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

Інститут телекомунікаційних систем

Кафедра Інформаційно-телекомунікаційних мереж

До захисту допущено:

В.о. завідувача кафедри

_____ Лариса ГЛОБА

«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційно-комунікаційні технології»
спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Метод кодування данних в мережі Інтернету Речей за допомогою нейромережі»

Виконав:
студент IV курсу, групи ТІ-71
Ушаков Сергій Михайлович _____

Керівник:
асистент кафедри ІТМ ІТС,
Курдеча Василь Ваисльович _____

Рецензент:
Зав. кафедри промислової електроніки проф., д.т.н.
Ямненко Юлія Сергіївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

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

Інститут телекомунікаційних систем

Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 172 «Телекомунікації та радіотехніка»
Освітньо-професійна програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Лариса ГЛОБА
«___»_____2021 р.

ЗАВДАННЯ

на дипломну роботу студенту

Ушаков Сергій Михайлович

1. Тема роботи «**Метод кодування данних в мережі Інтернету Речей за допомогою нейромережі**», керівник роботи керівник роботи асистент кафедри інформаційно-телекомунікаційних мереж ІТС Курдеча Василь Васильович, затверджені наказом по університету від «14» квітня 2021 р. № 1007-с

2. Термін подання студентом роботи 7 червня 2021 р.

3. Вихідні дані до роботи наукові статті про технології Інтернету Речей.

4. Зміст роботи

1. Провести аналіз проблем кодування даних мережі Інтернету речей.
2. Проаналізувати існуючі методи удосконалення процесу кодування даних в мережі Інтернету речей та вибрати прототип.
3. Вдосконалити метод за рахунок впровадження процесу перенавчання.
4. Провести емпіричне моделювання та аналітичну оцінку запропонованого рішення

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Титульний слайд
2. Актуальність
3. Мета, основні задачі
4. Проблеми передачі даних в мережах інтернету речей
5. Аналіз сучасних методів прискорення передачі даних
6. Таблиця порівняння методів передачі даних в мережі
7. Вибір методу за прототип
8. Емпіричне моделювання
9. Результати моделювання
10. Результат оцінки
11. Загальні висновки
12. Публікації на тему

6. Дата видачі завдання 28 жовтня 2020 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблем передачі даних в мережах інтернету речей	05.10.2020 – 15. 11. 2020	Виконано
2	Аналіз сучасних методів прискорення передачі даних	15.11.2020 – 27.12.2020	Виконано
3	Порівняння методів передачі даних в мережі	27.12.2020 - 02.02.2021	Виконано
4	Проведення моделювання запропонованого методу	02.02.2021 – 05.03.2021	Виконано
5	Написання наукової статті (ПРІТС-2021) та виступ	06.03.2021 – 14.04.2021	Виконано
6	Аналіз результатів запропонованого методу	15.04.2021 – 17.05.2021	Виконано
7	Оформлення дипломної роботи	17.05.2021 – 07.06.20201	Виконано

Студент Сергій УШАКОВ

Керівник Василь КУРДЕЧА

РЕФЕРАТ

Робота містить 74 сторінки, 15 рисунків, 3 таблиці. Було використано 24 джерела.

Актуальність: актуальність дослідження полягає в тому, що кількість пристроїв в мережі IoT постійно збільшується. Разом з цим збільшується кількість рішень на ринку IoT технологій, що в купі призводить до збільшення об’ємів передачі даних. Тим самим збільшується кількість ресурсів, що витрачається на забезпечення їх передачі.

Збільшення кількості користувачів технологій Інтернету речей призводить до стрімкого збільшення даних, що передаються мережею. Чим більше трафіка попадає до мережі, тим більша швидкість передачі даних потребується. Таким чином ускладняється процес обміну трафіків, що в свою чергу призводить до потреб витрати ресурсів на розширення пропускних здатностей мереж.

Мета роботи: оптимізувати процес передачі даних, специфічного для мережам інтернету речей за рахунок удосконалення нейромережевого автоенкодеру за допомогою використання навчання меншої нейромережі на вихідних даних основного автоенкодеру.

Запропонований метод дозволить зменшити кількість даних, що передаються для зменшення використання ресурсів пропускної здатності мережі.

Ключові слова: інтернет речей, кодування даних, нейронна мережа, автоенкодер, навчання на вихідну модель, стиснення даних

ABSTRACT

The work contains 74 pages, 15 figures, 3 tables. 24 sources were used.

Topicality: The relevance of the research lies in the fact that the number of devices in the IoT network is constantly increasing. At the same time, the number of solutions in the market of IT technologies is increasing, in combination leads to an increase in the volume of data transmission. Thus, there is an increase in the amount of resources spent to ensure their transmissions.

The increase in the number of users of IoT technologies leads to a rapid increase in the amount of data transmitted over the network. The more traffic that enters the networks, the higher the data transfer rate is required. Thus, the process of traffic exchange becomes more complicated, which in turn leads to the need to spend resources to expand the capacity of networks.

The goal of the work: to optimize the process of data transfer specific to the Internet of Things networks by improving the neural autoencoder by using the training of a smaller neural network on the output data of the main autoencoder.

The proposed method will reduce the amount of data transmitted to reduce the use of network bandwidth resources.

Keywords: Internet of Things, data coding, neural network, autoencoder, training on the original model, data compression

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IoT	*Internet of Things*
p2a	*Person-2-Application*
RFID	*Radio Frequency IDentification*
DoS	*Denial-of-service*
NFC	*Near-field communication*
WSN	*Wireless sensor network*
ITS	*Intelligent transportation system*
PSCM	*Procurement and Supply Chain Management*
RDF	*Resource Description Framework*
OWL	*Web Ontology Language*
EXI	*Efficient XML Interchange (EXI)*
OS	*Operating System*
EPC	*Engineering, procurement, and construction*
MQTT	*Message Queuing Telemetry Transport*
IpV4 IpV6	*Internet Protocol version 4/6*
HTTPS	*Hypertext Transfer Protocol Secure*
SMTP	*Simple Mail Transfer Protocol*
FTP	*File Transfer Protoco*
m2m	*Machine to machine*
QoS	*Quality of service*
CoAP	*Constrained Application Protoco*
DDS	*Data Distribution Service*
XMPP	*Extensible Messaging and Presence Protocol*
XML	*Extensible Markup Language*
AMQP	*Advanced Message Queuing Protocol*
UDP	*User Datagram Protocol*
REST	*Representational state transfer*
MQTT-SN	*MQTT-Sensor Network*

MQTT-GW	*MQTT-Gateway*
STOMP	*Streaming Text Oriented Messaging Protocol*
ONS	*Object Name Service*
SOAP	*Simple Object Access Protocol*
JSON	*JavaScript Object Notation*
AE	*AutoEncoder*
DCT	*Discrete cosine transform*
PCA	*Principal component analysis*
LEC	*lossless encoding algorithm*
LTC	*Linear timecode*
RNN	*Recurrent neural network*
LSTM	*Long short-term memory*
GRU	*Gated recurrent units*
BERT	*Bidirectional Encoder Representations from Transformers*

ВСТУП

Мережі IoT – серед нових парадигм є одним з найбільш перспективних рішень для підвищення рівня автоматизації в різних сферах людської діяльності. В даний час існують технології IoT що мають унікальну ідентифікацію і дозволяють об’єктам взаємодіяти один з одним для отримання даних на веб-сервері, для їх зберігання, збору і для спільної роботи з користувачами, що успішно впроваджується не лише у сферу послуг, а також у реальний сектор економіки, напр. видобувна та нафтова промисловість, газова промисловість. Одне з найпопулярніших додатків IoT технологій - це система “Розумний дім”, яка дозволяє спрощення взаємодії людини з системами та пристроями їхнього будинку. Також такі технології широко поширені використовується в медицині для вилучення,

зберігання та передача інформації про показники життєдіяльності людини функції. Аналіз великих даних, накопичених пристроями IoT дозволяє формувати не тільки аналітичні звіти, а й рекомендації для осіб, що приймають рішення. Більше того, аналітичні дані, зібрані системами IoT, можуть бути корисно під час експлуатації та тестування різних пристроїв.

У зв'язку з стрімким розвитком технології, обсяги трафіку, що використовуються фізичними об'єктів, підключеними до мережі, зростає з безпрецедентною швидкістю, що реалізує ідею Інтернету речей. Однак Інтернет речей з хмарної підтримкою стикається з низкою труднощів у передачі даних, таких як затримки передачі, обмеження пропускної здатності, недостатній рівень захищеності і високе енергоспоживання. Таким чином, можна сказати, що необхідно вдосконалення технологій передачі даних з урахуванням особливостей парадигми IoT.

Мета роботи: підвищення ефективності передачі інформаційних потоків в системах IoT за рахунок алгоритму ефективного стиснення даних використовуючи потужності навченої нейронної мережі на граничному пристрої.

Враховуючи це, основним завданнями дослідження будуть наступними:

1) Дослідити потенціальні можливості та проблеми IoT при передачі даних;

2)Проаналізувати основні формати обміну даними між хмарним середовищем та кінцевими пристроями IoT;

3) Провести аналіз існуючих програмно-апартних рішень для поліпшення передачі даних в системах IoT.

4) Проаналізувати потреби щодо швидкості передачі даних в мережі

IoT та запропонувати більш ефективний метод обміну трафіком для спеціалізованих систем.

Об’єкт дослідження: кодування даних в мережі Інтернету речей.

Предмет дослідження: процес передачі трафіку в мережі IoT .

Теоретичний результат дослідження: Проведено аналіз існуючих методів поліпшення процесу передачі пакетів даних та розглянуто їх ефективність.

Практичний результат дослідження: Запропоновано метод, що пришвидшить процес передачі даних, знизить трафіко та енерго-споживання а також збільшить захищеність даних.

Публікації:

1. Ушаков С. М. Аналіз кодування даних в мережі інтернету речей IoT // XV Міжнародна Науково-технічна Конференція "Проблеми телекомунікацій 2021"

2. Ушаков С. М. Метод обробки інформації на основі нейромереж для мережі інтернету речей // XV Міжнародна Науково-технічна Конференція студентів та аспірантів "Перспективи розвитку інформаційно-телекомунікаційних технологій та систем-2021"

РОЗДІЛ 1
ОСНОВНІ АСПЕКТИ РОБОТИ МЕРЕЖІ IoT

1.1. Базові визначення в мережі IoT

Інтернет речей (The Internet of Things, IoT) – це новий етап розвитку технології Інтернет та способу використання фізичних предметів («речі»), оздоблених вбудованими пристроями, поєднуючи їх у мережу для подальшого збору, аналізу, застосування та передачі даних отриманих від людини.

Інтернет речей (IoT) пов’язує об’єкти з Інтернетом, завдяки чому відкриваються можливості у їх використанні, а також зборі і аналізі інформації, що раніше були недоступні. Технологія IoT відкриває альтернативи не тільки у поєднанні речі, пристроїв, приладі у єдину мережу та отримання даних з її допомогою, а й просуває новітні методи їх накопичення, структуризації та обробки.

Не зважаючи на фактичний набор фізичних сенсорів та датчиків, зручно і правильно буде розглядати Інтернет речей як реалізацію тісної інтеграції світу матеріального, а також світу віртуального – інформаційного, що необхідно для подальшого спілкування людей та пристроїв (в форматі p2a) в якості методу дистанційної взаємодії з фізичним об’єктами. Сам термін «IoT» відноситься до різноманітних додатків – підключених у мережу пристроїв. Вони в свою чергу поділяються на два елементи: пристрої шо

виконують функцію керування (зберігння та використання) і пристрої які накопичують інформацію (сенсори та датчики для збору та обробки отриманих даних). Такі пристрої використовують повсемісно у розумних містах, будинках та будівлях, управлінні енергетикою та енергосистемами, охороні здоров’я, транспорті, виробництві, екологічному контролі, на реальному виробництві та підприємстває та у багатьох інших сферах.

В майбуньому Інтернету-речей прогнозують місце серед активних гравців у інформаційних, соцільних, економічних та бізнес-процесах, де вони будуть використовуватись як перевага, що має змогу обмінюватись між собою збираними та оброблюваними даними про стан навколишнього середовища, отримуючи необхідну інформацію без зайвого втручання людини.

Для реалізації ідеї IoT використовується декілька технологій і датчиків. Комунікаційні технології, які використовуються для реалізації ідеї IoT, - це радіочастотна ідентифікація (RFID), комунікація ближнього поля (NFC), бездротові сенсорні мережі (WSN) та ін.

Існує безліч додатків, в яких застосовується IoT. Вони стали розумними і виконують свою роботу роботизовано, за допомогою Інтернету. Перша з них – галузь охорони здоров’я, де датчики використовуються для перевірки температури тіла людини, кров’яного тиску і частоти серцевих скорочень. Інша область застосування – розумний будинок, оскільки люди використовують вдома безліч електронних пристроїв, таких як холодильники,

мікрохвильові печі, вентилятори, обігрівачі та кондиціонери. Датчики встановлюються для того, щоб виявити проблему і повідомити про неї компанії-виробнику для її вирішення. Третя застосування IoT задля відстеження тварин. Датчики GPS встановлені на тілі тварини надають можливість легко відстежити її. Вони також використовуються для моніторингу та годування домашніх улюбленців та худоби. Ще одне застосування IoT – інтелектуальні захвати з використанням робототехніки, які безпосередньо контактують з об’єктом для збору сенсорної інформації. В пристроях інтелектуальних захватів встановлено безліч датчиків і приладів, таких як датчики дотику, руху, зору, оптичні та силові датчики. Рівень інтелектуальності таких приладів залежить від оснащеності сенсорами, оскільки вони збирають інформацію в режимі реального часу, а зібрана інформація використовується для прийняття рішень. Саме тому їм потрібно бути обмежені такими критеріями проектування, як вартість, вага і компактність. Крім того, існує безліч областей застосування IoT, таких як інтелектуальний транспорт, управління інфраструктурою (автомагістралі, мости та залізничні шляхи), виробництво, інтелектуальне будівництво, інтелектуальне сільське господарство, інтелектуальна роздрібна торгівля тощо. У таблиці 1 показані сфери застосування IoT. Крім того, в ній також порівнюються різні сфери застосування з точки зору кількості користувачів, технології зв’язку, розміру мережі, пропускної спроможності і їх випробувальних середовищ.

Таблиця 1.1.

Порівняння різних областей застосування IoT.[1, 2, 3, 4]

	Дім/Офіс	Місто	Транс-порт	Сільське госпо-дарство	Роздріб на торгівля
Число користувачів	*Дуже мало*	*Багато*	Багато	*Мало*	*Мало*
Комунікації	*RFID та WSN*	*RFID та WS*	*WSN*	*WSN*	*RFID та WSN*
Мережа	*Маленька*	*Середня*	*Середня*	*Середня*	*Маленька*
Тип з'єднання	$Wi-Fi$, $3G, 4G$	$Wi-Fi$, $3G, 4G$	$Wi-Fi$, *Супутник*	$Wi-Fi$, *Супутник*	$Wi-Fi, 3G, 4G$
Пропусказдатність	*Низька*	*Велика*	*Середня*	*Середня*	*Низька*
Тестові середовища	*Розумні будинки*	*Розумні міста*	*ITS*	*PSCM*	*Торгові центри*

Кількість пристроїв IoT зростає з кожним днем. Причина збільшення кількості IoT-пристроїв полягає в тому, що вони забезпечують комфорт в житті людини і виконують роботу з кращими результатами, ніж люди. Повідомляється, що в 2020 році кількість IoT-пристроїв збільшилась більш ніж в три рази з 2012 року і налічує навколо 50 мільярдів пристроїв, які будуть працюють в Інтернеті. На рисунку 1.1 показано кількість підключених IoT-пристроїв з 2012 по 2020 рік згідно businessresearcher.

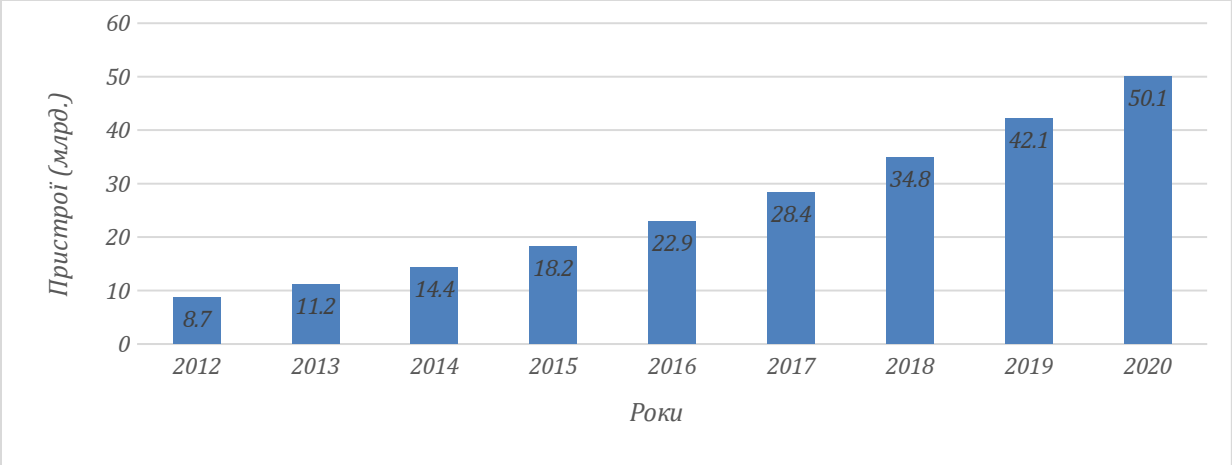


Рис. 1.1. Кількість підключених IoT-пристроїв з 2012 по 2020 рік

Люди використовують нову технологію під назвою «Інтернет речей» (IoT) не тільки через популярність, але і завдяки перевагам і послугам, які вона надає. За допомогою IoT можна виконувати завдання без участі людини, що робить життя простіше і легше. Це дозволяє людям автоматизувати, досягати і контролювати завдання, які необхідні для життя, і в результаті отримувати більш якісні рішення. IoT надає безліч переваг, але, з іншого боку, у нього є і деякі проблеми, такі як низька пропуска спроможність, неефективне обслуговування, енергоефективність, управління ідентифікацією, безпека і конфіденційність. У IoT всі пристрої підключені до Інтернету, оскільки без Інтернету вони не можуть виконувати свої завдання. В Інтернеті існує безліч атак, які викрадають конфіденційну інформацію об’єктів. Зловмисники можуть використовувати інформацію користувачів будь-яким незаконним способом у відповідності зі своїми потребами, що може привести до великих втрат для користувачів. Конфіденційність також стала проблемою для IoT. Безпека і конфіденційність повинні бути забезпечені шляхом запобігання несанкціонованій ідентифікації, доступу до даних користувача, які знаходяться під його контролем і нічиїм іншим.

1.2. Елементи IoT

IoT надає користувачам безліч переваг і можливостей. Таким чином, для того щоб використовувати їх так яка задумано, необхідні деякі елементи. У цьому підрозділі приведена класифікація елементів IoT. На малюнку 3 показані елементи, необхідні для забезпечення функціональності IoT. Назви і подробиці цих елементів наведені нижче.

Рисунок 1.2. Елементи IoT

- Ідентифікація

Ідентифікація забезпечує явну ідентифікацію кожного об’єкта в мережі. Існує два процеси ідентифікації: іменування та адресація. Іменування – це найменування конкретного об’єкта, а адресація – його унікальна адреса. Ці два терміни сильно відрізняються один від одного, оскільки два або більше об’єктів можуть мати однакові імена, але завжди різні і унікальні адреси. Існує безліч методів, що забезпечують можливість

присвоєння імен об’єктам в мережі, наприклад, електронні коди продуктів (EPC) і повсемісні коди. Для присвоєння унікальної адреси кожному об’єкту використовується протокол Ipv6. Спочатку для присвоєння адреси використовувався протокол Ipv4, але він не зміг задовольнити потребу в адресації через великої кількості пристроїв IoT. Зараз використовується саме Ipv6, оскільки він використовує схему адресації 128-бітних чисел.

- Сприйняття (сенсорний)

Процес збору інформації з об’єктів відомий як зчитування. Зібрана інформація відправляється на носій. Існує безліч сенсорних пристроїв для збору інформації з об’єктів, таких як виконавчі механізми, RFID-мітки, інтелектуальні датчики, що носяться, сенсорні пристрої і т.д.

- Комунікація

Зв’язок є однією з основних цілей IoT, в якій різні пристрої підключаються один до одного і обмінюються інформацією. У процесі комунікації пристрої можуть відправляти і отримувати повідомлення, знаки та іншу інформацію. Існує безліч технологій, що забезпечують зв’язок, таких як радіочастотна ідентифікація (RFID), зв’язок ближнього поля (NFC), Bluetooth, Wi-Fi і Long Term Evolution (LTE).

- Обчислення

Обчислення виконуються на основі інформації, зібраної з об’єктів за допомогою датчиків. Вони використовуються для видалення зайвої інформації, яка не є необхідною. Для виконання обчислень в додатках IoT розроблено безліч апаратних і програмних платформ. Для апаратних платформ використовуються Audrino, Rasperry Pi і Intel Galileo, а для програмних платформ важливу роль грає операційна система. Існує безліч

типів використовуваних операційних систем, таких як Tiny OS, Lite OS, Android тощо.

- Сервіси (рівень послуг)

На даний час, основними є чотири основні типи послуг, які надаються додатками IoT. Перший – це сервіс, пов'язаний з ідентифікацією. Він використовується для отримання ідентифікаційних даних об'єктів, що відправили запит. Агрегація інформації – це сервіс, метою якого є збір всієї інформації від об'єктів. Обробка також виконується службою агрегації. Третя служба – це служба спільної роботи, яка приймає рішення відповідно до зібраною інформацією і відправляє необхідні відповіді пристрою. Останній сервіс – це повсюні сервіс, який використовується для моментального реагування пристроїв без прив'язки до часу і місця.

- Семантика

IoT відповідає за те, щоб полегшити користувачам виконання їх завдань. Це найважливіший елемент IoT для виконання своїх обов'язків. Він діє як мозок IoT – отримує всю інформацію і приймає відповідні рішення для відправки відповідей пристроїв.

У таблиці 1.2 наведено опис ключових технологій, задіяних у кожному елементі IoT.

Таблиця 1.2.

Елементи і ключові технології IoT

Елементи IoT	Технології
Ідентифікація *Іменування адресації*	Електронника, Код товару, Ucode Ipv4 та Ipv6
Сприйняття	Розумні датчики, RFID-мітки, Сенсорні пристрої що носяться

		та приводи
Сприйняття		Ідентифікація радіочастот, бездротова сенсорна мережа, зв’язок ближнього поля (NFC), Bluetooth, тривала еволюція (LTE)
Обчислення	Апаратне та програмне забезпечення	Arduino, Raspberry Pi, Intel Galil Операційна система
Сервіси		Пов’язані з ідентичністю, Агрегація інформації, Інформація про співпрацю та повсюдні
Семантика		RDF, OWL, EXI

1.3. Багаторівневі архітектури IoT та їх вразливості

Не існує єдиного і спільної думки про архітектуру IoT, яке було б узгоджено з усім світом і розробниками. Безліч різних архітектур було запропоновані різними дослідниками. На думку деяких фахівців, архітектура IoT складається з трьох рівнів, але деякі розробники підтримують чотирьох-рівневу. Вони вважають, що в зв’язку з розвитком IoT тришарова архітектура не може задовольнити вимоги додатків. Через проблеми безпеки і конфіденційності в IoT також була запропонована архітектура з п’яти рівнів. Вважається, що нещодавно запропонована архітектура може задовольнити вимоги IoT щодо безпеки та конфіденційності.

1.3.1. Трирівнева архітектура

Це дуже базова архітектура, яка відповідає основній ідеї IoT. Вона була запропонована на ранніх стадіях розвитку IoT [5-7]. Складається вона з трьох рівнів. Ці три рівня називаються: рівень сприйняття, мережевий рівень і рівень додатків, як показано на рисунку 4.

Рис. 1.3. Трирівнева архітектура IoT.

1.3.1.1. Рівень сприйняття

Він також відомий як сенсорний рівень. Він працює подібно очам, вухам і носу людини. Відповідає ж він за ідентифікацію об’єктів і збір інформації з них. Існує безліч типів датчиків, що прикріплюються до об’єктів для збору інформації, таких як RFID, двомірні штрих-коди і сенсори. Датчики вибираються відповідно до вимог додатків. Інформація,

яку збирають цими датчиками, може стосуватися місця дислокації, змін в повітрі, навколишньому середовищу, русі, вібрації тощо. Однак вони є основною метою зловмисників, які хочуть використовувати їх для заміни датчиків своїми власними. Тому більшість загроз пов’язано саме з датчиками. Загальна загрозами безпеці рівня сприйняття є:

- Підслуховування:

Підслуховування – це несанкціонована атака в режимі реального часу, при якій зловмисник перехоплює приватні комунікації, такі як телефонні дзвінки, текстові повідомлення, факси або відеоконференції. Він намагається вкрасти інформацію, передану по мережі. Зловмисник використовує переваги незахищеною передачі для доступу до відправляємої і одержуваної інформації.

- Захвати вузла:

Це одна з небезпечних атак, з якою стикаються на рівні сприйняття IoT. Зловмисник отримує повний контроль над ключовим вузлом, наприклад, шлюзовим вузлом. Він може викачати всю інформацію, включаючи зв’язок між відправником і отримувачем, ключ, який використовується для шифрування та забезпечення безпечного зв’язку, і інформацію, що зберігається в сховищі пам’яті.

- Фальшивий і шкідливі вузли:

Це типи атак, при якій зловмисник додає вузол в систему і вводить підроблені дані. Його мета – припинити передачу реальної інформації. Вузол, доданий зловмисником, витрачає дорогоцінну енергію справжніх вузлів і потенційно може керувати ними, щоб зруйнувати мережу.

- Атака відтворення:

Це атака, при якій зловмисник підслуховує збережені дані між відправником і отримувачем і забирає справжню інформацію у відправника. Зловмисник відправляє жертві ту ж аутентифіковану інформацію, яка вже була отримана в його повідомленні, пред'являючи докази своєї особистості і справжності. Повідомлення знаходиться в зашифрованому вигляді, тому одержувач може розцінити його як коректний запит і вжити заходів, що входять в інтереси зловмисника.

- Атака по таймінгу:

Зазвичай використовується в пристроях, які мають слабкі обчислювальні можливості. Вона дозволяє атакуючому виявити уразливості і витягти секрети, що зберігаються в системі безпеки, спостерігаючи за тим, скільки часу потрібно системі, щоб відповісти на різні запити, вхідні дані або криптографічні алгоритми.

1.3.1.2. Мережевий рівень

Мережевий рівень, також відомий як рівень передачі даних, діє як міст між рівнем сприйняття і прикладним рівнем. Він переносить і передає інформацію, зібрану з фізичних об'єктів за допомогою датчиків. Середовище передачі може бути бездротовим або з'єднане дротом. Цей рівень також відповідає за з'єднання «розумних» речей, мережевих пристроїв і мереж один з одним. Тому він дуже чутлива до атак з боку зловмисників та має серйозні проблеми безпеки, що стосуються цілісності і аутентифікації інформації, яка передається по мережі. Спільними загрозами і проблемами безпеки для мережевих рівнів є:

- Атаки «відмова в обслуговуванні» (Denial of Service):

DoS-атака – це атака, спрямована на запобігання доступу справжніх користувачів до пристроїв або інших ресурсів мережі. Зазвичай вона здійснюється шляхом завалювання цільових пристроїв або мережевих ресурсів надлишковими запитами, щоб зробити неможливим або складним їх використання деякими або всіма автентичними користувачами.

- Атаки «людина посередині» (Man-in-The-Middle):

MiTM атака – це атака, при якій зловмисник таємно перехоплює і змінює зв’язок між відправником і отримувачем, які вважають, що вони безпосередньо спілкуються один з одним. Оскільки зловмисник контролює комунікацію, він або вона може змінювати повідомлення у відповідності зі своїми потребами. Це становить серйозну загрозу для безпеки в Інтернеті, оскільки дає зловмиснику можливість перехоплювати і маніпулювати інформацією в режимі реального часу.

- Атаки на сховище:

Інформація користувачів зберігається на пристроях-сховищах або в хмарі. І пристрої зберігання і хмара можуть бути атаковані зловмисником, і інформація користувача може бути змінена на невірні дані. Тиражування інформації, пов’язане з доступом до іншої інформації різними типами людей дає більше шансів для атак.

- Експлойт-атака:

Експлойт – це будь-яка аморальна або незаконна атака у вигляді програмного забезпечення, фрагментів даних або послідовності команд. Вона використовує уразливості безпеки в додатку, системі або

обладнанні. Зазвичай вона проводиться з метою отримання контролю над системою і крадіжки інформації, що зберігається в мережі.

1.3.1.3. Прикладний рівень

Прикладний рівень визначає всі програми, які використовують технологію IoT або в яких розгорнуто IoT. Додатками IoT можуть бути розумні будинки, розумні міста, розумна охорона здоров'я, відстеження тварин тощо. На ньому лежить відповідальність за надання послуг додаткам. Послуги можуть бути різними для кожної програми, оскільки послуги залежать від інформації, зібраної датчиками. На прикладному рівні існує безліч проблем, в яких безпека є ключовим питанням. Зокрема, коли IoT використовується для створення розумного будинку, це тягне за собою безліч загроз і вразливостей зсередини і зовні. Для забезпечення надійної безпеки в розумному будинку на базі IoT однією з основних проблем є те, що пристрої, що використовуються в розумних будинках, мають слабку обчислювальну потужність і невеликий обсяг пам'яті, наприклад ZigBee. Загальними загрозами безпеці та проблемами прикладного рівня є:

- Міжсайтовий скриптинг:

Це ін'єкційна атака. Вона дозволяє зловмисникові вставити сценарій на стороні клієнта, наприклад, java-скрипт, на довірений сайт, який переглядають інші користувачі. Таким чином, зловмисник може повністю змінити вміст програми відповідно до своїх потреб і використовувати вихідну інформацію в незаконний спосіб.

- Атаки шкідливим кодом:

Це код в будь-якій частині програмного забезпечення, призначений для того, щоб викликати небажані ефекти і нанести шкоду системі. Цей тип загрози не може бути заблокований або контролюватися за допомогою антивірусних інструментів. Він може або активуватися сам, або бути схожим на програму, що посилає запит користувачеві для виконання якої-небудь дії.

- Здатність працювати з масовими даними:

У зв'язку з великою кількістю пристроїв і масовим обсягом передачі даних між користувачами трапляється так, що прикладний рівень не здатний обробляти дані відповідно до вимог. В результаті це призводить до порушення роботи мережі і втрати даних.

1.3.2. Чотирьохрівнева архітектура

Трирівнева архітектура була самою базовою, через постійний розвиток IoT вона не могла задовольнити всі його вимоги. Тому розробники запропонували архітектуру з чотирма рівнями. Вона має три рівні, як і попередня архітектура, але також має і один додатковий рівень, званий допоміжним. На малюнку 1.3 представлена багаторівнева архітектура, а також рекомендовані механізми безпеки, які використовуються для захисту від зловмисників. Три шару мають ту ж функціональність, що і тришарова архітектура, що було обговорено вище. Функціональність допоміжного шару і атак безпеки виглядає наступним чином:

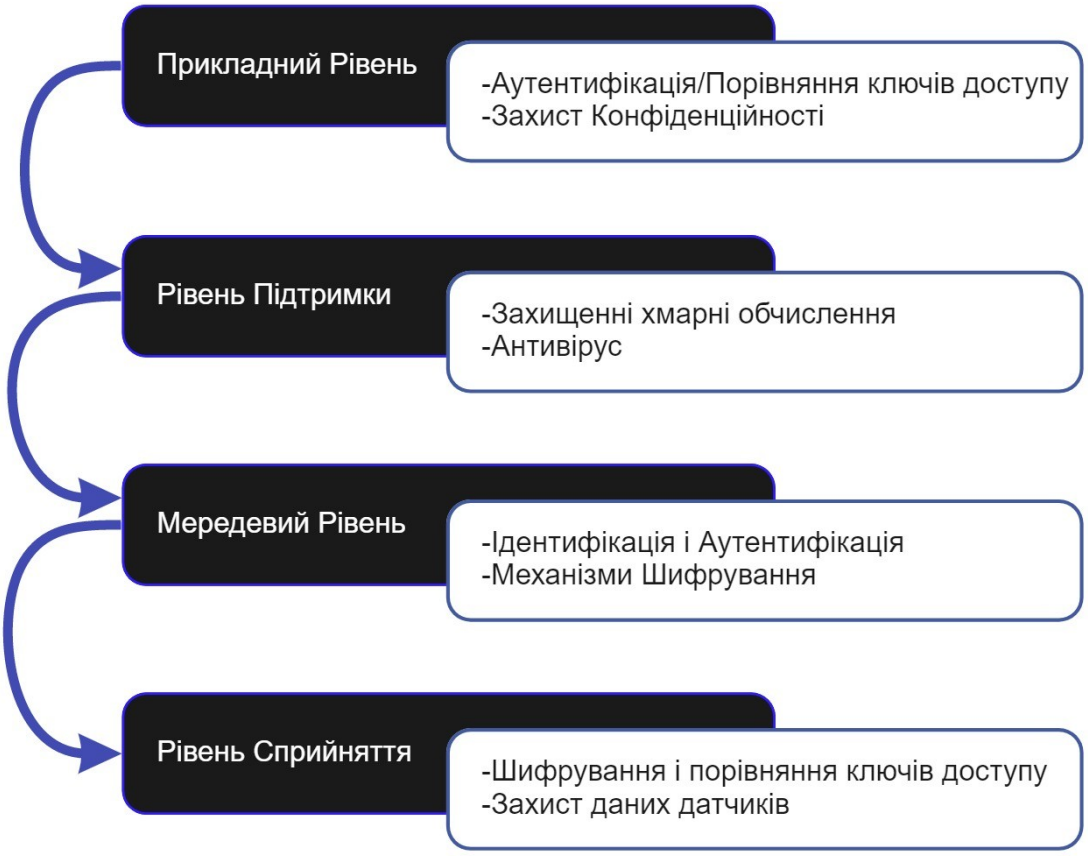


Рис. 1.3. Чотирьохрівнева архітектура IoT і рекомендовані механізми безпеки для неї.

1.3.3. Рівень підтримки

Причиною створення четвертого рівня є безпека в архітектурі IoT. У тришарової архітектурі інформація відправляється безпосередньо на мережевий рівень. Через те, що інформація відправляється безпосередньо на мережевий рівень, ймовірність виникнення загроз зростає. У зв'язку з недоліками, що були в трирівневій архітектурі, був запропонований новий рівень. У чотирирьохрівневій архітектурі інформація відправляється на допоміжний рівень, одержуваний від рівня сприйняття. У нього є два обов'язки: він стежить за тим, щоб інформація відправлялася справжніми

користувачами і була захищена від загроз. Існує безліч способів перевірки користувачів і інформації. Найбільш часто використовуваним методом є аутентифікація. Вона здійснюється за допомогою попередньо розділених секретів, ключів і паролів. Другий обов'язок допоміжного рівня – відправка інформації на мережевий рівень. Засіб передачі інформації з рівня підтримки на мережевий рівень може бути бездротовим й дротовим. Існують різні атаки, які можуть вплинути на цей рівень, такі як атака DoS, зловмисний інсайдер, несанкціонований доступ тощо. Загальними загрозами та проблемами рівня підтримки є:

- DoS-атака:

DoS-атака на рівні підтримки пов'язана з мережевим рівнем. Зловмисник відправляє велику кількість даних, що призводить до переповнення мережі. Таким чином, масове споживання системних ресурсів виснажує IoT і робить користувача нездатним отримати доступ до системи.

- Зловмисна атака інсайдерів:

Вона відбувається зсередини IoT з метою отримання доступу до особистої інформації користувачів. Здійснити її можуть авторизовані користувачі для доступу до інформації іншого користувача. Це дуже різноманітна і складна атака, яка потребує різних механізмів для запобігання загрози.

1.3.4. П’ятирівнева архітектура

Чотирьохрівнева архітектура зіграла важливу роль у розвитку IoT, але у ній також були проблеми з безпекою і зберіганням даних. Розробниики запропонували п’ятирівнева архітектуру для безпеки IoT [8-10]. Вона має три рівні, що вже були у попередніх архітектурах: рівень сприйняття, транспортний рівень і рівень додатків та ще два рівня крім них. Назви цих нових запропонованих рівнів – рівень обробки і бізнес-шар. Вважається, що запропонована архітектура здатна задовольнити всі вимоги IoT. Вона також здатна зробити застосування IoT безпечним. Робота цих рівнів і атаки безпеки, які можуть впливати на них, описані нижче:

1.3.4.1. Рівень обробки

Рівень обробки також відомий як рівень проміжного програмного забезпечення. Він збирає інформацію, відправлену з транспортного рівня. Він виконує обробку зібраної інформації. В його обов'язки входить видалення зайвої інформації, що не має сенсу, і витяг корисної інформації. Однак це також усуває проблему великих даних в IoT. У великих даних надходить велика кількість інформації, що може вплинути на продуктивність IoT. Існує безліч атак, які можуть вплинути на рівень обробки і порушити продуктивність IoT. До поширених атак відносяться:

- Вичерпання:

Зловмисник використовує вичерпання для порушення обробки структури IoT. Це відбувається як наступний ефект атак, таких як DoS-

атака, при якій зловмисник посилає жертві безліч запитів, роблячи мережу недоступною для користувачів. Це може бути результатом інших атак, спрямованих на вичерпання ресурсів системи, таких як ресурси батареї і пам'яті. IoT має розподілену природу, тому в ній немає великої кількості небезпек. Набагато простіше реалізувати процедури захисту від них [11].

- Зловмисні програми:

Це атака на конфіденційність інформації користувачів. Вона являє собою застосування вірусів, шпигунських програм, рекламного ПЗ, троянських програм для взаємодії з системою. Ця атака може приймати форму виконуваних кодів, сценаріїв і вмісту та діє проти вимог системи, щоб вкрасти конфіденційну інформацію [12].

1.3.4.2. Бізнес-рівень

Бізнес-рівень є провісником поведінки додатка і діє як менеджер всієї системи. Він відповідає за управління і контроль додатків, моделі і прибутку. Цей рівень також управляє конфіденційністю користувача. Він має можливість визначати, як інформація може бути створена, збережена і змінена. Уразливість в цьому рівні дозволяє зловмисникам використовувати додаток в обхід бізнес-логіки. Більшість проблем, пов'язаних з безпекою, є слабкими місцями в додатку, які виникають в результаті порушення або відсутності контролю безпеки. Поширеними проблемами, пов'язаними з безпекою бізнес-рівня, є:

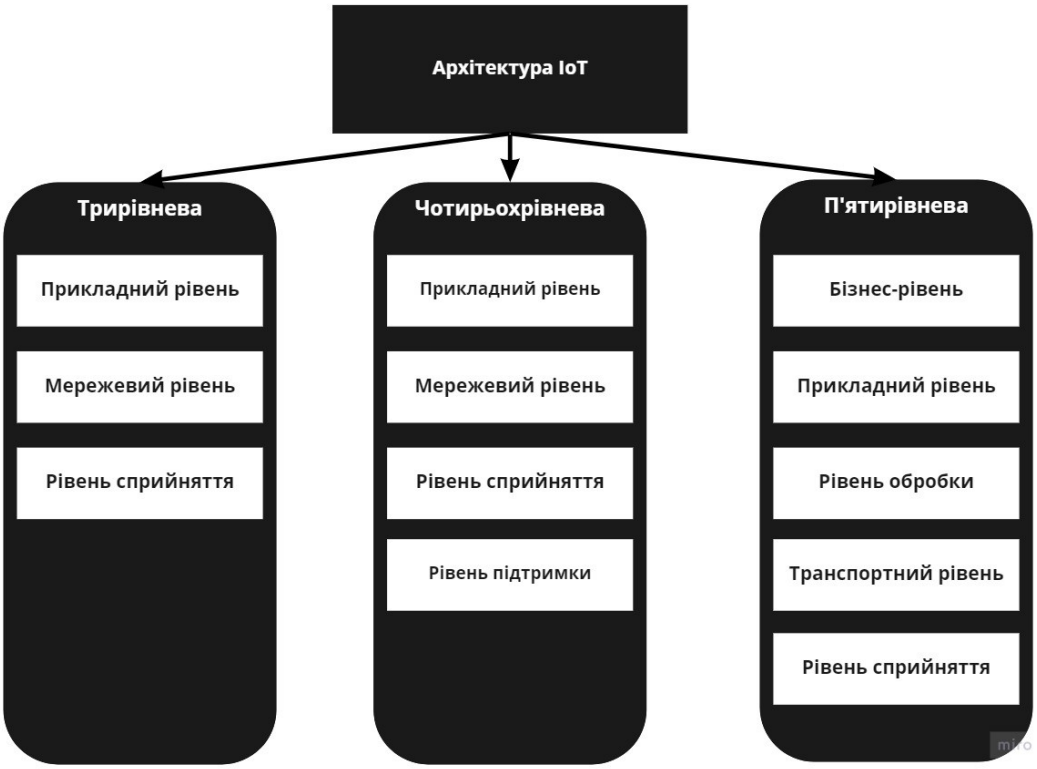
- Атака на бізнес-логіку:

Вона використовує переваги недоліків в програмуванні та контролює і управляє обміном інформацією між користувачем і базою даних програми. У бізнес-рівня існує кілька загальних недоліків, таких як невірно написаний код, перевірка та відновлення пароля, помилки під час введення і методи шифрування.

- Атака нульового дня:

Вона відноситься до прогалин в безпеці або проблеми в додатку, яка незнайома постачальнику. Ці прогалини в безпеці використовується зловмисником для отримання контролю без згоди користувача і без його відома [13,14].

Ієрархія всіх запропонованих багаторівневих архітектур Інтернету речей (IoT) показана на рисунку 1.4, де показані багаторівневі архітектури IoT, що складаються з трьох, чотирьох і п’яти рівнів відповідно.



Рису 1.4 Багаторівневі архітектури IoT (трьох-, чотирьох- і п'ятирівнева)

Висновки:

В першому розділі було розв’язано такі питання:

- Розглянуто основні терміни мережі Інтернет речей;
-Наведено характеристики всіх основних елементів;
- Досліджено архітектуру та основні компоненти IoT.

Після аналізу моделі IoT можна зробити висновок, що попри те, що переваг більше ніж недоліків, але це не означає, що недоліків мало. У системі є проблеми з різними видами атак на неї (більшість з яких блокуються за допомогою стандартних заходів безпеки), а також важливим остаються питання щодо енергоспоживання та високих об’ємів трафіку, що передаються мережею.

РОЗДІЛ 2
АНАЛІЗ МЕТОДІВ СТИСНЕННЯ ДАННИХ В ІОТ

2.1. Протоколи для передачі даних в IoT (протоколи прикладного рівню)

Прикладний рівень відноситься до рівнів OSI 5, 6 і 7 в моделі TCP-IP. В архітектурі ІОТ цей рівень знаходиться над рівнем відкриття послуг та є самим верхнім рівень , що тягнеться від клієнтської частини. Це інтерфейс між кінцевими пристроями і мережею, що реалізується через спеціальний додаток на стороні пристрою. Браузер реалізує протоколи прикладного рівня, такі як HTTP, HTTPS, SMTP і FTP. Точно так само існують протоколи прикладного рівня в контексті ІОТ.

Прикладний рівень відповідає за форматування і представлення даних. В Інтернеті він зазвичай заснований на протоколі HTTP, однак HTTP не підходить для середовища з обмеженими ресурсами, оскільки він дуже громіздкий і, отже, вимагає великих затрат по ресурсам. Тому існує безліч альтернативних протоколів, які були розроблені для середовищ ІОТ. Деякі з них наведені далі: MQTT; SMQTT; CoAP; DDS; XMPP; AMQP; RESTful HTTP; MQTT-SN; STOMP; SMCP; LLAP; SSI; LWM2M; M3DA; XMPP-IOT; ONS 2.0; SOAP; Websocket; Reactive Streams; HTTP/2; JavaScript IOT

2.1.1. MQTT

Message Queuing Telemetry Transport - це легкий протокол обміну повідомленнями. Він використовує спосіб зв'язку видавець – абонент і

тому застосовується для зв'язку M2M (machine to machine). MQTT заснований на протоколі TCP-IP і призначений для роботи в умовах обмеженої пропускної здатності. У термінології протоколу обмежена пропускна здатність мережі називається "малим відбитком коду". Однак точне значення обмежену пропускну здатність мережі неясно в специфікації.

Цей протокол був спеціально розроблений для сенсорних мереж і бездротових сенсорних мереж. Він дозволяє пристроям відправляти або публікувати інформацію про дані по заданій темі на сервер. Між видавцем і абонентом знаходиться MQTT брокер (Broker-Mosquitto). Брокер передає інформацію клієнтам, які попередньо підписалися на розсилку.

Датчики взаємодіють з брокером, який представляє собою IOT-пристрій або сервер, що зчитує і публікує дані датчиків. Інші пристрої, які підписуються на дані датчиків і запитують їх, називаються клієнтами. Самі датчики називаються видавцями в мережі. Клієнтом може бути ноутбук, смартфон, планшет або інший мобільний пристрій. Клієнтські пристрої повинні підписатися на брокера в мережі, щоб отримувати дані датчиків. Для отримання даних підписані клієнтські пристрої повинні встановити з'єднання з брокером і зробити запит. Брокер отримує дані від видавця (бездротових датчиків) і відправляє їх запитувачу клієнту. Навіть якщо зв'язок з клієнтським пристроєм переривається після виконання запиту, брокер зберігає дані в кеші, щоб при повторному з'єднанні з брокером клієнтський пристрій міг отримати запитані дані датчиків. Аналогічно, якщо з'єднання між видавецем і брокером розривається після виконання запиту, брокер пересилає відповідні інструкції, відправлені видавцем, щоб клієнтський пристрій міг знову підключитися і отримати запитані дані.

Таким чином, MQTT добре працює навіть тоді, коли зв'язок між брокером і видавцем або брокером і клієнтом переривається через

обмежену пропускну здатність мережі. Така здатність справлятися із затримками або затримками в мережі і робить цей протокол цілком відповідним для бездротових мереж.

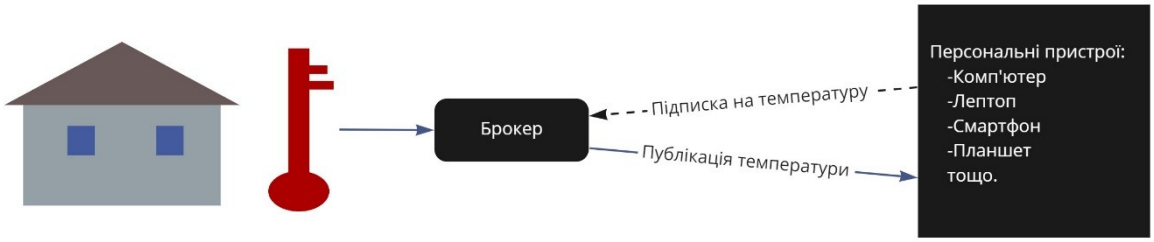


Рисунок 2.1 Архітектура MQTT

MQTT сесію можна розділити на чотири етапи:

- З'єднання
- Аутентифікація
- Комунікація
- Завершення

З'єднання і аутентифікація - На цьому етапі клієнт (наприклад, мобільний пристрій або ноутбук) ініціює TCP-IP з'єднання з брокером (сервером), використовуючи стандартний порт або порт, визначений оператором мережі (стандартними портами зазвичай є 1883 для незашифрованого зв'язку і 8883 для зашифрованої зв'язку через SSL або TLS). При шифрованому зв'язку сервер відправляє свій сертифікат для аутентифікації себе клієнтом, клієнт також може відправити сертифікат сервера для аутентифікації себе, хоча навіть при зашифрованому зв'язку аутентифікація клієнта не є частиною специфікації і не є широко поширеною. Проте, зазвичай аутентифікація клієнта здійснюється шляхом передачі клієнтом брокера (сервера) імені користувача та пароля.

Комунікація - сенсорні дані передаються клієнту за допомогою невеликих частин коду. “Мала частина коду” зазвичай містить заголовок довжиною 2 байта, необов'язковий заголовок змінної довжини, дані датчика в якості корисного навантаження повідомлення розміром не більше 250 Мб і індикатор рівня якості обслуговування. Може бути три рівня якості обслуговування (QoS) - невизнана обслуговування (QoS Level 0), підтверджене обслуговування (QoS Level 1) і гарантоване обслуговування (QoS Level 2). Кожне підвищення рівня QoS вимагає більшої пропускної здатності мережі та стійкості до затримок.

Існує чотири типи операцій, які може виконувати клієнт - публікація, підписка, відписка і пінг. Клієнт може підписатися на певні теми в мережі, наприклад, мобільний пристрій може підписатися на брокера, щоб дізнатися поточну температуру в місті. Для підписки клієнт повинен відправити брокеру пару пакетів SUBSCRIBE/SUBACK. Аналогічно, для відписки (від теми) клієнту необхідно відправити пару пакетів UBSUBSCRIBE/UNSUBACK на сервер. Операція ping може бути виконана клієнтським пристроєм для перевірки поточного з'єднання з брокером. В операції публікації дані передаються між брокером і клієнтом по певній темі.

Завершення – видавець або клієнт може перервати з'єднання, відправивши брокеру повідомлення DISCONNECT, після чого з'єднання TCP-IP закривається. Якщо з'єднання розривається раптово (через обмежену пропускну здатність мережі) без запиту на розрив, брокер відправляє клієнту кушоване повідомлення від видавцем.

2.1.2. SMQTT

Заснований на MQTT, протокол Secure Message Queue Telemetry Transport (SMQTT) являє собою базований на шифруванні легкий протокол обміну повідомленнями. У порівнянні з сесією MQTT, сесія SMQTT складається з чотирьох етапів - установка, шифрування, публікування і дешифрування. Він має архітектуру на основі брокера, аналогічну MQTT, проте в цьому протоколі і абонент, і видавець повинні зареєструватися у брокера, використовуючи секретний майстер-ключ. Дані також шифруються перед публікацією видавцем, а потім розшифровуються на стороні абонента. Розробником може бути використаний будь-який алгоритм шифрування.

2.1.3. CoAP

Constrained Application Protocol (протокол обмежених додатків) спеціально розроблений для обмеженого обладнання. Устаткування, яке не підтримує HTTP або TCP/IP, може використовувати протокол CoAP. Отже, в основному розробники цього протоколу, під впливом HTTP, розробили протокол CoAP, що використовує протоколи UDP і IP. CoAP легкий протокол, який необхідний для IOT-додатків з низьким енергоспоживанням, наприклад, для зв'язку між IOT-пристроями, що працюють від батарей. Як і HTTP, він також працює по моделі клієнт-сервер. Клієнти можуть отримувати, PUT, DELETE або POST інформаційні ресурси по мережі.

CoAP використовує два типи повідомлень - запити і відповіді. Повідомлення містять базовий заголовок, за яким слідує тіло повідомлення, довжина якого визначається датаграммой. Повідомлення

або пакети даних мають невеликий розмір, що дозволяє передавати їх між пристроями з обмеженнями без втрат даних. Повідомлення можуть бути підтверджуємими або непідтверджуємими. Для підтверджуємих повідомлень клієнт повинен відповісти підтвердженням після отримання пакету даних. Повідомлення запиту можуть містити рядки запиту для реалізації додаткових функцій, таких як пошук, сортування або пейджинг.

Для безпеки протокол використовує Datagram Transport Layer Security (DTLS) через UDP. У HTTP дуже складно дізнатися новий стан змінної, клієнт повинен знову і знову проводити опитування, що означає, що клієнт повинен питати кожну секунду, чи є новий стан змінної, за якою він спостерігає. У CoAP сервіси намагаються вирішити цю проблему шляхом створення флагу спостереження, так що якщо оригінальний пристрій посилає флаг спостереження з командою GET, кожен раз, коли сервер або інші пристрої бачать, що є оновлення в стані змінної, сервер або пристрої будуть посилати push-повідомлення оригінальному пристрою, яке фактично знаходить флаг спостерігача.

Протокол також надає додатковий механізм для виявлення ресурсів клієнтськими пристроями. Сервер повинен надати список ресурсів разом з метаданими, які можуть бути доступні у вигляді посилання або додатку медіа-типу. Переміщаючись по списку, клієнт може знайти доступні ресурси (інформацію або дані) і виявити їх медіа-типи.

Сенсорні вузли в мережі самі виступають в ролі сервера, а не клієнта. Датчики безпосередньо маршрутизується, а обмін даними між датчиками і клієнтськими пристроями здійснюється за принципом "один до одного". Для реалізації цього протоколу датчики повинні бути здатні приймати пакети даних і відповідати на них.

2.1.4. DDS

Розроблений Object Management Group (OMG), Data Distribution Service є протоколом прикладного рівня M2M для систем реального часу. Заснований на схемі "публікація - підписка", як і MQTT, протокол має вузли, налаштовані як видавець, абонент або як обидва. Протокол не вимагає мережевого проміжного програмного забезпечення, тому видавці можуть публікувати інформацію з певних тем (наприклад, датчик температури може публікувати поточну температуру в певному місці), а сам протокол управляє її доставкою абоненту (наприклад, мобільний пристрій показує поточну температуру в певному місці). Реалізація протоколу не вимагає мережевого програмування, оскільки протокол не вимагає перевірки існування та місцезнаходження вузлів, а також підтвердження доставки повідомлення.

2.1.5. XMPP

Extensible Messaging and Presence Protocol (XMPP) - це протокол обміну повідомленнями на основі XML. XML - це мова розмітки для кодування документів, які можуть бути як зрозумілим для людини, так і в форматі машинної мови. XML був розроблений для розширення HTML і додавання користувацьких тегів та елементів в веб. Як і розширення HTML, він дозволяє структурувати дані, а також збільшувати їх. Традиційно XMPP використовується для комунікацій в реальному часі, таких як миттєвий обмін повідомленнями, присутність, багатосторонній чат, голосові і відеодзвінки, спільна робота, синдикація контенту і т.д.

Використання XMPP для IOT дозволяє створювати масштабовані мережі в реальному часі між пристроями або речами. Речі (пристрої)

мають один або декілька вузлів, і кожен вузол має кілька (інформаційних) полів. Кожне поле містить значення, доступне для читання і запису. Вузли повинні «дружити» один з одним, посилаючи і приймаючи запити на «дружбу». Як тільки запит на від одного вузла прийнятий іншим, він може обмінюватись даними з другим. Якщо іншому вузлу також необхідно отримувати оновлення від першого вузла, він також повинен відправити запит на «дружбу» і отримати підтвердження. Коли обидва вузла стають взаємними «друзями» в мережі, це називається подвійний підпискою, в іншому випадку - односторонньої підпискою. Дані передаються між вузлами за принципом "один до одного", коли один вузол може читати або записувати значення полів іншого вузла.

2.1.6. AMQP

Як і XMPP, Advanced Message Queue Protocol (AMQP) також є відкритим стандартним протоколом прикладного рівня для проміжного програмного забезпечення, орієнтованого на повідомлення. Він використовується для передачі ділових повідомлень між додатками або організаціями та з'єднує системи, забезпечуючи бізнес-процеси необхідної їм інформацією і надійно передає інструкції, які досягають поставлених цілей.

Це сумісний і багатоплатформовий стандарт обміну повідомленнями. В архітектурі протоколу повідомлення разом із заголовком передається клієнтом брокеру або біржі. Таким чином, існує єдина чергу, в яку повідомлення передається виробником. Від біржі або брокера повідомлення може бути передано в одну або безліч черг. Заголовок в протоколі містить інформацію про кожний байт повідомлення. Він також містить інформацію про маршрутизації. Брокер

або біржа залишається відповідальним за читання заголовків, отримання, маршрутизацію і доставку повідомлень клієнтським застосуванням. Зв'язок в цьому протоколі залишається один до одного між двома вузлами.

2.1.7. RESTful HTTP

Representational State Transfer (REST) або RESTful – це протокол без статичних даних і комунікацій. Він дозволяє ідентифікувати веб-ресурси (інформаційні ресурси) по унікальним URL і надавати їх у вигляді HTTP, JSON або XML файлу. Ресурси в ньому представлені у вигляді URI структури каталогів. Клієнт може отримати доступ до ресурсів за допомогою повідомлень GET, PUT, DELETE або POST до уявлень, де уявлення - це об'єкти даних і їх атрибути в форматі HTML, XML або JSON. GET, PUT, DELETE і POST - це методи HTTP-запиту для отримання, зміни і відправки даних. Будучи комунікацією без стану, підтвердження про доставку повідомлення не надсилається і не приймається. Однак у відповідь на HTTP-запит може бути отриманий код стану, який вказує на успіх, перенаправлення, інформацію, помилку клієнта або помилку сервера.

2.1.8. MQTT-SN

MQTT-Sensor Network - це відкритий і легкий протокол видавник/підписник, розроблений спеціально для обмежених пристроїв, тобто бездротових сенсорних мереж. Бездротові сенсорні мережі, які не мають стека TCP/IP поверх нього, наприклад, бездротові сенсорні мережі на базі Zigbee або Bluetooth, можуть відправляти свої дані в Інтернет за

допомогою протоколу MQTT-SN.

MQTT-SN розроблений як максимально наближений до MQTT, але адаптований до особливостей бездротової середовища зв'язку, таким як низька пропускна здатність, велика кількість відмов з'єднання, мала довжина повідомлення і т.д. Він також оптимізований для реалізації на недорогих пристроях з батарейним живлення і обмеженими ресурсами обробки і зберігання даних.

У порівнянні з MQTT, в MQTT-SN були внесені наступні зміни.

- Імена тем замінені на ідентифікатори тем, що знижує накладні витрати при передачі.
- Теми не потребують реєстрації, оскільки вони заздалегідь зареєстровані.
- Повідомлення також розбиваються на частини, так що передається тільки необхідна інформація.
- Для економії енергії передбачена автономна процедура для клієнтів, які перебувають у стані сну. Повідомлення можуть бути буферізіровані і пізніше прочитані клієнтами, коли вони прокинуться.
- Клієнти підключаються до брокера через шлюзовий пристрій, що знаходиться всередині сенсорної мережі.

В архітектурі MQTT-SN існує три види компонентів MQTT-SN - клієнти MQTT-SN, шлюзи MQTT-SN (GW) і форвардери MQTT-SN. Клієнти MQTT-SN підключаються до сервера MQTT через MQTT-SN GW, використовуючи протокол MQTT-SN. MQTT-SN GW може бути інтегрований з MQTT-сервером, а може і не бути. У разі окремого GW, протокол MQTT використовується між сервером MQTT і MQTT-SN GW. Його основна функція - переклад між MQTT і MQTT-SN.

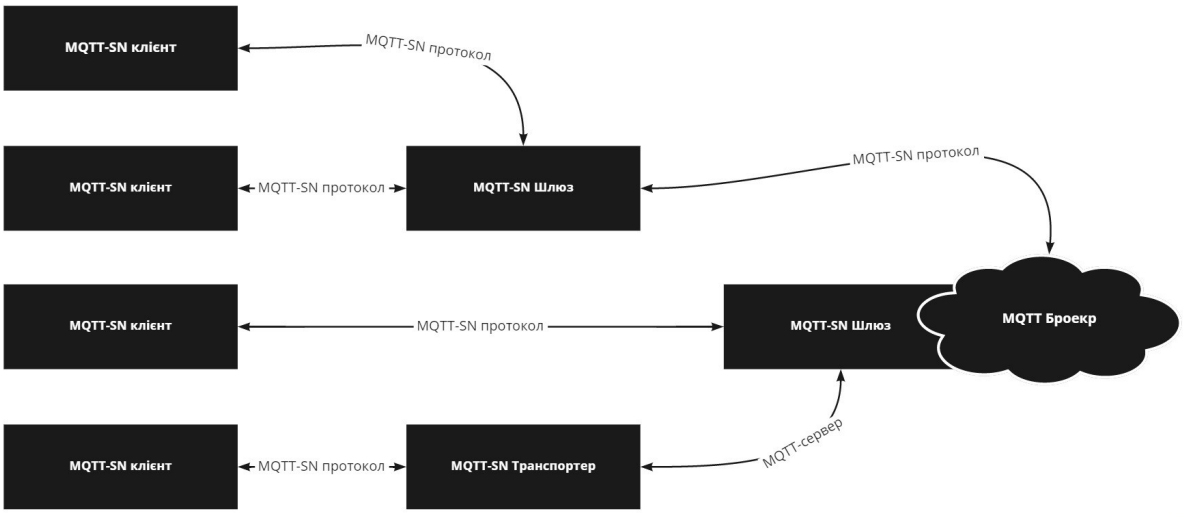


Рис. 2.2 архітектура протоколу MQTT-SN

Клієнти MQTT-SN також можуть отримати доступ до GW через форвардер в разі, якщо GW не підключений безпосередньо до їх мережі. Форвардер просто інкапсулює кадри MQTT-SN, які він отримує на бездротової стороні, і направляє їх без змін на GW; в зворотному напрямку він звільняє кадри, отримані від шлюзу, і також без змін відправляє їх клієнтам.

2.1.9. STOMP

Simple or Streaming Text Orientated Messaging Protocol (STOMP) - текстовий протокол для проміжного програмного забезпечення, орієнтованого на повідомлення. Він заснований на TCP і використовує команди, подібні HTTP. Він був розроблений для забезпечення сумісності між платформами, мовами і брокерами. Дані передаються між клієнтом і брокером у вигляді багаторядкових фреймів, що містять команду, заголовок і вміст. Команда може бути CONNECT, DISCONNECT, ACK,

NACK, SUBSCRIBE, UNSUBSCRIBE, SEND, BEGIN, COMMIT або ABORT.

2.1.10. SMCP

Simple Media Control Protocol (SMCP) - це стек протоколів CoAP для вбудованих пристроїв. Він розроблений на мові C і може використовуватися для надсилання та отримання асинхронних відповідей CoAP.

2.1.11. LLAP

Lightweight Local Automation Protocol - це простий і короткий протокол обміну повідомленнями, який може працювати на будь-якому засобі зв'язку. Протокол був розроблений для прямого обміну повідомленнями між вбудованими пристроями незалежно від протоколу низько-рівневих фізичних рівів рівня, що використовується пристроями.

2.1.12. SSI

Simple Sensor Interface (SSI) - це протокол прикладного рівня, призначений для зв'язку між комп'ютерами і датчиками. Заснований на UART, протокол дозволяє опитувати датчики і передавати дані датчиків. Дані передаються від датчиків до комп'ютерів у вигляді невеликого коду. У протоколі сполучення між датчиком і комп'ютером містить 2-байтовий заголовок і корисне навантаження. Тема містить однобайтовиу адресу і однобайтову команду або тип повідомлення.

2.1.13. LWM2M

Lightweight M2M (LWM2M) - це протокол керування пристроями для сенсорних мереж. Open Mobile Alliance (OMA) розробив цей протокол для управління легкими і малопотужними пристроями в різних мережах. Стек протоколів заснований на REST і CoAP.

2.1.14. M3DA

M3DA - це стек протоколів для обміну двійковими даними між серверами M2M і додатками, що працюють на вбудованих шлюзах. Він призначений для управління пристроями і активами, щоб оптимально використовувати пропускну здатність мережі.

2.1.15. XMPP-IOT

Як і XMPP використовується для взаємодії, XMPP-IOT - це стек протоколів для незалежної від машини M2M комунікації.

2.1.16. ONS 2.0

Служба імен об'єктів (ONS) - це стек протоколів для пошуку метаданих та послуг за заданим ідентифікаційним ключем GS1. GS1 - це стандарт обміну електронними бізнес-повідомленнями для автоматизації бізнес-операцій в ланцюжку поставок.

2.1.17. SOAP

Простий протокол доступу до об'єктів (SOAP) - це стек протоколів на основі XML, що використовується для реалізації веб-сервісів через Інтернет. Він використовує методи запиту HTTP для обміну повідомленнями між клієнтами по мережі.

2.1.18. Websocket

Інтерфейс WebSocket JavaScript визначає повнодуплексні односокетні з'єднання, через які можуть передаватися сполучення між клієнтом і сервером. Цей стандарт спрощує двосторонню веб-зв'язок і управління з'єднаннями.

2.1.19. Reactive Streams

Reactive Streams - це стек протоколів для асинхронної обробки потоків з неблокуючим зворотним тиском. Це специфікація, яка може бути реалізована на Java або JavaScript. Вона використовується для обробки потоків даних в реальному часі, обсяг яких невідомий.

2.1.20. HTTP/2

HTTP версії 2 це наступна версія протоколу HTTP. Він буде використовувати стиснення полів заголовків для зменшення затримки і дозволить здійснювати кілька одночасних обмінів по одному і тому ж з'єднанню.

2.1.21. JavaScript IOT

Це не стандарт ані протокол. Він відноситься до використання JavaScript і особливо Node.js (JavaScript на стороні сервера) з платами IOT для різних додатків. Деякі з проектів IOT на JavaScript включають NodeMCU, Jerryscript, Socket.IO, Ruff, NodeRed, KinomaJS, CHIRIMEN, Mosca, Nodebots, heimcontrol.js, Resin, UPM, i2C, node-http2, onoff, node-serialport, mdns, IoT .js, Favor, Serverless, noduino, duino, Pijs.io і ін.

2.2. Останні тенденції в алгоритмах стиснення даних

Стиснення даних є популярною проблемою серед дослідників завдяки можливості його використання практично в кожній функціональній області. Згідно з різними дослідженнями, стиснення даних ділиться на дві категорії: стиснення без втрат і стиснення з втратами. Стиснення з допомогою автокодіровщіков (AE) алгоритм відомий як стиснення з втратами [15]. Такий метод поліпшує терміну служби IoT-пристроїв, завдяки обмеженню енергоспоживання, оптимізації внутрішнього простору пам'яті, і ефективної передачі даних за допомогою бездротової технології було поставлено на чільне місце в експерименті[15], в той час як біомедичні сигнали, такі як ЕКГ, фотоплетізмографія і сліди дихання , були використані як набір даних. Потім продуктивність алгоритму стиснення AE була зрівняна з дискретним косинусним перетворення (DCT), вейвлет-перетворенням, аналізом головних компонент (PCA) і лінійною апроксимаціэю. Виходячи з результатів, отриманих в роботі [15], споживання енергії, необхідне для

стиснення сигналу онлайн, досить мало в порівнянні зі схемою, що використовує лінійні апроксимації. На думку дослідників, у цього проекту є і недолік. Він вимагає автономної фази навчання, що означає, що його необхідно бути виконана тільки один раз для кожного класу сигналів.

Веккіо, Джаффреда і Марчеллоні [16] запропонували наділити алгоритм стиснення без втрат (LEC) для бездротових сенсорних мереж двома простими схемами адаптації, які грунтувалися на новій концепції відповідним чином повернених таблиць без префіксів. Цей проект був спрямований на поліпшення енергоспоживання IoT-пристроїв під час передачі і прийому даних. За словами дослідників, основна ідея алгоритму LEC заснована на поділі алфавітів або цифр на групи, розміри яких будуть швидко збільшуватися. Таким чином, кодове слово LEC представляло собою гібрид унарних і бінарних кодів, де унарний код (код змінної довжини) визначав групу, а двійковий код (код фіксованої довжини) представляв індекс всередині групи, що майже аналогічно кодуванню Голомба і Еліаса. В їх проекті дослідники ввели ще два різних блока адаптації, відомих як GA-LEC і FA-LEC. Як набор даних використовувалися температура повітря, температура поверхні, відносна вологість і сонячне випромінення. Обидва блоки адаптації GA-LEC і FA-LEC дали високий відсоток ефективності.

Домінго-Прієто і ін. [17] зосередилися на взаємозв'язку між кількістю повідомлень, які посилає датчик, що впливає на термін служби батареї в. В даному проекті в якості стиснення даних застосовується кодування Хаффмана. Набір даних був витягнутий з IoT-пристрої, і продуктивність алгоритму стиснення Хаффмана була зрівняна з кластеризацией адрес. Згідно з отриманими результатами, марнотрати (що відправляють більше повідомлень) збільшили свій термін служби в порівнянні з вузлами, що відправляють менше повідомлень. На думку дослідників, реалізація кодування Хаффмана не представляється

тривіальним завданням в реальному розгортанні, в той час як кластеризація адрес є прямолінійною.

В роботі [18] Хсу і ін. Запропонували розробку системи WSN, яка збирає дані про віброприскорення з мостів в реальному часі і відправляє їх через інтернет. Цей проект був спрямований не тільки на віддалений моніторинг, а й на досягнення найменших виробничих витрат, найкоротшого часу установки і зручного переміщення. Для стиснення даних в даному проекті використовувалося кодування Хаффмана, а в якості набору даних використовувалася частота вібрації моста. Потім продуктивність алгоритму стиснення Хаффмана порівнювалася з трьома іншими алгоритмами стиснення, які включали LZ77, Run Length Encoding (RLE) і Rice coding. Згідно з отриманими результатами, кількість вузлів кінцевих пристроїв WSN зросла на 300% в порівнянні з аналогічною вибіркою, в якій алгоритм не був реалізований в мережі WSN. За даними дослідників, корисне навантаження бездротової передачі була зменшена приблизно на 60%, а кількість впроваджених вузлів збільшилася приблизно в три рази.

На думку деяких розробників, алгоритм стиснення MAS [19] є одним з перших алгоритмів, спеціально призначених для стиснення даних з плаваючою точкою. У цьому проекті дослідники зосередилися на використанні диспропорції в споживанні енергії між передачею і обробкою даних. Алгоритм стиснення MAS розшифровується як Minimalist, Adaptive, and Streaming. Показники вуглекислого газу, рівня води, радіоактивності, температури, вологості і рівня морськогу тиск були використані в якості набору даних в цьому експерименті. Дослідники провели шість різних типів експериментів: ступінь стиснення, час і енергія обчислень, енергія передачі, загальна споживана енергія, економія енергії і вимоги до пам'яті. Результати показали, що MAS дає більш високу ступінь стиснення, ніж алгоритми стиснення S-LZW і K-RLE.

Отже, використання алгоритму MAS дозволяє досягти кращих результатів у порівнянні з іншими алгоритмами. Крім того, MAS споживає найменшу кількість енергії, що передається серед інших алгоритмів. Незважаючи на те, що K-RLE споживає найменшу кількість обчислювальної енергії, він все одно споживає найбільшу кількість загальної енергії, оскільки відправляє значні обсяги даних по мережі. Тим часом, MAS споживав найменше енергії, доводячи тим самим, що MAS - найсильніший кандидат на стиск в WSN. Він також зміг заощадити більше енергії, ніж S-LZW і K-RLE. MAS споживав найбільшу кількість флеш-пам'яті, оскільки мав довший програмний код з двома поданнями для цілих і дійсних чисел. Однак це не викликало ніяких проблем, оскільки флеш-пам'ять була зарезервована для програмного коду, а не для довільного доступу. Більш того, MAS споживає всього 44 байта оперативної пам'яті. За словами дослідників, результати експерименту показали, що MAS забезпечує економію енергії в середньому на 54%, перемагаючи K-RLE і S-LZW при збереженні найвищих коефіцієнтів стиснення.

Інші дослідники [20] запропонували простий і ефективний алгоритм стиснення даних, який точно підходить для використання на вузлах WSN, де енергія, пам'ять і обчислювальні ресурси обмежені. Вони зосередилися на розробці простого алгоритму стиснення, який дозволяє скоротити пам'ять і обчислювальні ресурси вузла WSN. Як стиснення даних в цьому проекті був представлений простий алгоритм стиснення без втрат, який використовував комбінацію алгоритмів Хаффмана і JPEG. Як набору даних дослідники використовували температуру і вологість. Згідно з отриманими результатами, запропонований алгоритм стиснення досяг вищого ступеня стиснення, ніж S-LZW. Нарешті, що не менш важливо, було доведено, що цей алгоритм може перевершити gzip і bzip2.

2.3. Порівняння різних типів алгоритмів стиснення

Всі дані алгоритми стиснення будуються по-різному. Існують різні типи алгоритмів стиснення, деякі з них можуть підтримувати файли зображень, а деякі - тільки текстові файли. Деякі алгоритми стиснення даних не можуть зберегти оригінальність даних після декодування, наприклад, алгоритм Autoencoder [15]. Алгоритми стиснення даних аналізуються і відображаються в таблиці для порівняння. У таблиці 2.1 показано узагальнення порівняння між різними типами алгоритмів стиснення.

Таблиця 2.1

Узагальнення порівняння між різними типами алгоритмів стиснення

Опис	[15]	[16]	[17]	[18]	[19]	[20]
Рік	2015	2014	2014	2014	2013	2008
Мета розробки	Збільшує термін служби пристроїв IoT (обмеження енергії, оптимізація внутрішнього простору пам'яті та ефективна передача даних).	Покращує енергоспо-живання пристроїв IoT	Зосереджується на взаємозалеж-ності між часом роботи акумулятора пристрою та кількістю повідомлень, які кожен з них надсилає	Віддалений моніторинг, низька вартість, короткий час встановлення та зручне переміщення	Використати диспропорційність у споживанні енергії між передачею та обробкою даних	Простий алгоритм стиснення, придатний для зменшення пам'яті та обчислюваль-них ресурсів вузла WSN
Тип стиснення даних	Автокодер (AE) - стиснення з втратами	GAS-LEC, FAS-LEC	Кодування Хаффмана	Кодування Хаффмана	Мінімальний, адаптивний та потоковий алгоритми стиснення (MAS)	Простий алгоритм стиснення без втрат - поєднання Хаффмана та кодування JPEG
Порівняння методу стиснення інших даних	Дискретне косинусне перетворення s (DCT), DCT-LW, дискретне вейвлет-перетворення (DWT), DWT LW, LTC (на основі лінійного наближення), Аналіз принципових компонентів (PCA)	Алгоритм стиснення без втрат (LEC)	Кластеризація адрес (стиснення нових даних)	LZ77, кодування довжини пробігу (RLE)	S-LZW, K-RLE	S-LZW, gzip, bzip2

Продовження таблиці 2.1

Набір даних	Біомедичний сигнал (ЕКГ, фотоплетиз-мографічні та дихальні сліди)	Повітря, поверхня, температура, сонячне випромінювання, відносна вологість	Витягнуто з розгортання пристрою IoT	Частота вібрацій мосту	Вуглекислий газ, рівні води, радіоактивність, температура, вологість, тиск у морі	Температура, вологість
Переваги	Легкість в розгортанні, висока ефективність стиснення, низьке споживання енергії	Покращує ефективність стиснення LEC	Має більші загальні покращення, оскільки він використовував пакети змінного розміру, а також стискав дані	Корисне навантаження бездротової передачі можна зменшити на 60%, кількість вузлів реалізованої мережі можна збільшити втричі	Краще щодо ступеня стиснення, швидкості стиснення, вимог до пам'яті та енергозбереження	Вищий ступінь стиснення
Недоліки	Потрібен етап офлайн навчальний (має виконуватися лише один раз для кожного класу сигналів)	Не зазначено	Не здається тривіальним завданням у реальному розгортанні	Не зазначено	Не зазначено	Не зазначено

Виходячи з узагальнення в таблиці 1, проект [15] має найбільший потенціал і є більш відповідним проектом для використання в якості прототипу даного експерименту. Подібність проекту [15] з попередніми дослідженнями можна побачити в перевагах, методі і спрямованості. Проект [15] показав кращий коефіцієнт стиснення, що також схоже з проектом [10]. Що стосується методу, то проект зосередився на реалізації алгоритму стиснення даних за допомогою нейронних мереж. Цей підхід має недолік – він може підходити для одних данних, що передаються в IoT (фото- та відео-спостереження, фіксації) та зовсім не підходити для інших (показники температури, вологості та інші текстові данні). Такий тип проекту можна застосувати для зниження енергоспоживання при передачі даних, що було схоже на проекти [16], [17] і [19] а також, за рахунок шифрування, для уникнення деяких проблем безпеки в IoT.

Отже, необхідно більш детально дослідити перспективу використання нейронних мереж для обробки трафіку.

2.4. Огляд рекурентних нейронних мереж

2.4.1. RNN

Рекурентні нейронні мережі (RNN) - це клас нейронних мереж, який ефективний для моделювання даних послідовності [21], таких як тимчасові ряди або природна мова. На відміну від багатошарових перцептронів, рекурентні мережі можуть використовувати свою внутрішню пам'ять для обробки послідовностей довільної довжини. Тому мережі RNN застосовні в таких завданнях, де щось цілісне розбите на частини, наприклад: розпізнавання рукописного тексту або розпізнавання

мови. Було запропоновано багато різних архітектурних рішень для рекурентних мереж від простих до складних. Останнім часом найбільшого поширення набули мережу з довготривалою і короткочасною пам'яттю (LSTM) і керований рекурентний блок (GRU).

Схематично RNN використовуються наступним чином: на кожну ітерацію циклу for на вхід моделі подається попередній внутрішній стан і нова порція даних. Наприкінці береться вихід нейронної мережі, який відповідає усій послідовності.

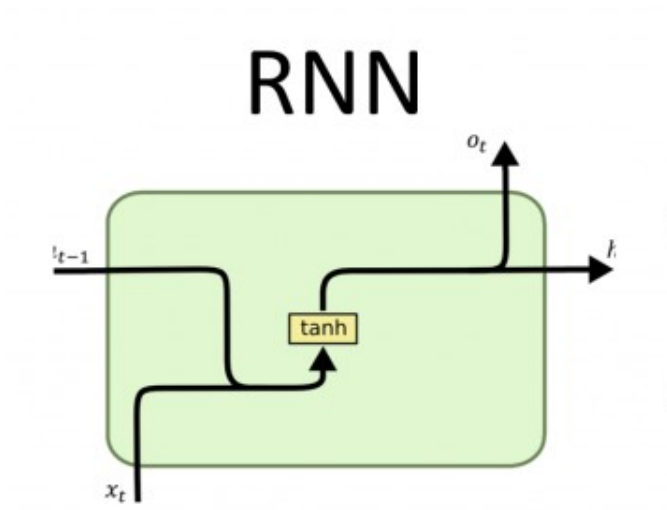


Рис. 2.1 Схема роботи ланцюгу рекурентної нейронної мережі

2.4.2. LSTM

Тривала короткочасна пам'ять (LSTM) - це архітектура штучної рекурентної нейронної мережі (RNN), яка використовується в області глибокого навчання. На відміну від стандартних нейронних мереж з прямим зв'язком, LSTM має звортній зв’язок з попередніми шарами мережі [21]. Вона може обробляти не тільки окремі точки даних (наприклад, зображення), а й цілі послідовності даних (такі, як мова або

відео). Наприклад, LSTM застосовна для таких завдань, як розпізнавання несегментованого рукописного тексту, розпізнавання мови і виявлення аномалій в мережевому трафіку або як системи виявлення вторгнень (intrusion detection systems; IDS).

Звичайний блок LSTM складається з комірки, вхідних вентилей, вихідних вентилей і вентилей забування. Комірка запам'ятовує значення за довільні проміжки часу, а три гейта (вентилі) регулюють потік інформації в комірку і з неї.

LSTM-мережі добре підходять для класифікації, обробки і прогнозування на основі даних часового ряду, оскільки між важливими подіями тимчасового ряду можуть бути проміжки невідомої тривалості. LSTM були розроблені для вирішення проблеми зникаючого градієнта, яка може виникнути при навчанні традиційних RNN. Відносна нечутливість до довжини проміжку є перевагою LSTM перед RNN, прихованими моделями Маркова та іншими методами навчання послідовностей в численних додатках.

У теорії, класичні RNN можуть відстежувати довільні довгострокові залежності у вхідних послідовностях. Проблема з класичним RNN носить обчислювальний (або практичний) характер: при навчанні класичної RNN методами зворотного поширення градієнти зворотного поширення можуть "зникнути" (тобто прагнути до нуля) або "вибухнути" (тобто прагнути до нескінченності), оскільки в процесі обчислень використовуються числа кінцевої точності. RNN, що використовують блоки LSTM, частково вирішують проблему зникаючого градієнта, через те, що блоки LSTM дозволяють градиентам текти без змін. Однак мережі LSTM все ще можуть страждати від проблеми вибухає градієнта.

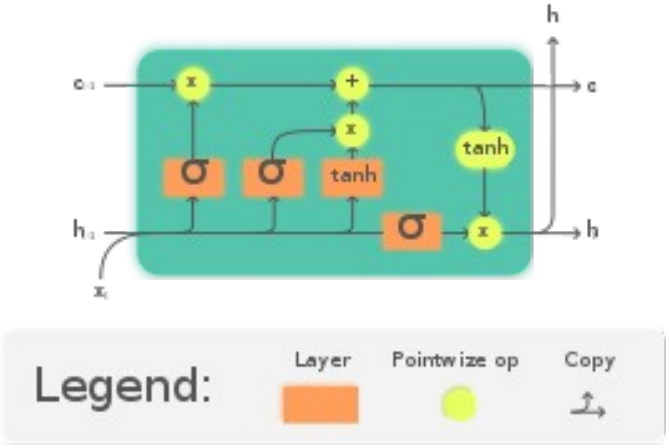


Рис. 2.2 Схема роботи ланцюгу LSTM

2.4.3. GRUs

Gated recurrent units (GRUs) - це механізм будування в рекурентних нейронних мережах, представлений в 2014 році Kyunghyun Cho et al. GRU схожий на довгу короткочасну пам'ять (LSTM) зі механізмом забування, але має менше параметрів, ніж LSTM, оскільки у нього немає вихідного стану. Продуктивність GRU в деяких задачах моделювання поліфонічної музики, моделювання мовних сигналів і обробки природної мови виявилася аналогічною продуктивності LSTM [21]. GRU показали кращу точність на деяких невеликих і менш частих наборах даних.

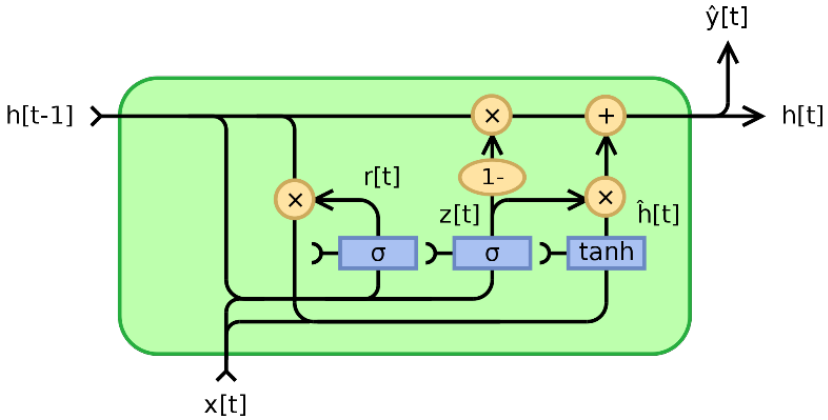


Рис. 2.3 Схема роботи ланцюгу GRU

2.5. Стискання без втрат алгоритмом DeepZIP

DeepZIP поєднує RNN з методами теорії інформації для створення ефективного компресора без втрат.

RNN для оцінювання ймовірності:

Перший компонент моделі DeepZip називається RNN для оцінювання ймовірності. Як випливає з назви, це RNN, яка на кожному часовому кроці приймає на вхід символ вихідної послідовності. Тут символ - це будь-який будівельний блок вхідної послідовності; це може бути біт або байт, основа в ланцюзі ДНК, англійська літера, що завгодно. Після обробки символу RNN для оцінювання ймовірності видає вектор. Кожен запис в векторі - це передбачення RNN про те, наскільки ймовірно, що певний символ з'явиться в послідовності. При кодуванні послідовності бітів вихід [0.25, 0.75] буде означати, що модель вважає, що наступним бітом буде 1 з ймовірністю 75%. Потім RNN можна показати наступний символ в послідовності, для якого вона видасть ймовірності, з огляду на символи, які їй були показані раніше.

RNN для оцінювання ймовірності в DeepZip цікавий в порівнянні з іншими нейронними мережами. Більшість мереж використовують набір даних для навчання, щоб дізнатися свої внутрішні параметри. Коли навчання завершено, параметри заморожуються, і тільки після цього мережа використовується для прогнозування нових вхідних даних. RNN для оцінювання ймовірності, однак, не проходить такого навчання, перш ніж йому показують новий вхід. Коли RNN отримує щось для кодування, вона починає з випадковими параметрами. В процесі обробки символів на вході вона не тільки оновлює свій приховане стан за звичайними правилами RNN, а й оновлює свої вагові параметри, використовуючи втрати між своїми імовірнісними прогнозами і справжнім символом. RNN

для оцінювання ймовірності не тільки намагається вивчити, які залежності існують в новій послідовності, вона повинен дізнатися, як вивчати ці залежності.

Арифметичний кодер:

RNN виробляє ймовірності символів на кожному кроці вхідної послідовності, але алгоритму потрібен спосіб фактично перевести ці ймовірності в кодування вхідних даних. Для цього використовується класичний інструмент теорії інформації, званий арифметичним кодером.

Арифметичний кодер використовує числовий діапазон для подання вхідної послідовності. Спочатку діапазон становить від 0,0 до 1,0, і оновлюється наступним чином:

1) Прогнозування ймовірності появи кожного символу наступним у вхідній послідовності. Ймовірно, вона буде отримана з якоїсь статистичної моделі, наприклад, з RNN для оцінювання ймовірності.
2) Поділ поточного діапазону енкодера на підрозділи. Для кожного можливого символу буде один підрозділ, а довжина підінтервалу пропорційна ймовірності відповідного символу (отриманого на кроці 1).
3) Зчитування наступного символу з вхідної послідовності.
4) Установка поточного діапазону енкодера на підінтервал цього символу. Тепер діапазон енкодера строго менше, ніж раніше. Чим вище передбачена можливість символу, тим довше буде новий діапазон.
5) Якщо у вхідній послідовності є ще символи, повертаємося до кроку 1. Енкодер буде продовжувати оновлювати свій діапазон для кожного символу у вхідній послідовності.

Після зчитування всієї вхідної послідовності у енкодера залишається інтервал. Остаточне кодування вхідної послідовності - це уявлення двійковій дробу для будь-якого числа в цьому діапазоні, інакше кажучи кожен новий біт в послідовності ділить інтервал навпіл, а саме 0 переведе інтервал в його ліву частину, а 1 – в праву.

Коли необхідно декодувати цю стислу послідовність, використовується арифметичний кодер в зворотному порядку. Починаючи знову з діапазону від 0,0 до 1,0, кодер генерує вихідну послідовність наступним чином:

1) Прогнозування ймовірності появи кожного символу наступним у вихідний послідовності. Важливо відзначити, що ці ймовірності повинні бути отримані тією ж моделлю, яка створила ймовірності в енкодері.
2) Розподіл поточного діапазону енкодера на підрозділи. Знову ж таки, для кожного можливого символу буде один підрозділ, а довжина підрозділу пропорційна ймовірності, передбаченої на кроці 1 для цього символу.
3) Визначення підрозділу, який містить закодоване число. Слід пам'ятати, що кодування вхідної послідовності - це двійкове подання числа від 0,0 до 1,0, і це число містилося в кінцевому діапазоні арифметичного кодера на етапі кодування.
4) Додавання символа, присвоєного цьому підінтервалу, в вихідну послідовність. Кінцевий діапазон кроку кодування містився в діапазоні, обраному для кожного попереднього кроку, тому який би підрозділ символу ні містив вхідний код, він повинен бути в вихідний послідовності.

5) Установка поточного діапазону кодера на цей підрозділ. Тепер діапазон кодера точно такий же, яким він був під час відповідного кроку процесу кодування.
6) Якщо є ще символи для декодування, необхідно повернутися до кроку 1. Кінець послідовності може бути позначений спеціальним символом кінця повідомлення, або заздалегідь можна знати кількість символів, які повинні бути на виході. Якщо цього символуконечного символа немає, або було досягунто выдомий вінець послідовності, декодування продовжується.

2.6. Метод кодування

На практиці ми зазвичай не знаємо точну довжину вхідній послідовності заздалегідь, тому ми, ймовірно, будемо використовувати спеціальний символ кінця повідомлення, щоб вказати кодеру, коли він повинен припинити додавати нові символи в вихідну послідовність; в іншому випадку кодер міг би продовжувати додавати символи нескінченно.

DeepZip об'єднує RNN для оцінювання ймовірності і арифметичний кодер для кодування вхідних послідовностей, як показано на рисунку 2.4. [22].

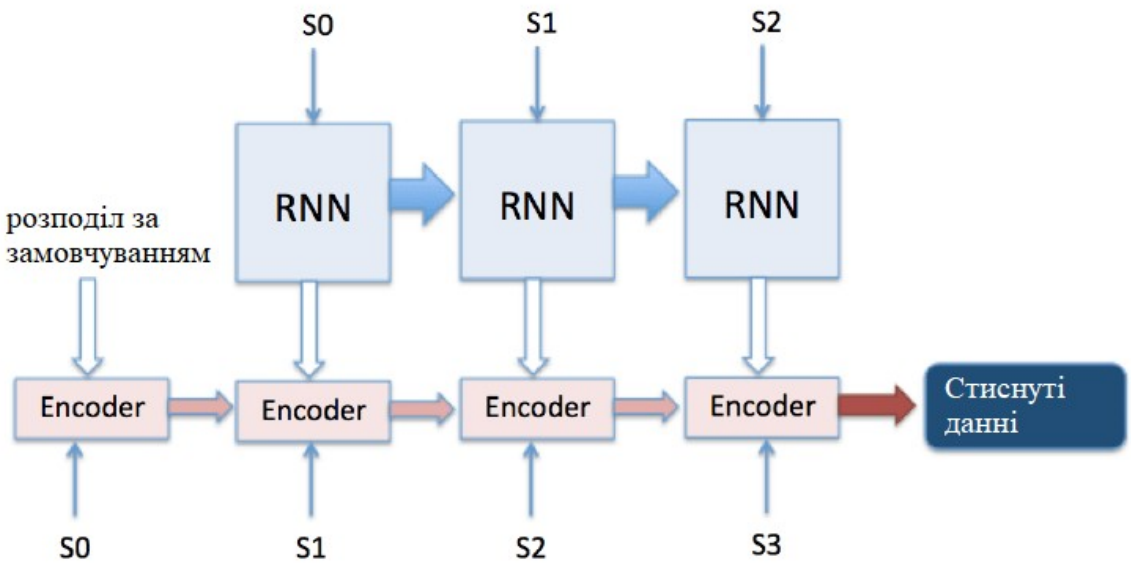


Рис. 2.4 Схема стиснення даних

Після ініціалізації RNN для оцінювання ймовірності з випадковими вагами арифметичний кодер кодує перший символ у вхідній послідовності, використовуючи розподіл символів за замовчуванням. Потім цей перший символ передається в RNN для оцінювання ймовірності, який видає ймовірності для наступного символу. Арифметичний кодер використовує ці ймовірності для кодування другого символу. Ваги RNN для оцінювання ймовірності оновлюються шляхом порівняння передбачених їм ймовірностей для другого символу з фактичної ідентичністю другого символу. Потім другий символ вводиться в RNN для оцінювання ймовірності, яка видає ймовірності для третього символу, і процес триває до тих пір, поки вхідна послідовність не буде повністю закодована.

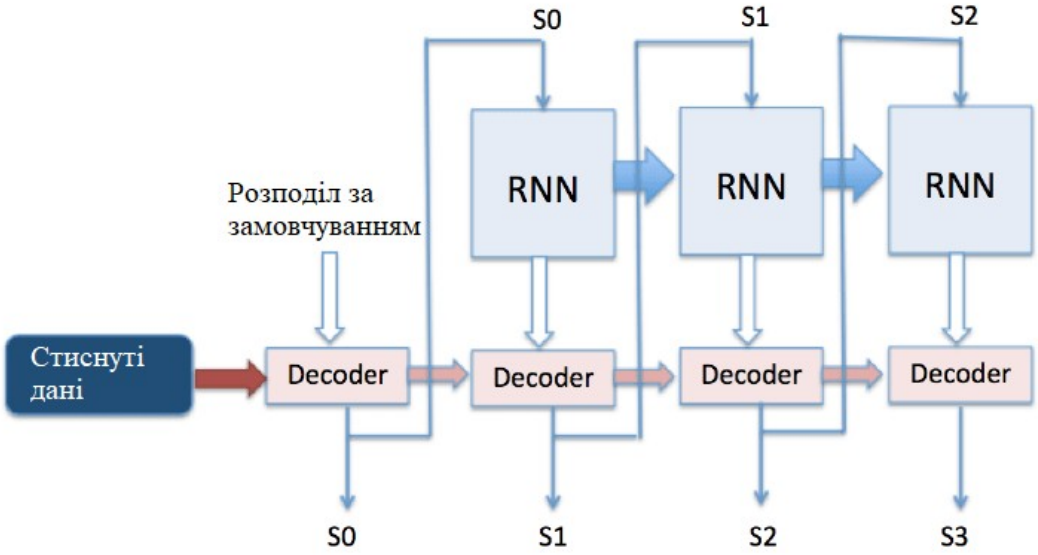


Рис. 2.5 Схема декодування даних

Важливо відзначити, що RNN для оцінювання ймовірності ініціалізується з тими ж вагами, які використовувалися на початку кодування; це можна зробити, відправивши випадкове значення, яке використовувалося під час кодування, разом з фактичним кодуванням вхідних даних. Потім арифметичний енкодер використовує розподіл символів за замовчуванням для розбору першого символу з кодування. Даний символ передається в RNN для оцінювання ймовірності, яка видає набір ймовірностей для наступного символу; при правильній ініціалізації це будуть точні ймовірності, видані RNN під час першого кроку кодування. Ці ймовірності використовуються кодером для розбору другого символу, який використовується для поновлення ваг RNN. Оновлення ваг повинно бути точно таким же, як і після першого кроку кодування. Після цього другий символ передається в мережу, яка видає ймовірності для третього символу, і процес триває до тих пір, поки кодер не прочитаєте символ кінця повідомлення.

Висновки:

- Розглянуто протоколи передачі даних в мережах IoT;
- Порівняно найпопулярніші методи стиснення даних;
- На основі одного з найперспективніших методів обробки даних обрано подальший шлях розвитку методу;
- Розглянуто різні типи рекурентних нейронних мереж та їх застосування для кодування даних без втрат.

При оцінці вже існуючих методів помітно значне просунення із року в рік у цьому питанні, що обумовлено розробкою нових технологій та методів обробки даних. Як прототип обрано технологію стиснення інформації за допомогою рекурентної нейромережі з використанням методу DeepZIP.

РОЗДІЛ 3

ПРОВЕДЕННЯ ЕМПІРИЧНОГО МОДЕЛЮВАННЯ ТА НАДАННЯ АНАЛІТИЧНОЇ ОЦІНКИ ЗАПРОПОНОВАНОМУ МЕТОДУ.

3.1. Метод BERT

У попередньому розділі було розглянуто алгоритм стиснення із застосуванням нейронних мереж DeepZip. Однак при його розробці основною платформою розглядалися класичні обчислювальні системи, такі як веб-сервера і персональні комп'ютери. Для застосування в IoT-пристроях модель може виявитися занадто важкою: як в сенсі необхідної пам'яті, так і в сенсі споживання обчислювальних ресурсів.

В області нейронних мереж проблема важких моделей встає не вперше. Прикладом можна стати BERT [23], архітектура нейронної мережі від Google, яка дозволяє розпізнавати сенс досить складних речень і навіть виділяти з них смислові значення окремих словосполучень. Саме по собі навчання BERT відбувається дуже довго, в результаті чого виходить досить об'ємна мережа, яку не вийшло б використовувати для розбору всіх пошукових запитів. З іншого ж боку, зважаючи на технічні та архітектурні обмеження, немає можливості якісно навчити більш просту архітектуру відразу. Таким чином, є можливість навчити велику модель, проте зважаючи на свій розміру остання не може бути застосована на практиці.

З цієї причини застосовується наступний принцип: спочатку відбувається навчання об'ємної моделі до прийнятного рівня, а вже потім відбувається навчання, стиснення та полегшеної версії моделі на вихідну модель. Спрощено кажучи, ми не намагаємося відразу навчитися

відповідати на питання про текст, а спочатку вчимося розуміти його сенс, спостерігаючи за тим, як його розбирає батьківська модель, і навчаючись повторювати за нею.

3.2. Проведення емпіричного моделювання та аналітична оцінка нейромереж

Прийнято рішення відмовитися від навчання паралельно кодування на користь попередньо навченій моделі, що дозволить по-перше закласти в модель особливості виду даних, що передаються, а по-друге дозволить застосувати вищезазначених принцип стиснення рекурентною нейронною мережею. Таким чином першим кроком буде навчання RNN на згенерованому наборі даних, що плануються до передачі. Таким чином буде отримана нейронна мережа, з деяким рівнем надмірності, який надалі буде усунутий за допомогою стиснуті до нижчих розмірностей.

Було вирішено провести експеримент з використанням найбільш поширеного прикладного формату JSON. Він дозволяє задавати досить складні структури даних, при цьому будучи досить простим і мінімалістичним на противагу громіздкому XML. Нижче наведено приклад даних що передаються.


```
{
 "device_id": "abcdef12345",
 "sensors": [
  {
   "sensor_id": "abcdef321321",
   "type": "temperature",
```

```
      "unit": "celcius",
      "value": 27.62
    },
    {
      "sensor_id": "baca123baca321",
      "type": "pressure",
      "unit": "millibars",
      "value": 990
    },
    {
      "sensor_id": "sens_humid_2",
      "type": "humidity",
      "unit": "percent",
      "value": 91
    }
  ]
}
```


Лістинг 3.1 Приклад JSON запису

Його особливості будуть використані для ефективного стиснення даних. Для цього була розроблена утиліта для генерації вхідних даних. Генерація відбувається за допомогою рекурсивної функції generate (size_left, max_depth), що повертає випадковий об'єкт, який при серіалізації в JSON дасть рядок не довше size_left байт і глибиною не

більше max_depth. На кожному кроці рекурсії ми намагаємося додавати в об'єкт випадкові ключі, частково зі словника, для яких значеннями будуть або випадкові рядки, або інші ж випадкові об'єкти, або масиви з випадкових об'єктів.

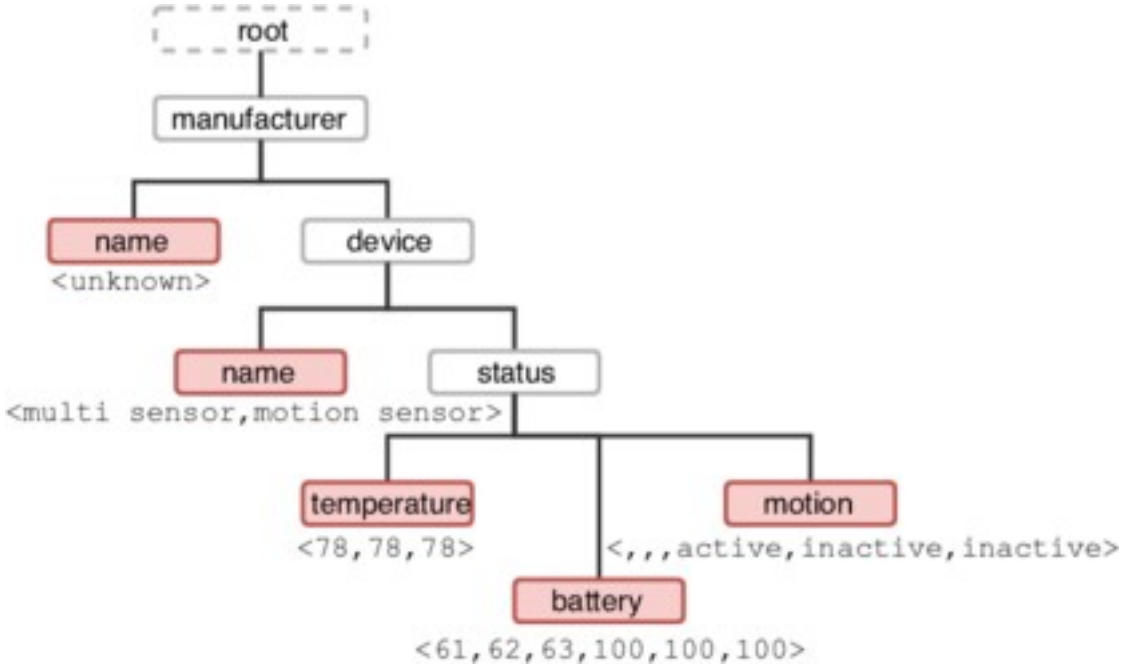


Рис. 3.1 Структура JSON, відповідна однієї з варіацій рекурсивного виклику [24]

Оригінальна архітектура DeepZIP [25], як згадувалося вище, заснована на навчанні паралельно самому процесу стиснення. Так, для стиснення із застосуванням даного алгоритму необхідно оновлювати ваги після кожної порції вхідних даних, тобто модель дізнається особливості даних у міру проходу по ним. У нашому ж випадку робота йде з форматом JSON із заздалегідь відомим набором ключів, що дозволяє закласти знання про структуру вхідних даних ще до того, як мережа буде застосовуватися на практиці. Так, в ході даної роботи була навчена велика нейронна мережа, після чого, вже на основі великої, була навчена простіша модель, яка в подальшому може застосовуватися на цільових IoT-пристроях.

Отже, буде використано для навчання набір випадково згенерували за допомогою описаного вище алгоритму JSON-рядків з IoT-подібними даними. Згенеровано 200000 рядків розміру до 10 (приблизно 2 МБ даних), 20000 рядків розміру до 50, 10000 рядків розміру до 100 і 5000 рядків розміру до 10000. Дана вибірка використовувалася для навчання великої архітектури мережі.

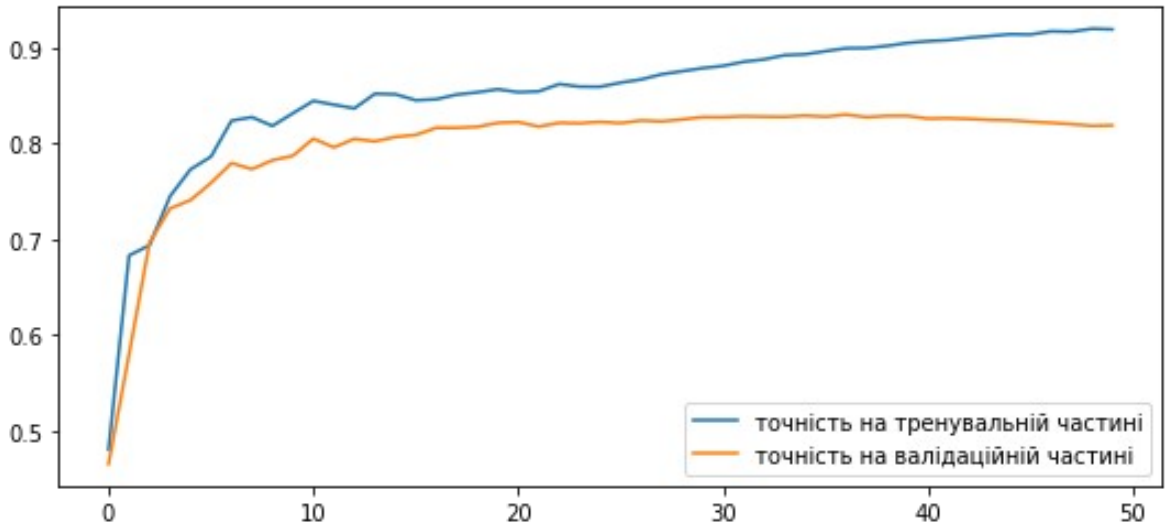


Рис. 3.2 Порівняння точностей мережі

Для навчання маленької моделі згодом буде використана частина цієї вибірки, а також додатково згенерований випадковий набір JSON-рядків. Так, для навчання маленької архітектури використовується 50% навчальної вибірки великий архітектури, а також решту 50%, згенеровані заново щоб уникнути перенавчання.

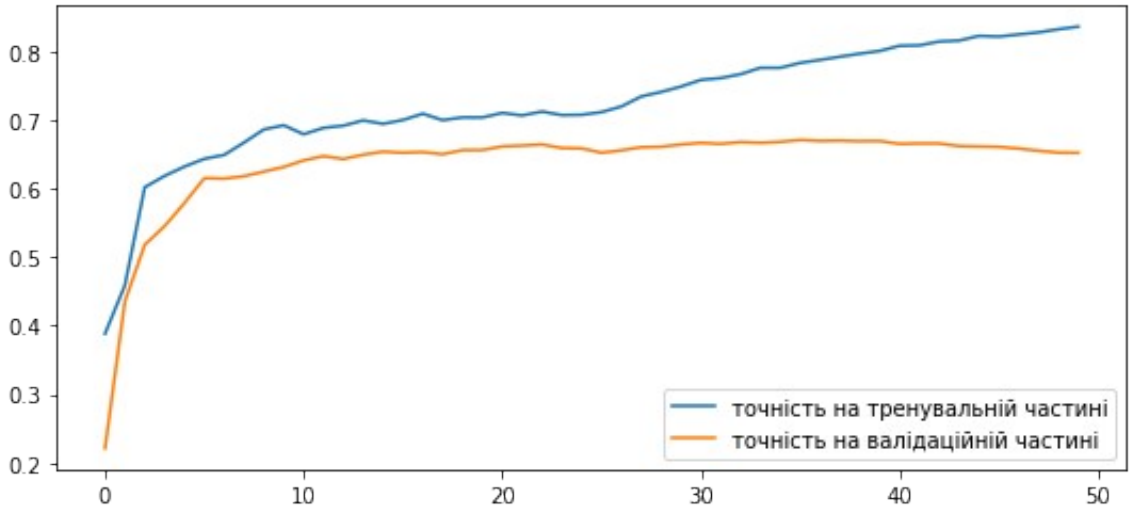


Рис. 3.2 Порівняння точностей нової мережі

При цьому з огляду на відмінності архітектур маленька виконується в 3 рази швидше. Вище наведені графіки, що показують розходження в ступені стиснення в залежності від числа епох для великої і для маленької архітектур.

Спрощення архітектури DeepZIP дозволило отримати більший ступінь стиснення конкретно для IoT-специфічних даних. При цьому навчена на кінцеві ваги великої мережі менша архітектура дала приблизно на 12% менший ступінь стиснення при прискоренні в 3рази.

Висновки:

У цьому розділі було запропоновано вдосконалення моделі під потреби систем IoT та її подальша реалізація і порівняння з оригиналом. При аналітичній оцінці виявлено незначне зменшення проценту стиснення данних і точності моделі та вагому різницю в співвідношенні між швидкостями обробки даних, їх кодування та декодування.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1) Проведено огляд мереж інтернету речей та їх проблем, що стосуються кодування даних, безпеки та енергоспоживання, за рахунок чого було обрано напрямок у вирішенні проблем кодування даних мережі Інтернету речей.
2) Проаналізовано сучасні рішення щодо кодування даних мереж Інтернету речей, на основі чого було найбільш перспективний метод для стискання даних, що використовує непромережений автоенкодер.
3) Удосконалено нейромережевий автоенкодер за рахунок використання алгоритму, що додатково включає в себе арифметичний кодер та подальшого навчання нової моделі на вихідні данні повноцінного автоенкодеру.
4) Виконано емпіричне моделювання запропонованого методу, та проведено ряд порівнянь, що підтверджують релевантність запропонованого методу. При подальшій оцінці та порівнянні виявлено, що нова, навчена модель, має більший ступінь стискання даних при меншій потребі у часі, що дозволяє говорити про успішне досягання поставленої мети.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gao, C.; Ling, Z.; Yuan, Y. The research and implement of smart home system based on Internet of things. In Proceedings of the 2011 International Conference on Electronics, Communications and Control (ICECC), Ningbo, China, 9–11 September 2011;

2. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Sensing as a service model for smart cities supported by Internet of things. Trans. Emerg. Telecommun. Technol. 2014.

3. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for smart cities. IEEE Internet чThings J. 2014,1, 22–32. Zhang, M.; Yu, T.; Zhai, G.F. Smart transport system based on “The Internet of Things”. Appl. Mech. Mater. 2011.

4. Zhou, Z.; Zhou, Z. Application of Internet of Things in agriculture products supply chain management. In Proceedings of the 2012 International Conference on Control Engineering and Communication Technology (ICCECT), Liaoning, China, 7–9 December 2012.

5. Mashal, I.; Alsaryrah, O.; Chung, T.Y.; Yang, C.Z.; Kuo, W.H.; Agrawal, D.P. Choices for interaction with things on Internet and underlying issues. Ad Hoc Netw. 2015.

6. Miao, Y.; Bu, Y.X. Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid. In Proceedings of the 2010 International Conference on Advances in Energy Engineering (ICAEE), Beijing, China, 19–20 June 2010.

7. Said, O.; Masud, M. Towards Internet of things: Survey and future vision. Int. J. Comput. Netw. 2013.

8. Madakam, S.; Ramaswamy, R.; Tripathi, S. Internet of Things (IoT): A literature review. J. Comput. Commun. 2015.

9. Khan, R.; Khan, S.U.; Zaheer, R.; Khan, S. Future Internet: The Internet of things architecture, possible applications and key challenges. In Proceedings of the 2012 10th International Conference on Frontiers of Information Technology (FIT), Islamabad, India, 17–19 December 2012.

10. Sethi, P.; Sarangi, S.R. Internet of Things: Architectures, Protocols, and Applications. J. Electr. Comput. Eng. 2017.

11. Ashraf, Q.M.; Habaebi, M.H. Autonomic schemes for threat mitigation in Internet of Things. J. Netw. Comput. Appl. 2015.

12. Canzanese, R.; Kam, M.; Mancoridis, S. Toward an automatic, online behavioral malware classification system. In Proceedings of the IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), Philadelphia, PA, USA, 9–13 September 2013.

13. Bilge, L.; Dumitras, T. Before we knew it: An empirical study of zero-day attacks in the real world. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; ACM: New York, NY, USA, 2012.

14. Kaur, R.; Singh, M. A survey on zero-day polymorphic worm detection techniques. IEEE Commun. Surv. Tutor. 2014,16,.

15. D.D. Testa and M. Rossi, IEEE Signal Processing Letters, 2015.

16. M. Vecchio, R. Giaffreda and F. Marcelloni, IEEE Transactions on Wireless Communications, 2014.

17. M. Domingo-Prieto, B. Martinez, M. Monton, I. Vilajosana-Guillen, X. Vilajosana-Guillen and J. Arnedo-Moreno, 2014 Eighth International

Conference on Complex, Intelligent and Software Intensive Systems (CISIS), 2014.

18. C-H. Hsu, C-T. Lin, H-P. Tserng and J-Y. Han, 2014 IEEE World Forum on Internet of Things (WFIoT), pp 2014.

19. M.E. Assi, A. Ghaddar, S. Tawbi and G. Fadi, International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC), 2013.

20. F. Marcelloni and M. Vecchio, IEEE Communications Letters 2008.

21. Рекуррентные нейронные сети (RNN) с Keras // електрон. текст. дані
URL: https://www.tensorflow.org/guide/keras/rnn (дата звернення: 01.06.2021)

22. M. Goyal, K. Tatwawadi, S. Chandak, I. Ochoa, Data Compression Conference (DCC), May 2019

23. Compressing BERT for faster prediction // електрон. текст. дані URL: https://blog.rasa.com/compressing-bert-for-faster-prediction-2/ (дата звернення: 01.06.2021)

24. Convert the sample JSON file from a tree to a table. // електрон. текст. дані
URL: https://www.researchgate.net/figure/Convert-the-sample-JSON-file-from-a-tree-to-a-table_fig3_332591327 (дата звернення: 01.06.2021)

25. NN based lossless compression – DeepZip // електрон. текст. дані URL: https://github.com/mohit1997/DeepZip (дата звернення: 01.06.2021)