperature
if (Smoking) then SMS_No_Smoking_Allowed
```

Моделюємо ситуацію. Уявіть, що:

Smoke_Sensor (датчик задимленості) вимкнений;

Heat_Sensor (датчик тепла) увімкнено.

Пряме зчеплення зупинилося на другій умові, роблячи висновок у тому, що температура записується до журналу.

Зворотне зчеплення намагається довести, що опалення увімкнено. Для цього воно рухається у протилежному напрямку:

1) Чи можемо ми довести, що температура записується до журналу? Погляньте на наступний код:

```
if (Furnace_On) then Log_Temperature
```

2) оскільки температура записується в журнал, умова (Furnace_On) стає новою метою:

```
if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor == Heat_Detected) then
    Furnace_On
```

3) ми вже довели, що опалення включене, тому нова умова складається з двох частин: Smoke_Sensor та Heat_Sensor. Система правил розбиває його на дві цілі:

```
Smoke_Sensoroff
Heat_Sensoron
```

4) тепер система правил намагається досягти обох цілей. Під час цього процесу завершується логічний висновок.

Пряме зчеплення дозволяє реагувати на нові дані з їх надходження, що може призвести до нових логічних висновків.

Семантична мова Drools була спеціально спрощена. Вона складається з таких базових елементів:

- сесії, що визначають правила за умовчанням;
- точки входу, що визначають правила, які використовуватимуться;
- умовні вирази (when);
- дії, що виконуються (then).

У наступному псевдокоді представлено просте правило в Drools. Операція insert виконує зміну робочої пам'яті. Зазвичай це робиться у разі, коли умова виявляється істинною.

```
rule "Furnace_On"
when
Smoke_Sensor(value &gt; 0) &&Heat_Sensor(value &gt; 0)
then
insert(Furnace_On())
end
```

Після виконання всіх правил програма може зробити запит до робочої пам'яті, щоб дізнатися, які умови виявилися дійсними. Для цього використовується наступний синтаксис:

```
query "Check_Furnace_On"
$result: Furnace_On()
end
```

Це правило має два шаблони:

- синтаксичний: формат даних, відповідність, хеш, діапазон значень;
- семантичний: значення має входити до списку, а лічильник значень з високою температурою не повинен перевищити 20 за годину. По суті це значні події.

Drools підтримує створення дуже складних та докладних правил, для зберігання яких може навіть знадобитися окрема база даних. Семантика мови передбачає шаблони, входження в діапазон, підрахунок кількості спрацьовувань правила, зіставлення типів та роботу з колекціями об'єктів.

Споживання інформації: потоки, обробка та озера даних

IoT-пристрій зазвичай з'єднаний з якимсь датчиком або іншим пристроєм, який вимірює чи відстежує умови у реальному світі. Це робиться асинхронно по відношенню до інших технологій Інтернету речей. Тобто, датчик намагається транслювати дані незалежно від того, слухає його хмарний або туманний вузол. Це важливо, оскільки в багатьох організаціях дані мають найбільшу цінність. Навіть якщо більшість інформації виявляється надмірною, у якийсь момент може статися важлива подія. У зв'язку з цим дані передаються як потоки.

Потік інформації, що передається з датчика в хмару, сприймається як:

- постійний та нескінченний;
- асинхронний;
- неструктурований чи структурований;
- максимально близький до режиму реального часу.

Раніше ми вже обговорювали часові затримки, властиві хмарам. Ми також дізналися, що туманні обчислення допомагають вирішити цю проблему. Але й без туманних вузлів робляться зусилля для оптимізації хмарної архітектури, щоб забезпечити підтримку режиму реального часу в Інтернеті речей. Для цього хмара має вміти працювати з безперервним потоком даних. Альтернативою є *пакетна обробка*. Більшість апаратних платформ використовують ті самі методи роботи з

потоками, переміщуючи дані з одного блоку в інший; при цьому на кожному етапі їх надходження активує наступну функцію. Важливим елементом зниження загальної латентності є ретельне використання сховища даних та доступ до файлової системи.

У зв'язку з цим більшість потокових фреймворків виконують свої операції в пам'яті, намагаючись повністю виключити тимчасове зберігання інформації у файловій системі. Важливість цього підходу наголошується у статті Майкла Стоунбрейкера, Угура Кетінтемела та Стена Здоніка «8 вимог до обробки потоків у режимі реального часу» (SIGMOD Rec. 34, 4, 2005). Цьому шаблону сприяє добре спроектована черга повідомлень.

Побудова успішної хмарної архітектури, здатної масштабуватися на сотні та мільйони вузлів, потребує ретельного планування. До того ж потоки даних рідко бувають ідеальними. У ситуації, коли сотні тисяч датчиків асинхронно передають свої показання, деякі дані будуть губитися (при втраті зв'язку з датчиком) мати некоректний формат (помилка при передачі) або надходити не в тому порядку (якщо хмара приймає інформацію з кількох джерел). Поточна система повинна як мінімум:

- масштабуватись при зростанні та сплесках кількості подій;
- надавати можливість публікації/підписки у своєму API-інтерфейсі;
- прагнути до мінімізації затримок;
- забезпечувати масштабування правил опрацювання;
- підтримувати озера та сховища даних.

Фонд Apache пропонує кілька відкритих проектів (з ліцензією Apache 2), які мають допомогти побудувати архітектуру для обробки потоків. Один із них, Apache Spark, обробляє дані у вигляді невеликих пакетів. Це вкрай зручно, коли об'єм пам'яті в кластері хмари обмежений (наприклад, < 1 Тб). Фреймворк Spark виконує свої операції у пам'яті, що знижує латентність і залежність від файлової системи. Пакетна обробка даних добре підходить при роботі з моделями машинного навчання. Підтримка пакетних даних реалізована в декількох моделях, таких як згорткова нейронна мережа (англ. Convolutional Neural Network). Ще однією альтернативою від Apache є проект Storm, який намагається максимально наблизити хмарні обчислення до режиму реального часу. На відміну від Spark, він має низькорівневий API-інтерфейс і обробляє дані у вигляді великих наборів подій, не розбиваючи їх у пакети. Це дозволяє досягти низької латентності (менше секунди).

Щоб скормлювати дані фреймворкам для обробки потоків, можна використовувати Apache Kafka або Flume. Apache Kafka – це черга

MQTT, яка приймає показання з різних IoT-датчиків та клієнтів, а потім передає їх у Spark або Storm. MQTT не займається буферизацією, тому, якщо з хмарою взаємодіють тисячі клієнтів, нам знадобиться якась система, яка накопичуватиме повідомлення з вхідного потоку. Це дозволить Kafka масштабуватися залежно від навантаження та належним чином реагувати на сплески подій. Kafka підтримує потоки на швидкості 100000 повідомлень в секунду. З іншого боку, ми маємо розподілену систему Flume, яка збирає, агрегує та переміщає дані з одного джерела до іншого; її простіше використовувати із коробки. Вона також має тісну інтеграцію з Hadoop. Flum масштабується не так добре, як Kafka, оскільки додавання нових споживачів вимагає зміни архітектури системи. Обидва фреймворки здатні зберігати потоки у пам'яті, не скидаючи їх на диск; але зазвичай це не вдалий підхід, так як показання датчиків, які ведуть передачу одночасно, доцільніше зберігати у вигляді, якомога ближчому до оригіналу.

Якщо наша IoT-система складається з тисяч чи мільйонів датчиків та кінцевих вузлів, у хмарному середовищі є сенс використовувати озеро даних. *Озеро даних* – це, по суті, величезне сховище, куди стікається необроблена і невідфільтрована інформація з багатьох джерел. На відміну від звичайних файлових систем, воно абсолютно плоске і не підтримує ієрархії, томів, директорій, файлів та папок. Для структурування вмісту кожного запису прикріплюється елемент метаданих (ярлик). Класичним прикладом озера даних є Apache Hadoop, хоча практично всі хмарні провайдери використовують цю модель у тому чи іншому вигляді.

Озеро даних відмінно підходить для Інтернету речей, дозволяючи зберігати інформацію незалежно від того, структурована вона чи ні. Всі дані в ньому вважаються цінними та зберігаються на постійній основі. Таке скупчення інформації є оптимальним для систем аналізу даних. Якість роботи алгоритмів залежить від обсягу даних, які вони споживають або використовують для навчання своїх моделей.

На рис. 4.19 представлено концептуальну архітектуру на основі традиційної пакетної та потокової обробки. Озеро даних заповнюється за допомогою екземпляра Kafka. Хмара Kafka надає пакетний інтерфейс до Spark та передає дані до сховища.

Компоненти використовують стандартні з’єднання, тому топологію цієї архітектури можна модифікувати кількома способами.

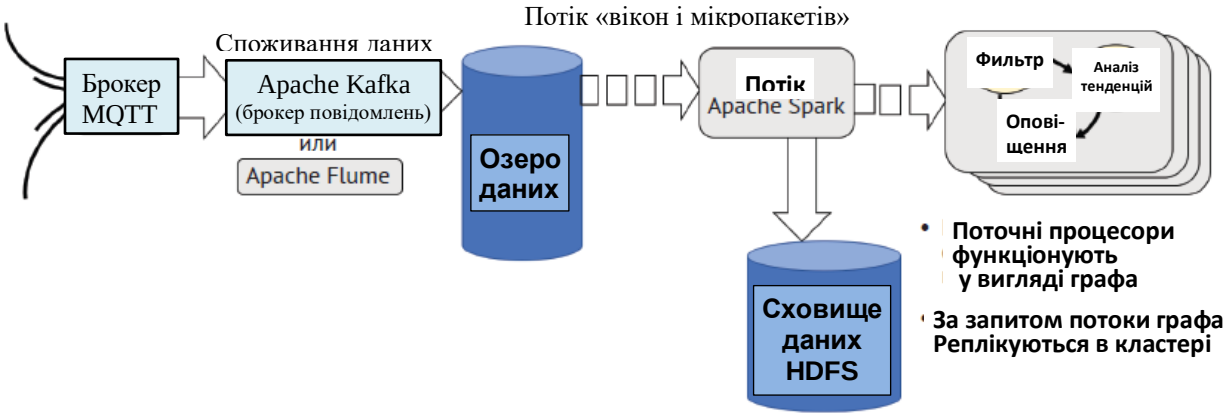


Рис. 4.19. Принцип подачі даних до хмарного сховища. Spark грає роль потокового сервісу

4.3.3. Обробка складних подій

Обробка складних подій (Complex event processing, або CEP) – це ще одна аналітична система, яка використовується для розпізнавання шаблонів.

У 1990-х роках. її почали використовувати у дискретно-подійному моделюванні та торгівлі на фондовому ринку. За своєю природою вона дозволяє аналізувати живі потоки даних у реальному часі. Сотні і тисячі подій, що надходять у систему, фільтруються та очищаються, перетворюючись на події вищого рівня – абстрактніші, ніж вихідні показання датчиків. Системи CEP забезпечують швидший аналіз навіть у порівнянні з потоковими процесорами, які можуть опрацювати подію в межах кількох мілісекунд. Але, на відміну від Apache Spark, системи CES мають меншу надмірність і здатність до динамічного масштабування.

Системи CES використовують SQL-подібні запити, проте задані шаблони або правила шукаються не в базі даних, а у вхідному потоці. CEP складається з кортежу (окремого елемента даних з часовою міткою) і використовує різні аналітичні шаблони, описані на початку цього розділу, які добре поєднуються з ковзними вікнами подій. За своєю семантикою мова запитів схожа на SQL, але працює значно швидше за звичайні бази даних, тому всі правила та дані зберігаються в пам'яті (зазвичай усередині багатогігабайтної БД). Крім того, дані повинні постачатися сучасною системою потокових повідомлень, як Kafka.

Системи CEP підтримують такі операції, як ковзні вікна, з'єднання та виявлення послідовностей. Крім того, як і системи правил, вони можуть покладатися як на пряме, так і зворотне зчеплення. Промисловим стандартом у цій галузі є система Apache WSO2 CEP, яка у поєднанні з Apache Storm здатна обробляти понад 1 мільйон подій на

секунду, не використовуючи при цьому постійне сховище. У системі WSO2 використовуються SQL-запити, але вона також підтримує скрипти на мовах JavaScript та Scala. Ще однією її перевагою є можливість розширення функцій за допомогою пакету під назвою Siddhi, який надає такі послуги:

геолокація;

обробка природної мови;

машинне навчання;

кореляція та регресія часових рядів;

математичні операції;

робота з рядками та RegEx.

Архітектор повинен розуміти, у яких ситуаціях слід застосовувати системи правил та CEP. Якщо умова оперує простими показаннями, такими як діапазони температур, система позбавлена стану і може бути реалізована з урахуванням простого механізму BRMS. Якщо система зберігає тимчасові дані або ряд станів, слід використовувати CEP.

Lamda-архітектура [1]

Lambda-архітектура намагається знайти компроміс між латентністю та пропускною спроможністю (рис. 4.20). На цьому рисунку пакетний рівень містить дані в сховищі HDFS, а потоковий рівень передає свої результати безпосередньо системі аналізу в режимі реального часу, використовуючи Spark. По суті Lambda-архітектура поєднує пакетну і потокову обробку. За аналогією із загальною хмарною топологією OpenStack та інших хмарних фреймворків Lambda зберігає споживані дані у репозиторії, доступному лише читання. Така топологія складається із трьох рівнів:

Пакетний рівень – зазвичай ґрунтується на кластерах Hadoop. Обробка у ньому відбувається набагато повільніше, ніж на потоковому рівні. Приносячи в жертву латентність, він максимізує пропускну здатність та точність.

Поточний рівень – це потік даних, які проходять через пам'ять у реальному часі. Дані можуть губитися, містити помилки та надходити не в тому порядку. Як ми бачили, Apache Spark надає відмінну систему обробки потоків.

Сервісний рівень – тут відбувається збереження, аналіз та візуалізація пакетних та потокових результатів. Зазвичай тут застосовуються такі компоненти: Druid (надає засоби об'єднання пакетного та потокового рівнів), Apache Cassandra (масштабована база даних) та Apache Hive (довготривале сховище даних).

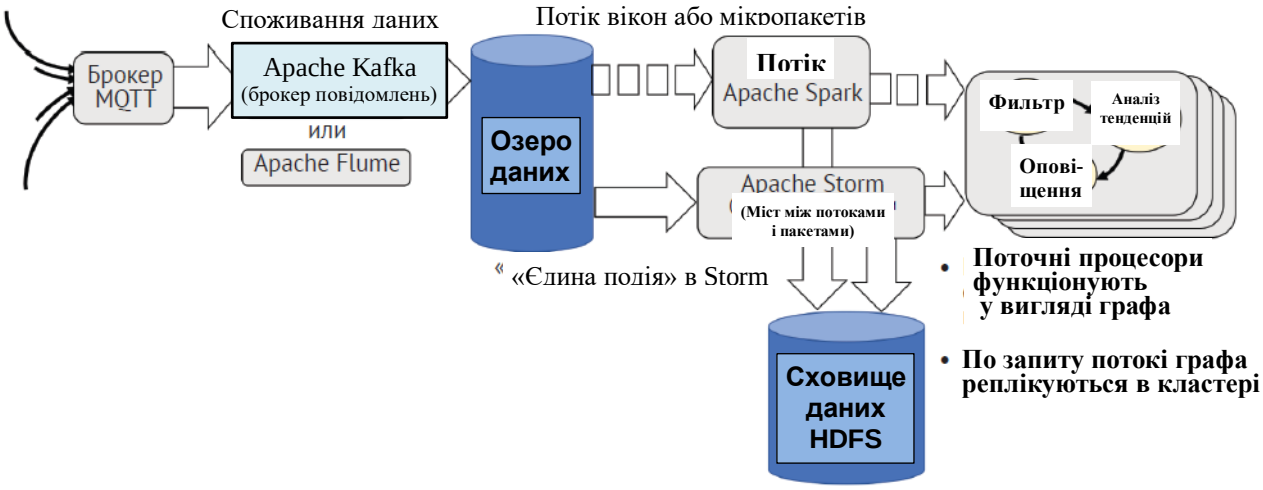


Рис. 4.20. Складнощі Lambda-архітектури [1]

Lambda-архітектури за своєю природою мають підвищену складність, якщо порівнювати їх з іншими аналітичними системами. Це гібридний підхід, для успішної реалізації якого потрібні додаткові ресурси.

Машинне навчання в Інтернеті речей

Останнім часом машинне навчання почали широко застосовувати для аналізу даних Інтернету речей. Однак оскільки методи машинного навчання детально вивчаються у рамках курсу «Системи штучного інтелекту», то у цьому посібнику їх розглядати не будемо.

4.4. Big Data в Інтернеті речей

Терміном **Big Data ("великі дані")** окреслюють групу технологій та методів, за допомогою яких аналізують та обробляють величезну кількість даних, як структурованих так і неструктурованих, для отримання якісно нових знань. Якщо підсумувати, то це інформація, що не піддається обробці класичними способами через її величезний об'єм.

4.4.1. Основні характеристики Big Data

Давайте детальніше розберемося, які характеристики мають дані, які можна віднести в категорію Big Data. Ось *п'ять ключових умов (характеристик), які ще називають "п'ять V"* [28]:

Volume (об'єм) – накопичена база даних охоплює настільки великий обсяг інформації, що його практично нереально обробляти та зберігати традиційними способами. Для них потрібен зовсім новий підхід та вдосконалені інструменти.

Velocity (швидкість) – ця характеристика вказує на швидкість накопичення даних, яка постійно збільшується. Наприклад, 90 відсотків всієї інформації, якою оперує людство, зібрано за останні два роки. Також ця характеристика має на увазі швидкість обробки даних. Останнім часом збільшується попит на технології, що дозволяють використовувати обробку даних в режимі реального часу.

Variety (різноманітність) – можливість одночасно обробляти структуровану та неструктуровану інформацію. Структурована інформація – це така, яку можна класифікувати. Наприклад, це може бути інформація з банківської бази даних, де чітко вказаний перелік клієнтів та їхні фінансові транзакції.

Неструктурована інформація охоплює різноманітні масиви даних, такі як фото, відео, текстові записи та інші дані. Найкращим прикладом є соціальні мережі. Її об'єм складає приблизно 80 відсотків від всієї інформації. Неструктурована інформація потребує комплексного аналізу перед можливістю її використання.

Veracity (достовірність) – оскільки обсяг інформації постійно збільшуються, важливе місце займає виокремлення достовірних даних. Якість зафіксованих даних може сильно відрізнятися, тим самим впливаючи на точний аналіз.

Variability (мінливість) – невідповідність інформації ускладнює та подекуди заважає процесам обробки та управління даними.

Виходячи з вищеназваних визначень, основні принципи роботи з великими даними такі:

Горизонтальна масштабованість. Це — базовий принцип обробки великих даних. Як вже було зазначено, великих даних з кожним днем стає все більше. Відповідно, необхідно збільшувати кількість обчислювальних вузлів, за якими розподіляються ці дані, при чому обробка має відбуватись без погіршення продуктивності.

Відмовостійкість. Цей принцип витікає з попереднього. Оскільки обчислювальних вузлів у кластері може бути багато (іноді десятки тисяч) та їх кількість, не виключено, буде збільшуватись, зростає ймовірність виходу машин з ладу. Методи роботи з великими даними мають враховувати ймовірність таких ситуацій і передбачати превентивні заходи.

Локальність даних. Оскільки дані розподілені по великій кількості обчислювальних вузлів, то, якщо вони фізично знаходяться на одному сервері, а обробляються на іншому, витрати на передачу даних можуть бути невиправдано великими. Тому обробку даних бажано проводити на тій же машині, на якій вони зберігаються.

Ці принципи відрізняються від тих, які характерні для традиційних, централізованих, вертикальних моделей зберігання добре

структурованих даних. Власне, для роботи з великими даними розробляються і інші підходи та технології.

Нескінченно великий інформаційний потік, який складає основу Big Data, дозволяє нам отримувати кардинально нові знання, які були недоступні ще кілька років тому.

4.4.2. Загальна характеристика процесу Data Science

Процес Data Science зазвичай складається з шести кроків, як видно зі схеми на рис. 4.21. Зараз ми коротко розглянемо основні пункти [28].

Рис. 4.21. Процес Data Science

1. Призначення мети дослідження.

Методи Data Science застосовується в основному в контексті організацій. Коли бізнес пропонує вам взятися за проект Data Science , ви спочатку готуєте проектне завдання. У проектному завданні описано, що ви збираєтеся досліджувати, яку користь це принесе компанії, які дані і ресурси вам будуть потрібні, представлений календарний план робіт і опис вихідних результатів.

2. Збір даних

На другому етапі процесу відбувається збір даних. У проектному завданні зазначено, які дані вам будуть потрібні і де їх можна знайти. На цьому етапі ви вживаєте заходів до того, щоб дані могли використовуватися у вашій програмі, що означає перевірку існування, якості та доступності даних. Дані також можуть надаватися третіми сторонами і існувати в різних формах, від таблиць Excel до різного роду баз даних.

3. Підготовка даних

Процес збору даних схильний до помилок; в цій фазі дослідник підвищує якість даних і готує їх до подальшого використання. Ця фаза складається з трьох підфаз: очищення даних видаляє некоректні значення з джерела даних і усуває розбіжності між джерелами, інтеграція даних розширює інформацію за допомогою об'єднання інформації з декількох джерел, а перетворення даних гарантує, що дані знаходяться в потрібному форматі для використання в ваших моделях.

4. Дослідження даних

Дослідження даних спрямоване на досягнення більш глибокого розуміння даних. Ви намагаєтесь зрозуміти, як змінні взаємодіють одна з одною, оцінити розподіл даних і визначити наявність викидів. Для цього в основному використовуються описові статистики, візуальні методи і просте моделювання. Цей крок часто позначається скороченням EDA від «Exploratory Data Analysis («дослідницький аналіз даних»).

5. Моделювання даних або побудова моделі

У цій фазі моделювання знання предметної області та інформація про дані, отримана на попередніх етапах, використовуються для отримання відповіді на питання дослідження. В ході моделювання використовуються методи з області статистики, машинного навчання, дослідження операцій і т. д. Побудова моделі є ітеративним процесом, в ході якого дослідник вибирає змінні для моделі, застосовує модель і проводить діагностику моделі.

6. Відображення і автоматизація

Нарешті, результати дослідження повинні бути представлені бізнес-стороні. Такі результати можуть існувати в різних формах, від презентацій до звітів по науково-дослідній роботі. Іноді виконання процесу доводиться автоматизувати, тому що бізнес-сторона хоче використовувати інформацію, отриману в іншому проекті, або задіяти робочий процес, який використовує результати вашої моделі.

Ітеративний характер процесу

Наведений опис процесу Data Science може створити враження, що процес має лінійний характер, але в дійсності доводиться часто повертатися назад і щось переробляти. Наприклад, в фазі дослідження даних можуть виявитися викиди, які вказують на помилки імпортування даних. В ході процесу Data Science ви отримуєте нову інформацію, яка може породити нові питання. Щоб вам не довелося переробляти вже виконану роботу, простежте за тим, щоб питання з боку бізнесу з самого початку були визначені досить ясно і докладно.

Отже, тепер суть процесу Data Science трохи прояснилася і ми можемо перейти до огляду методів.

4.4.3. Технології та тенденції роботи з Big Data

Big Data працює за принципом чим більшою кількістю інформації ми володіємо, тим точніший прогноз можливо зробити. Також можливість порівняння певних даних та взаємозв'язків між ними дозволяє знайти закономірності, які були приховані до цього. Все це забезпечує глибинне розуміння проблем та, в кінцевому результаті, дозволяє знайти рішення, або можливості керування потрібними процесами.

Найчастіше процес обробки великих об'ємів даних включає в себе побудову моделей та запуск симуляцій, під час яких постійно змінюються ключові налаштування, при цьому система постійно відслідковує, як ці зміни впливають на можливий результат. Це все відбувається в автоматичному режимі, допоки не буде знайдено ключовий момент, який допоможе вирішити поставлену задачу.

Оскільки переважна більшість даних є неструктурована, то для перетворення їх у такі, що сприйматимуться людьми, використовуються найсучасніші технології аналізу. До них можна віднести штучний інтелект та машинне навчання.

Початково у сукупність підходів і технологій включались засоби масово-паралельної обробки невизначено-структурованих даних, такі як СУБД NoSQL, алгоритми MapReduce і засоби проекту Hadoop. У подальшому до технологій великих даних почали відносити й інші рішення, що забезпечують схожі за характеристиками можливості обробки надвеликих масивів даних, а також деякі апаратні засоби.

MapReduce — модель розподілених обчислювань у комп’ютерних кластерах, представлена компанією Google. Згідно з цією моделлю, додаток розділяється на значну кількість однакових елементарних завдань, що виконуються на вузлах кластера і потім, природнім шляхом зводиться у кінцевий результат.

NoSQL (від англ. **Not Only SQL**, не лише SQL) — загальний термін для різних нереляційних баз даних і сховищ, не означає якусь конкретну технологію чи продукт. Звичайні реляційні бази даних добре підходять для досить швидких і однотипних запитів, а на складних і гнучко побудованих запитах, характерних для великих даних, навантаження перевищує розумні межі і використання СУБД стає неефективним.

Hadoop — набор утиліт, бібліотек і фреймворків, що вільно розповсюджується, для розробки і виконання розподілених програм, які працюють на кластерах із сотень і тисяч вузлів. Вважається однією з основоположних технологій великих даних.

R — мова програмування для статистичної обробки даних і роботи з графікою. Широко використовується для аналізу даних і фактично стала стандартом для статистичних програм.

Апаратні рішення. Корпорації Teradata, EMC та ін. пропонують апаратно-програмні комплекси, призначені для обробки великих даних. Ці комплекси поставляються як готові до установки телекомунікаційні шафи, що містять кластер серверів і керівне програмне забезпечення для масово-паралельної обробки. Сюди іноді відносять апаратні рішення для аналітичної обробки в оперативній пам'яті, зокрема, апаратно-програмні комплекси Hana компанії SAP і комплекс Exalytics компанії Oracle, незважаючи на те, що така обробка початково не є масово-паралельною, а об'єми оперативної пам'яті одного вузла обмежуються кількома терабайтами.

Консалтингова компанія McKinsey, окрім технологій NoSQL, MapReduce, Hadoop, R, які розглядає більшість аналітиків, включає у контекст придатності для обробки великих даних також технології Business Intelligence і реляційні системи управління базами даних з підтримкою мови SQL.

Міжнародна консалтингова компанія McKinsey, що спеціалізується на розв'язанні задач, пов'язаних зі стратегічним управлінням, виділяє 11 методів і технік аналізу, що застосовуються до великих даних [14].

Методи класу Data Mining (отримання даних, інтелектуальний аналіз даних, глибинний аналіз даних) — сукупність методів виявлення у даних раніше невідомих, нетривіальних, практично корисних знань, необхідних для прийняття рішень. До таких методів, зокрема, належать: навчання асоціативним правилам (association rule learning), класифікація (разгалуження на категорії), кластерний аналіз, регресійний аналіз, виявлення і аналіз відхилень тощо.

Краудсорсинг — класифікація і збагачення даних силами широкого, неозначеного кола особистостей, що виконують цю роботу без вступу у трудові стосунки.

Змішання та інтеграція даних (data fusion and integration) — набір технік, що дозволяють інтегрувати різнорідні дані з розмаїття джерел з метою проведення глибинного аналізу (наприклад, цифрова обробка сигналів, обробка природньої мови, включно з тональним аналізом).

Машинне навчання, включаючи навчання з учителем і без учителя – використання моделей, побудованих на базі статистичного аналізу чи машинного навчання для отримання комплексних прогнозів на основі базових моделей.

Штучні нейронні мережі, мережевий аналіз, оптимізація, у тому числі генетичні алгоритми (genetic algorithm — евристичні алгоритми пошуку, що використовуються для розв'язання задач оптимізації і

моделювання шляхом випадкового підбору, комбінування і варіації потрібних параметрів з використанням механізмів, аналогічних натуральному відбору у природі)

Розпізнавання образів – клас методів, що дозволяють підходити до задачі як до розпізнавання образів на зображеннях.

Прогнозна аналітика – клас методів, що за допомогою пошуку аналітичних залежностей дозволяє передбачити результат.

Імітаційне моделювання (simulation) — метод, що дозволяє будувати моделі, які описують процеси так, як вони б проходили у дійсності. Імітаційне моделювання модна розглядати як різновид експериментальних випробувань.

Просторовий аналіз (spatial analysis) — клас методів, що використовують топологічну, геометричну і географічну інформацію, що вилучається із даних.

Статистичний аналіз — аналіз часових рядів, A/B-тестування (A/B testing, split testing) — метод маркетингового дослідження; при його використанні контрольна група елементів порівнюється із набором тестових груп, у яких один чи кілька показників були змінені, щоб з'ясувати, які зі змін покращують цільовий показник.

Візуалізація аналітичних даних — подання інформації у вигляді малюнків, діаграм, з використанням інтерактивних можливостей і анімації, як для отримання результатів, так і для використання у якості вихідних даних для подальшого аналізу. Дуже важливий етап аналізу великих даних, що дозволяє показати найважливіші результати аналізу у найбільш зручному для сприйняття вигляді.

Контрольні питання до розділу 4

1. Які види хмар зараз виділяють?
2. Які моделі хмарних сервісів існують на сьогодні?
3. Пояснити особливості хмарної архітектури OpenStack.
4. Пояснити які є обмеження хмарних архітектур для IoT?
5. Виділити особливості туманних обчислень в порівнянні з хмарними.
6. Пояснити особливості архітектури OpenFog RA.
7. Які існують типи даних для аналізу в Інтернеті речей?
8. Що включає простий аналіз даних в Інтернеті речей?
9. Виділити основні характеристики Big Data.
10. Дати загальну характеристику процесу Data Science.

5. ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ СИСТЕМ IoT

Незважаючи на те, що проблеми безпеки в області інформаційних технологій самі по собі не нові, багато прикладів впровадження IoT ставлять перед нами нові унікальні проблеми в області безпеки. Вирішення цих проблем і забезпечення безпеки в області продуктів і послуг IoT має бути основним пріоритетом. Користувачі повинні бути впевнені в тому, що пристрої IoT і пов’язані з ними послуги в області даних не мають вразливих місць, особливо в міру поширення цієї технології і її інтеграції в наше повсякденне життя. Недостатньо захищені послуги та пристрої IoT можуть використовуватися як потенційні точки доступу для кібератак і відкривати можливість розкрадання даних у зв'язку з відсутністю необхідного захисту потоків даних [29].

Взаємопов'язаність пристроїв IoT означає, що кожен недостатньо захищений пристрій, підключений до Інтернету, може вплинути на загальний рівень безпеки і стійкості системи. Ця проблема посилюється іншими факторами, такими як масове розгортання пристроїв IoT з однорідною структурою, здатність деяких пристроїв автоматично підключатися до інших пристроїв, а також можливість використання цих пристроїв в незахищеному середовищі.

Згідно із загальним правилом, розробники і користувачі пристроїв та систем IoT несуть колективне зобов'язання забезпечити захист користувачів і Інтернету від потенційних загроз. У зв'язку з цим, для створення ефективних і правильних методів захисту IoT, що відповідають масштабам і рівням складності цих загроз, необхідний спільний підхід на основі співпраці.

5.1. Проблеми безпеки IoT

Забезпечення безпеки, надійності, стійкості і стабільності додатків і послуг Інтернету речей, має критично важливе значення для довіри і використання Інтернету.

Як користувачі Інтернету, ми повинні мати високий ступінь впевненості в тому, що Інтернет, його програми та підключені до нього пристрої мають досить високий ступінь безпеки для виконання різних завдань по відношенню до допустимого ризику, пов'язаного з їх виконанням. Інтернет речей нічим не відрізняється в цьому відношенні, і безпека IoT пов'язана, в основному, з довірою до середовища з боку користувачів. Якщо люди не вірять в захищеність підключених пристроїв і отриманої інформації від неприпустимого використання, то

цей недолік довіри призводить до відмови від використання Інтернету. Цей фактор робить глобальний вплив на електронну комерцію, технічні інновації, свободу висловлювань і практично всі інші аспекти онлайн діяльності. Забезпечення безпеки продуктів і послуг IoT має бути основним пріоритетом в даній галузі.

У міру постійного збільшення числа пристроїв, підключених до Інтернету, виникають нові потенційно вразливі місця. Недостатньо захищені пристрої можуть служити точками доступу для кібератак, дозволяючи зловмисникам перепрограмувати пристрій або викликати його несправність. Пристрої недосконалої конструкції можуть піддавати дані користувачів небезпеці розкрадання за рахунок недостатнього захисту потоків даних. Несправні або дефектні пристрої також можуть створювати вразливі точки. Для поширених недорогих пристроїв невеликого розміру ці проблеми стоять настільки ж гостро або навіть ще гостріше, ніж для комп'ютерів, які традиційно використовувалися для підключення до Інтернету. Конкурентоспроможна вартість і технічні обмеження пристроїв IoT змушують виробників вбудовувати в ці пристрої відповідні функції безпеки, щоб забезпечити рівень безпеки і довгострокового захисту вразливих місць, що перевищує аналогічні показники комп’ютерів [30].

Крім потенційних вразливих місць, суттєве збільшення кількості і типів пристроїв IoT також може сприяти збільшенню ймовірності кібератак. З урахуванням функції взаємопідключення пристроїв IoT, кожен підключений пристрій, що не має достатнього захисту, має потенційно негативний вплив на безпеку і стійкість Інтернету в глобальному масштабі, а не тільки локально. Наприклад, незахищений холодильник в США, заражений шкідливим програмним забезпеченням, може відправляти тисячі шкідливих повідомлень електронної пошти одержувачам у всьому світі за допомогою домашнього підключення Wi-Fi.

І на довершення всього, в гіперпідключеному світі наша здатність виконувати щоденні завдання без допомоги пристроїв або систем з підключенням до Інтернету, буде знижуватися.

Зараз стає все важче придбати пристрої без підключення до Інтернету, тому що деякі виробники виготовляють тільки підключаємі продукти. Кожен день ступінь нашої підключеності зростає і ми стаємо все більш залежними від пристроїв IoT для виконання основних завдань. Необхідно, щоб пристрої були захищеними, з урахуванням того, що ніякий пристрій не може бути повністю безпечним. Цей зростаючий рівень залежності від пристроїв IoT та інтернет-послуг, з якими вони взаємодіють, також відкриває зловмисникам можливості доступу до

пристроїв. Припустимо, ми можемо відключити підключений до Інтернету телевізор, якщо він піддасться кібератаці, але ми не зможемо також просто вимкнути електролічильник або систему регулювання руху транспорту або імплантований кардіостимулятор.

Саме тому **безпека пристроїв і послуг IoT** є основою темою обговорень і повинна бути визнана **критично важливою проблемою**. Ми все більшою мірою залежимо від цих пристроїв для виконання важливих повсякденних завдань, і їх поведінка може мати глобальний вплив [31].

5.1.1. Загальні аспекти безпеки IoT

Безпека Інтернету речей – це сектор інформаційних технологій, який фокусується на захисті кінцевих пристроїв, мереж та даних, що стосуються Інтернету речей – підключених пристроїв, які не є комп'ютерами, смартфонами або планшетами. **По суті, безпека Інтернету речей** – це широкий термін, що охоплює стратегії, політики, процеси та технології безпеки, які компанії використовують для захисту своїх пристроїв Інтернету речей від інтелектуальних холодильників та камер відеоспостереження до моніторів на реактивних двигунах або автомобілях та пов'язаних з ними даних, додатків та мережі від злому або іншої шкоди.

Говорячи про пристрої, підключені до Інтернету речей, необхідно розуміти, що їх безпека не є абсолютною. Безпека пристрою IoT не визначається бінарним поняттям захищеності або незахищеності. Безпеку IoT слід розглядати швидше як діапазон уразливості пристрою. Цей діапазон охоплює як незахищені пристрої, що не мають функцій безпеки, до надзвичайно безпечних систем з декількома рівнями захисту. У цій нескінченній грі в кішки-мишки постійно виникають нові загрози безпеці, і виробники пристроїв і мережеві оператори постійно вживають заходів для захисту від цих загроз.

Безпека і стійкість Інтернету речей – це ефективність оцінки ризиків та їх усунення. Безпека пристрою – це функція управління ризиком того, що пристрій буде зламано, з урахуванням збитків, які виникли в результаті, а також часу і ресурсів для забезпечення необхідного рівня захисту. Для користувачів, які не можуть дозволити собі високий ступінь ризику для безпеки, як у випадку оператора системи регулювання руху транспорту або людини з медичним пристроєм, підключеним через Інтернет, може бути доцільно витратити чимало часу і ресурси для захисту системи або пристрою від кібератак.

Аналогічним чином, якщо користувача не турбує можливість злому його холодильника і його використання для відправки спаму, він

не захоче платити за більш складну систему безпеки, якщо вона збільшить вартість пристрою або зробить його більш складним в роботі.

На цю оцінку ризиків і можливих наслідків впливає цілий ряд факторів. Ці чинники включають в себе наявність чіткого розуміння існуючих ризиків безпеки і потенційних ризиків в майбутньому; Приблизні економічні та інші наслідки в разі здійснення ризиків; А також приблизну вартість усунення їх наслідків.

Незважаючи на те, що це співвідношення плюсів і мінусів в області безпеки часто розглядається з точки зору окремого користувача або організації, необхідно також враховувати взаємозв'язок пристроїв IoT, як частини більш великої екосистеми IoT. Функція мережевого підключення пристроїв IoT означає, що рішення в області безпеки, прийняті на місці по відношенню до будь-якого пристрою IoT, можуть мати глобальний вплив на інші пристрої.

З принципової точки зору, розробники інтелектуальних предметів для Інтернету речей зобов'язані гарантувати, що ці пристрої не будуть піддавати небезпеці свого власника або інших людей. З точки зору бізнесу і економіки виробники зацікавлені в зменшенні витрат, зниженні рівня складності і скорочення часу до випуску на ринок. Наприклад, стають все більш поширеними пристрої IoT, що представляють собою компоненти масового випуску з низьким рівнем прибутку, самі по собі є додатковою вартістю продукту, в який вони вбудовані. Нарощування обсягу пам'яті або установка швидшого процесора може зробити ці продукти неконкурентоспроможними з комерційної точки зору.

З точки зору економіки, недостатній рівень безпеки пристроїв IoT призводить до негативних зовнішніх наслідків, де витрати покладаються однією стороною (або сторонами) на інші. Класичним прикладом є забруднення навколишнього середовища, де збиток і витрати на очистку (негативні зовнішні наслідки) в результаті дій порушників несуть інші сторони. Проблема в тому, що витрати на вирішення зовнішніх проблем, що покладаються на інших, як правило, не враховується в процесі прийняття рішень, за винятком тих випадків, коли, як у випадку із забрудненням, на порушника накладається штраф з метою змусити його знизити рівень забруднень. У разі інформаційної безпеки, як вказується в описі Брюса Шнайера, зовнішні проблеми виникають в тих випадках, коли виробник товару не несе відповідальність за витрати в результаті недостатнього рівня безпеки. У цьому випадку закон про відповідальність може змусити виробників брати до уваги зовнішні проблеми і випускати продукти з більш високим рівнем безпеки.

Таким чином, пристрої з підтримкою Інтернету створюють ряд проблем безпеки. Але хоча Інтернет речей дає можливість підключення

до нових пристроїв, загальні проблеми кібербезпеки не є новими (рис. 5.1) [32].

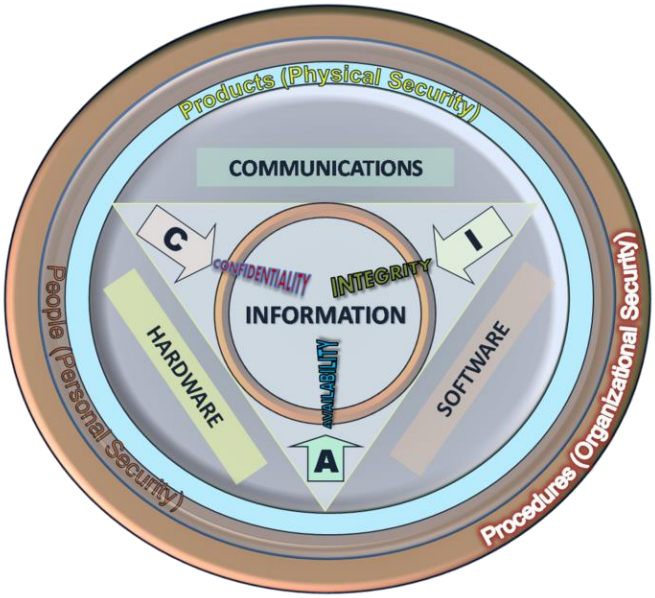


Рис. 5.1. Загальні види загроз IoT

Загрозами інформаційної безпеки «розумного будинку» є порушення конфіденційності, цілісності та доступності інформації.

Ці вимоги безпеки не є новими в області інформаційних технологій, але масштаб унікальних проблем, які можуть виникнути з впровадженням IoT, робить їх вельми істотними.

5.1.2. Унікальні проблеми безпеки пристроїв IoT

Пристрої IoT зазвичай відрізняються від традиційних комп'ютерів і обчислювальних пристроїв в дуже важливих аспектах, які становлять загрозу безпеці [33]:

- Багато пристроїв, що підключені до Інтернету речей, такі як датчики і предмети побутової техніки, призначені для масового розгортання, порівнялися з числом традиційних пристроїв, підключених до Інтернету. У результаті потенційна кількість взаємних підключень між цими пристроями є безпрецедентною. Крім того, багато з цих пристроїв зможуть самостійно встановлювати зв'язок один з одним непередбачуваним і динамічним способом. Тобто, може знадобитися переглянути існуючі інструменти, методи і стратегії, пов'язані з безпекою IoT.

- Багато систем IoT будуть складатися з груп ідентичних або майже ідентичних пристроїв. Така однорідність підсилює потенційний вплив кожної уразливості, множачи його на кількість пристроїв, що мають ті ж характеристики. Наприклад, вразливість протоколу зв'язку

лампочок бренду однієї компанії з підключенням до Інтернету може поширитися на всі марки і моделі пристроїв, що використовують той же протокол або мають аналогічну конструкцію.

- Розгортання багатьох пристроїв, підключених до Інтернету речей, буде здійснюватися з урахуванням терміну експлуатації, що набагато років перевищує звичайні терміни для високотехнологічного обладнання. Розгортання цих пристроїв може здійснюватися в умовах, що ускладнюють або роблять неможливою їх модернізацію або зміну конфігурації; Або ці пристрої можуть пережити свого виробника і залишитися без технічної підтримки в довгостроковій перспективі. Такі сценарії демонструють те, що механізми безпеки, що працюють в момент розгортання, можуть бути непридатні для всього терміну служби пристроїв у міру появи нових загроз. В результаті, це може привести до появи вразливостей, які будуть зберігатися протягом тривалого часу. Це йде врозріз з парадигмою традиційних комп'ютерних систем, модернізація яких здійснюється шляхом оновлень операційної системи протягом усього терміну служби комп'ютера для усунення загроз безпеки. Технічна підтримка в довгостроковій перспективі і управління пристроями IoT представляють собою серйозну проблему безпеки.

- Багато пристроїв IoT з самого початку не припускають можливості поновлення або ця процедура занадто незручна і непрактична. Як приклад можна взяти відзив 1,4 млн автомобілів Fiat Chrysler в 2015 році для усунення вразливості, завдяки якій зловмисник зміг зламати автомобіль за допомогою бездротової мережі. Ці автомобілі необхідно передати дилеру Fiat Chrysler для установки оновлень вручну, або власник автомобіля повинен встановити оновлення самостійно за допомогою ключа USB. Правда полягає в тому, що ці оновлення з великою ймовірністю НЕ будуть встановлені на значний відсоток таких автомобілів, так як процес оновлення незручний для користувачів, в результаті чого вони постійно наражаються на небезпеку кібератак, особливо якщо автомобіль у всіх інших аспектах працює справно.

- Багато пристроїв IoT працюють таким чином, що користувач не має або майже не має уявлення про внутрішнє функціонування пристрою або створюваних ним потоках даних. Це створює вразливість в області безпеки, коли користувач вважає, що пристрій IoT виконує певні функції, в той час як насправді він може виконувати небажані дії або збирати дані, які користувач не мав наміру надавати. Функції пристрою також можуть змінюватися без попередження при оновленні,

в результаті чого користувач наражається на небезпеку в результаті будь-яких змін, що вносяться виробником.

- Деякі пристрої IoT встановлюються в таких місцях, де важко або навіть неможливо забезпечити їх фізичну безпеку. Зловмисники можуть отримати прямий фізичний доступ до пристрою. У зв'язку з цим, для забезпечення безпеки необхідні функції захисту від злому і інші інновації.

- Деякі пристрої IoT, такі як датчики стану навколишнього середовища, непомітно вбудовуються в елементи оточення, де користувач не помічає пристрій і не може контролювати його роботу. Крім того, пристрої можуть не мати функції попередження користувача про виникнення проблем безпеки, в результаті чого користувач може не знати про наявність загрози безпеці в пристрої IoT.

Уразливість може зберігатися протягом тривалого часу до того, як вона буде помічена і виправлена, в тому випадку, якщо виправлення або пом'якшення наслідків є в принципі неможливо або недоцільно. Аналогічним чином, користувач може навіть не знати, що поблизу є датчик, в результаті чого уразливість може залишатися непоміченою протягом довгого часу.

- Перші моделі, підключені до Інтернету речей, засновані на передумові, що IoT буде продуктом великих приватних і/або державних технологічних підприємств, але в майбутньому створення свого власного Інтернету речей (BYIoT) може стати звичайною практикою, як це демонструють спільноти розробників, що розвиваються, Arduino і Raspberry Pi. І в цьому випадку передові галузеві стандарти безпеки можуть не завжди застосовуватися.

5.2. Проблеми конфіденційності в IoT

Повне розкриття потенціалу Інтернету речей залежить від вибору стратегій, що враховують право на конфіденційність відповідно до широкого спектру застосувань. Потоки даних і специфічна інформація про користувачів, що користуються пристроями IoT, можуть мати неймовірну цінність для інших користувачів IoT (чи зловмисників), тому проблеми конфіденційності і потенційних зловживань можуть перешкоджати повному впровадженню Інтернету речей. Це означає, що конфіденційність і повага до права на конфіденційність мають ключове значення для завоювання довіри користувача по відношенню до Інтернету, підключених пристроїв і пов'язаним з ними послугами [34].

Інтернет речей виводить на новий рівень полеміку про конфіденційність, так як широке застосування може в корені змінити

методи збору, аналізу, застосування і захисту особистих даних. Наприклад, IoT підсилює занепокоєння ймовірністю додаткового спостереження і стеження, неможливістю відмовитися від надання деяких даних і можливістю об'єднання декількох потоків даних IoT для створення докладних цифрових описів користувачів. Це важливі проблеми, але вони не є нерозв'язними. Щоб зрозуміти все, що відкриваються, необхідно розробити стратегії обліку особистих переваг щодо конфіденційності відповідно до очікувань користувачів, при цьому продовжуючи розвивати новаторські технології та послуги.

5.2.1. Загальні аспекти конфіденційності Інтернету речей

Дотримання права на недоторканність приватного життя і переваг конфіденційності є невід'ємною частиною вирішення проблеми довіри до Інтернету, і воно також впливає на можливість людей говорити, підключатися, вибирати - і робити це продуктивно. Ці права і очікування іноді зводяться до проблеми етичної обробки даних, підкреслюючи важливість задоволення очікувань дотримання прав конфіденційності і сумлінного використання даних.

Інтернет речей здатний поставити під сумнів ці традиційні очікування дотримання прав приватного життя [35].

Проблеми конфіденційності, що виникли з появою Інтернету речей, дуже важливо вирішити, так як вони стосуються основних прав людини і здатності нашого суспільства довіряти Інтернету і підключеним до нього пристроям.

Інтернет речей часто представляється масштабною мережею сенсорних пристроїв, які збирають дані про оточення і нерідко - про людей. Звичайно, ці дані можуть бути корисними для власників пристроїв, але дуже часто вони представляють інтерес і для виробників і постачальників пристроїв.

Збір і використання IoT-даних перетворюється на справжню проблему конфіденційності, коли уявлення людей, що знаходяться під наглядом IoT-пристроїв, про масштаб і використання даних, відрізняються від міркувань збирача даних.

Здавалося б нешкідливі комбінації потоків IoT-даних також можуть загрожувати конфіденційності. При об'єднанні або зіставленні кількох потоків даних іноді можна отримати більш точний цифровий портрет людини, ніж при використанні одного потоку IoT-даних. Наприклад, підключена до Інтернету зубна щітка може записувати і передавати нешкідливі дані про те, як її власник чистить зуби. Але якщо

його холодильник передає дані про те, що він їсть, а фітнес-трекер передає дані про його фізичні активності, то комбінація цих потоків дозволяє отримати більш детальний і точний опис загального стану здоров'я цієї людини. Цей ефект групування даних може бути особливо справедливим по відношенню до IoT-пристроїв, так як багато пристроїв генерують додаткові метадані (наприклад, час і місце розташування), які дозволяють отримати більш конкретну інформацію про людину.

В інших ситуаціях користувач може не знати, що IoT-пристрій збирає дані про нього і здатен передавати їх третім сторонам. Цей тип збору даних отримує все більш широке поширення в області побутових пристроїв, таких як «розумні телевізори» і ігрові приставки. Такі пристрої оснащені функцією розпізнавання голосу та зображення і тому можуть безперервно слухати або переглядати те, що відбувається в приміщенні і активно передавати ці дані в хмарний сервіс для подальшої обробки, і в цьому процесі іноді задіяні треті сторони.

Людина може перебувати в оточенні подібних пристроїв, не підозрюючи про те, що його розмови або дії відстежуються, а дані записуються. Такого роду функції можуть не тільки приносити користь обізнаним користувачам, але і створювати проблеми конфіденційності тим, хто не підозрює про присутність цих пристроїв і не може контролювати використання зібраних даних.

Незалежно від того, чи відомо це користувачеві і чи згоден він з тим, що його дані збираються і аналізуються, подібні ситуації лише підкреслюють цінність персоналізованих потоків даних для компаній і організацій, які прагнуть збирати і записувати IoT-дані. Потреба в цих даних призводить до появи юридичних і нормативних проблем, пов'язаних з законами про захист і конфіденційність даних.

Ці проблеми конфіденційності важливо вирішити, так як вони впливають на основні права людини і її здатність довіряти Інтернету.

В цілому люди усвідомлюють, що їхнє приватне життя дійсно представляє цінність, і у них є очікування у тому, що стосується збору та використання даних третіми сторонами. Це загальне уявлення про недоторканність приватного життя стосується і даних, що збираються IoT-пристроями, але ці пристрої можуть загрожувати можливості користувача висловлювати і домагатися дотримання його прав на приватне життя. Якщо користувач втратить довіру до Інтернету через недотримання його прав на приватне життя в Інтернеті речей, загальна цінність Інтернету може зменшитися.

5.2.2. Унікальні аспекти конфіденційності Інтернету речей

В цілому проблеми конфіденційності посилюються тим, що Інтернет речей значно розширює можливості і доступність відстеження і спостереження. Характеристики IoT-пристроїв і методи їх використання направляють дискусії про проблеми конфіденційності в нове русло, так як вони серйозно змінюють методи збору, аналізу, використання і захисту персональних даних [36].

Приклад: Традиційна Інтернет-модель «ознайомлення і згоди» з політиками конфіденційності, де користувачі висловлюють своє ставлення шляхом інтерактивної взаємодії з інформацією на комп'ютерному або мобільному екрані (наприклад, шляхом натискання кнопки «Прийняти»), не спрацьовує, коли системи не пропонують механізму взаємодії користувача з системою.

IoT-пристрої часто не мають призначеного для користувача інтерфейсу для зміни налаштувань конфіденційності, і в багатьох випадках користувачі не знають або не вміють контролювати збір або використання особистих даних. Це створює розрив між власними уподобаннями і діями IoT-пристроїв зі збору даних. Постачальники IoT-пристроїв можуть не бути зацікавлені в наданні користувачам механізму для вираження своїх переваг, якщо вважають, що зібрані дані не носять особистого характеру. Однак досвід показує, що дані, які звичайно не вважаються особистими, насправді можуть стати такими, якщо їх об'єднати з іншими даними.

Якщо ми зможемо розробити ефективний механізм для IoT-пристроїв, що дозволяє користувачеві усвідомлено висловлювати свої переваги конфіденційності, цей механізм повинен охоплювати велику кількість IoT-пристроїв, які повинен контролювати користувач. Нереально очікувати, що користувач буде безпосередньо взаємодіяти з кожним IoT-пристроєм з яким він стикається протягом дня, щоб висловити свої переваги конфіденційності. Замість цього механізми інтерфейсу конфіденційності повинні підлаштовуватися під розмір проблеми IoT, але при цьому залишатися досить функціональними і практичними з точки зору користувача.

Інтернет речей може змінити очікування людини про дотримання прав на приватне життя в звичайних ситуаціях. Існують соціальні норми і очікування в тому, що стосується прав на приватне життя, що відрізняються в громадських місцях і особистих просторах, і IoT-пристрої ставлять ці норми під сумнів.

Наприклад, IoT-технології спостереження (наприклад, камери спостереження або системи відстежування місцеположення), що

встановлюються зазвичай в громадських місцях, переходять в традиційно особисті простори (наприклад, в будинок або особистий автомобіль), де очікування до дотримання прав на приватне життя зовсім інші. При цьому ставиться під питання те, що в багатьох суспільствах розглядається як «право на усамітнення» в будинку або особистому просторі.

Крім того, права на приватне життя в місцях, які вважаються громадськими (парки, магазини, вокзали і т.д.), ставляться під сумнів через збільшення природи і масштабу спостереження в цих місцях.

IoT-пристрої часто використовуються в умовах, де багатолюдність призводить до збору одних і тих же даних про багатьох людей. Наприклад, геолокаційний датчик в автомобілі записує дані про місце розташування всіх, хто знаходиться в автомобілі, незалежно від їх бажання. Він може відслідковувати навіть людей в сусідніх автомобілях. У подібних ситуаціях буває важко або неможливо розмежувати або врахувати індивідуальні переваги конфіденційності.

Аналітика великих даних, що застосовується до згрупованим персональних даних, вже почала представляти ризик втручання в приватне життя і потенційної дискримінації. Цей ризик посилюється в Інтернеті речей за рахунок масштабу і більшого впізнавання зібраних особистих даних. IoT-пристрої можуть збирати дані про людину з безпрецедентною точністю і нав'язливістю; угруповання і кореляція цих даних може створити детальний портрет індивідуума, що створює можливості для дискримінації та нанесення іншої шкоди. Витонченість цієї технології може привести до ситуацій, що загрожують здоров'ю, благополуччю, фінансів або репутації людини.

Повсюдність, близькість і соціальне охоплення безлічі IoT-пристроїв може вести до помилкового почуття захищеності і змушувати людей розголошувати секретні або особисті дані, не усвідомлюючи і не оцінюючи в повній мірі можливих наслідків своїх дій.

Проблеми конфіденційності можуть виявитися занадто складними для вирішення, навіть якщо повністю врахувати інтереси і мотиви всіх учасників екосистеми Інтернету речей. Але ми знаємо, що можуть існувати незбалансовані або недобросовісні відносини і інтереси між тими, чиї персональні дані збирають, і тими, хто збирає, аналізує і використовує ці дані. Джерело даних може здійснювати небажане вторгнення в приватне життя людини, часто без її згоди, вибору або відома. Але особа, що збирає ці дані, може вважати це корисним ресурсом, здатним збільшити дохід від продуктів і послуг, а також запропонувати нові джерела доходу.

Таким чином, проблема конфіденційності в IoT – це складно проблема яка потребує свого вирішення у кожному конкретному випадку розробки системи IoT.

Контрольні питання до розділу 5

1. Пояснити загальні аспекти безпеки IoT?
2. Пояснити унікальні проблеми безпеки пристроїв IoT?
3. Пояснити загальні аспекти конфіденційності Інтернету речей?
4. Пояснити унікальні аспекти конфіденційності Інтернету речей?

ЗАКІНЧЕННЯ

Можливості, які надає інтернет речей, є неймовірними: саме інтернет речей буде основою для наступного глобального перевороту в промисловості, охороні здоров'я, системі державного управління та підприємництва. Він, безперечно, вплине на ВНП, найману працю і ринки по всьому світу. Вже зараз з тисяч оголошень про роботу на сайтах багато які призначені для IoT-архітекторів, технічних фахівців та координаторів, здатних створювати IoT-рішення, а не просто віджети.

Інтернет речей продовжує стрімко розвиватися і знаходить все більше і більше використання у повсякденному житті. Створються нові консорціуми та промислові об'єднання, які приносять суспільству відчутну користь – розробляють стандарти, технічні регламенти та домагаються інтероперабельності. Вони вносять свої різні нові протоколи та стандарти і конкурують один з одним за своє місце в IoT-просторі. Тому і не дивно що через кілька років зміст цього навчального посібника доведеться оновлювати.

СПИСОК ЛІТЕРАТУРИ

1. Ли П. Архитектура интернета вещей / пер. С анг. М. А. Райтмана. – М.: ДМК Пресс, 2019. – 454 с.: ил
2. Росляков, А.В. Интернет вещей: учебное пособие [текст] / А.В. Росляков, С.В. Ваняшин, А.Ю. Гребешков. – Самара: ПГУТИ, 2015. – 200 с.
3. Муромцев Д.И., Шматков В.Н. «Интернет Вещей: Введение в программирование на arduino» – СПб: Университет ИТМО, 2018. – 36 с.
4. Боцман Ярослав. Интернет вещей. CHANNELFORIT REVIEW №3/2016 с.26-30.
5. Енциклопедія сучасної України. Концепція. Електронний ресурс. Режим доступу: http://esu.com.ua/search_articles.php?id=3256. Дата звертання 07.10.2020 року.
6. Вікіпедія. Концепція. Електронний ресурс. Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%86%D1%96%D1%8F. Дата звертання 07.10.2020 року.
7. Вікіпедія. Інтернет речей. Електронний ресурс. Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D1%80%D0%B5%D1%87%D0%B5%D0%B9 Дата звертання 07.10.2020 року.
8. Википедия. Интернет вещей. Электронный ресурс. Режим доступа: https://ru.wikipedia.org/wiki/Internet_of_things . Дата звертання 07.10.2020 року.
9. Международный союз электросвязи. МСЭ-Т. Рекомендации Y.2069. Серия Y: глобальная информационная инфраструктура, аспекты межсетевого протокола и сети последующих поколений. Термины и определения для интернета вещей (07/2012) 14 с.
10. Международный союз электросвязи. МСЭ-Т. Рекомендации Y.2060. Серия Y: глобальная информационная инфраструктура, аспекты межсетевого протокола и сети последующих поколений. Обзор интернета вещей. (06/2012) 16 с.
11. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: навч. посіб. Для студ. спеціальності 126 «Інформаційні системи та технології», спеціалізація «Інформаційне забезпечення робототехнічних систем» / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.
12. Приемышев А. В., Крутов В. Н., Треяль В. А. Технологии

создания интеллектуальных устройств, подключенных к интернет / А. В. Приемышев, В. Н. Крутов, В. А. Треяль. – Лань, 2017. – 100 с.
13. Интернет вещей. Исследования и область применения: монография / Е.П. Зараменских, И.Е. Артемьев. - М.: НИЦ ИНФРА-М, 2015. - 200 с.: 60x90 1/16. http://znanium.com/catalog.php?bookinfo=526946
14. Tripathy B. Internet of Things (IoT): TeChnologies, AppliCations, Challenges and Solutions (англ.) / B. Tripathy, J. Anuradha. – Florida: CRC Press, 2017. – 334 с.
15. Charles Bell. Beginning IoT Projects: Breadboard-less Electronic Projects. Après. Warsaw, VA, USA. 2021 – 870 p. ISBN-13 (pbk): 978-1-4842-7233-6 ISBN-13 (electronic): 978-1-4842-7234-3. https://doi.org/10.1007/978-1-4842-7234-3
16. Котюк А. Ф. Датчики в современных измерениях. – М.: Радио и связь, Горячая линия – Телеком, 2006. – 96 с.
17. Назаров А. В., Козырев Г. И., Шитов И. В. И др. Современная телеметрия в теории и на практике. Учебный курс. – СПб.: Наука и Техника, 2007. – 672 с.
18. Алейников А. Ф., Гридчин В. А., Цапенко М. П. Датчики (перспективные направления развития): Учеб. Пособие / Под ред. проф. М.П. Цапенко. – Новосибирск: Изд-во НГТУ, 2001. – 176 с.
19. Jacob Fraden. Handbook of Modern Sensors: Physics, Designs, and Applications. 765 P. · 2015 · 27.15 MB
20. Сергеев А. Н. Основы локальных компьютерных сетей: Учебное пособие. — СПб.: Издательство «Лань», 2016. — 184 с.
21. Гольдштейн Б.С., Кучерявый А.Е. Сети связи пост-NGN. СПб.: БХВ-Петербург. — 2015. — 160 с.
22. Donald Noris. The Internet of Things: Do-It-Yourself Projects with Arduino, Raspberry Pi, and BeagleBone Black. 2015 by McGraw-Hill Education. – 580 p. ISBN: 978-0-07-183521-3.
23. Peter Waher. Learning Internet of Things. 2015 Packt Publishing 240 p.
24. Adrian McEwen, Hakim Cassimally. Designing the Internet of Things. Wiley 2014. - 338 p. ISBN 978-1-118-43062-0
25. Москаленко Т. А., Киричек Р. В., Кучерявый А. Е. Обзор протоколов интернета вещей. СПб.: Информационные технологии и телекоммуникации. 2017. Т. 5. № 2. с.1-12.
26. Кучерявый А. Е., Кучерявый Е. А., Прокопьев А. В. Самоорганизующиеся сети. СПб.: Любавич. 2011. 312 с.
27. Кононюк А. Е. Фундаментальная теория облачных технологий. — В 18-и книгах. Кн.1. —К. : Освіта України. 2018.—620 с.
28. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.
29. Соколов М.Н., Смолянинова К.А., Якушина Н.А. Проблемы безопасности интернета вещей: обзор. — Вопросы кибербезопасности : журнал. — 2015. — № 5(13). — 34с.

30. Бурячок, В. Л. Інформаційна та кібербезпека: соціотехнічний аспект: підручник / [В. Л. Бурячок, В. Б. Толубко, В. О. Хорошко, С. В. Толюпа]; за заг. ред. д-ра техн. наук, професора В. Б. Толубка.— К.: ДУТ, 2015.— 288 с.
31. Правила проведения оценки рисков. Национальный институт стандартов и технологий США (NIST). Набор инструкций 2012. http://www.nist.gov/customcf/get_pdf.
32. Международная организация по стандартизации (ISO) и Международная электротехническая комиссия (МЭК). Стандарт ISO/IEC 31010:2009 «Управление риском. Методы оценки риска». http://www.iso.org/iso/catalogue_detail?csnumber=51073
33. Брюс Шнайдера. Оценка рисков https://www.schneier.com/essays/archives/2007/01/information_security_1.html
34. Робин Уилтон. CREDS 2014 –Документ о позиции: четыре этические проблемы доверия Интернету. Краткий отчет № CREDS-PP-2.0. Internet Society, 2014 г. https://www.internetsociety.org/sites/default/files/Ethical Data-handling - v2.0.pdf
35. Frahim, J., et al., “Securing the Internet of Things: A Proposed Framework,” Cisco White Paper, March 2015.
36. Модель NIST Special Publication 800-183 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-183.pdf