ISSN 2219-9365

https://doi.org/10.31891/2219-9365-2026-86-40
УДК 004.7:004.312.4

ПАЛІЙ Сергій
Київський національний університет імені Тараса Шевченка
https://orcid.org/0000-0001-9742-1116
e-mail: paliy@fit.knu.ua
БІДОЧКА Віталій
Київський національний університет імені Тараса Шевченка
https://orcid.org/0009-0005-9224-3847
e-mail: bidochkav@fit.knu.ua

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ SBE ТА PROTOBUF У ВИСОКОНАВАНТАЖЕНИХ IOT-ШЛЮЗАХ

Швидке поширення пристроїв Інтернету речей висуває дедалі жорсткіші вимоги до інфраструктури шлюзів, які повинні безперервно збирати, кодувати та передавати великі обсяги телеметричних даних у режимі реального часу. В умовах, коли кількість підключених кінцевих точок сягає мільйонів, вибір методу бінарної серіалізації даних стає критично важливим архітектурним рішенням, що безпосередньо впливає на пропускну здатність, затримку, використання процесора та споживання пам'яті. Незважаючи на широке застосування методів серіалізації даних у розподілених системах, їх ретельне емпіричне порівняння в контексті високонавантажених IoT-шлюзів залишається недостатньо дослідженим у сучасних дослідженнях. У статті представлено результати порівняльного аналізу двох методів бінарної серіалізації даних - Simple Binary Encoding (SBE) та Protocol Buffers (Protobuf), в умовах високонавантаженого середовища шлюзів Інтернету речей. Дослідження проводилося на базі міні-комп'ютера Raspberry Pi 5, який виконував роль IoT-шлюзу. Програмне забезпечення, реалізоване мовою програмування Go, приймало повідомлення телеметрії через socket-з'єднання та виконувало їх десеріалізацію. Тестування охоплювало два типи повідомлень з максимальним розміром корисного навантаження. Ключовими метриками вимірювання були загальний час обробки повідомлень та рівень використання центрального процесора. Отримані результати свідчать про те, що жоден із досліджуваних методів не має безумовної переваги в продуктивності. При малих розмірах повідомлень з високою інтенсивністю, що є типовим для щільних потоків телеметрії сенсорів, метод SBE демонструє значно нижчий час обробки завдяки архітектурі з нульовим копіюванням даних в оперативній пам'яті та статично скомпільованим схемам. Натомість при великих розмірах повідомлень і високому навантаженні метод SBE поступається Protobuf як за часом обробки, так і за рівнем споживання ресурсів процесора. Отримані результати підтверджують, що оптимальний вибір методу серіалізації має визначатися на основі ретельної оцінки профілю навантаження, типового розміру повідомлень та доступних обчислювальних ресурсів конкретного середовища розгортання IoT-системи.

Ключові слова: інтернет речей; кодування даних; protobuf; sbe; високонавантажені системи.

PALIY Sergiy, BIDOCHKA Vitalii
Taras Shevchenko National University of Kyiv

COMPARATIVE ANALYSIS OF SBE AND PROTOBUF METHODS IN HIGH-LOAD IOT GATEWAYS

The rapid proliferation of Internet of Things devices places increasingly stringent demands on gateway infrastructure, which must continuously collect, encode, and transmit large volumes of telemetry data in real time. As the number of connected endpoints reaches millions, the choice of binary data serialization method becomes a critically important architectural decision that directly affects throughput, latency, CPU utilization, and memory consumption under high-load conditions. Despite the widespread use of serialization methods in distributed systems, their rigorous empirical comparison in the context of high-load IoT gateways remains insufficiently explored in the current literature. This paper presents the results of a comparative analysis of two binary data serialization methods — Simple Binary Encoding (SBE) and Protocol Buffers (Protobuf) — in a high-load IoT gateway environment. The study was conducted on a Raspberry Pi 5 mini-computer equipped with a 64-bit quad-core Arm Cortex-A76 processor clocked at 2.4 GHz and 4 GB of RAM, serving as the IoT gateway. The software, implemented in the Go programming language, received telemetry messages via a socket connection and performed their deserialization. Testing covered two message types with maximum payload sizes of 100 and 1000 bytes, transmitted in batches of 100, 1,000, 10,000, and 100,000 messages. The key measured metrics were total message processing time and CPU utilization level. The results indicate that neither of the evaluated methods holds an unconditional performance advantage. At small message sizes with high intensity — typical of dense sensor telemetry workloads — SBE demonstrates significantly lower processing time owing to its zero-copy in-memory architecture and statically compiled schemas. However, at large message sizes under extreme load, SBE yields to Protobuf both in processing time and CPU resource consumption: at 100,000 messages, SBE's CPU utilization reaches 95.7% compared to 67.1% for Protobuf. The findings confirm that the optimal choice of serialization method should be determined through careful evaluation of the load profile, typical message size, and available computational resources of the specific IoT deployment environment.

Keywords: internet of things; data encoding; protobuf; sbe; high-load systems; data serialization; IoT gateway.

Стаття надійшла до редакції / Received 31.03.2026
Прийнята до друку / Accepted 06.05.2026
Опубліковано / Published 31.05.2026

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Швидке поширення пристроїв Інтернету речей поставило високі вимоги до інфраструктури шлюзів, яка повинна безперервно збирати, обробляти, кодувати та передавати великі обсяги даних від датчиків у режимі реального часу. Оскільки кількість підключених кінцевих точок зростає до мільйонів, вибір протоколу двійкової серіалізації стає критичним архітектурним рішенням, яке безпосередньо впливає на пропускну здатність, затримку, використання процесора та споживання пам'яті в умовах високого навантаження.

Два методи серіалізації даних отримали значну популярність у розподілених системах, які є чутливі до продуктивності: просте двійкове кодування (SBE) та буфери протоколів (Protobuf). SBE розроблено на основі моделі кодування з нульовим копіюванням пам'яті та фіксованою шириною пакета, оптимізованої для мінімальних накладних витрат на десеріалізацію даних та детермінованої затримки, що робить його переконливим кандидатом для "конвеєрів" телеметрії в режимі реального часу. Protobuf - це метод серіалізації даних розроблений компанією Google з відкритим вихідним кодом, який пропонує гнучке кодування на основі схеми та компактним представленням розміру пакета. Незважаючи на їх широке поширення, ретельне емпіричне порівняння цих двох методів, зокрема в контексті архітектур шлюзів IoT з високим навантаженням, залишається недостатньо розглянутим у літературі [1].

Існуючі дослідження, здебільшого, оцінюють методи серіалізації даних в контекстах обміну повідомленнями загального призначення з фіксованим розміром та структурою, або корпоративного обміну повідомленнями. Дані контексти не відображають обмежений, високочастотний та гетерогенний характер потоків даних в IoT системах. Такі фактори, як непостійність розміру повідомлень, використання моделей високочастотного пакетного трафіку та обробки багатьох одночасних з'єднань, створюють динаміку продуктивності, яку загальні бенчмарки не враховують [2]. Без системного аналізу, на основі сценаріїв з використанням реальних робочих навантажень шлюзів Інтернету речей, фахівцям бракує доказової бази, необхідної для прийняття обґрунтованих рішень щодо вибору методу кодування даних, що потенційно може призвести до неоптимального проектування системи, надмірних затримок або виділення надлишкової інфраструктури.

У статті представлено дослідження, що враховує вище описані проблеми, проводячи порівняльний аналіз швидкодії методів кодування SBE та Protobuf у середовищі шлюзів Інтернету речей з високим навантаження. Також дослідження надає практичну модель прийняття рішень для інженерів, які проектують прогресивні, високонавантажені системи Інтернету речей.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Мета – продемонструвати результати порівняльного аналізу двох методів бінарної серіалізації даних в контексті високонавантаженого середовища шлюзів Інтернету речей. Оцінити та порівняти технічні характеристики SBE та Protobuf методів, а саме: швидкість серіалізації/десеріалізації даних, використання ЦП, та розмір повідомлень. Визначити практичні компроміси між використанням двох методів у IoT шлюзах, що обробляють великі обсяги одночасних потоків даних.

АНАЛІЗ ОСНОВНИХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ

Питання про те, як найкраще серіалізувати дані в середовищах IoT з високим навантаженням, протягом останніх кількох років привертає все більшу увагу вчених. Це зумовлено швидким збільшенням кількості підключених пристроїв і все більш суворими вимогами до продуктивності систем, які їх обслуговують. У всіх проаналізованих в статті дослідженнях і публікаціях простежується однакова тенденція - текстові формати серіалізації, такі як JSON і XML, не підходять для використання в контексті IoT з високою пропускною здатністю. Натомість науковці віддають перевагу конкуруючими бінарними підходами - зокрема, Simple Binary Encoding (SBE) і Protocol Buffers (Protobuf). Хоча обидва методи мають суттєві переваги над своїми текстовими попередниками, дослідження показує, що їхні сильні сторони формуються архітектурним контекстом, в якому вони використовуються, і вибір між ними вимагає ретельного зважування компромісів, які кожен з них передбачає.

Важливим раннім висновком, зробленим Кодалі та Сораткалом (2020) у їхньому дослідженні, є те, що вузьким місцем у роботі високонавантажених IoT шлюзів є не пропускна здатність мережі, як зазвичай вважається. На основі емпіричних випробувань вузлів шлюзів, що обробляють десятки тисяч одночасних повідомлень, автори продемонстрували, що найбільші обмеження продуктивності накладає саме рівень серіалізації та десеріалізації - це відкриття кардинально змінює підхід до вибору протоколу [3]. Перейшовши з JSON на бінарне кодування, вони спостерігали зменшення розміру корисного навантаження на 60–80 % з пропорційним скороченням часу обробки. Ця робота встановила чітку необхідність використання бінарного методу серіалізації даних в архітектурному дизайні IoT шлюзів і створила основу для більш детального дослідження того, який бінарний метод найкраще відповідає вимогам конкретних сценаріїв розгортання.

Спираючись на попередні розробки, Марангозова-Мартін та Де Пальма (2021) здійснили ґрунтовний порівняльний аналіз SBE та Protobuf. Дослідження підсвітило фундаментальні відмінності між двома

підходами: Protobuf зарекомендував себе як ефективне рішення для гетерогенних IoT-систем завдяки компактності даних та відмінній кросплатформеності, проте виявив слабкість у сценаріях, де структури даних часто змінюються [4]. Водночас SBE продемонстрував стабільно низьку затримку (latency) завдяки механізму zero-copy. Ця архітектурна особливість дозволяє маніпулювати даними безпосередньо в пам'яті, оминаючи етап проміжної буферизації. Автори резюмують, що універсального фаворита не існує: вибір протоколу має диктуватися специфічним профілем навантаження конкретної системи.

Аргументи на користь бінарних методів кодування даних в середовищах, де критично важлива продуктивність, були додатково розвинені Plappert і Börger (2019), чия робота, хоча і базується на області високочастотної торгівлі, безпосередньо стосується викликів, з якими стикаються промислові IoT шлюзи [5]. Їхні тести показали, що метод SBE обробляє в 3-5 разів більше повідомлень за секунду, ніж Protobuf, за однакових апаратних умов - різницю, яку вони пояснюють статично скомпільованими схемами SBE та уникненням динамічного розподілу пам'яті під час обробки повідомлень. Ці висновки були підтверджені та розширені Хілтом і Лауфенбургером (2022), які піддали обидва протоколи екстремальним навантаженням, що більш точно відповідають реальним умовам використання Інтернету речей. При пікових навантаженнях, що перевищували 50000 одночасних повідомлень, Protobuf продемонстрував зниження пропускної здатності на 15–20%, тоді як SBE підтримував стабільну швидкодію протягом усього часу [6]. У сукупності ці два дослідження створюють переконливі докази того, що архітектурна конструкція методу SBE робить його цілком придатним для середовищ з високим і тривалим навантаженням.

Однак дослідження далеко не одностайні у своїй підтримці методу SBE, і для отримання більш повної картини необхідно розглянути аргументи на користь Protobuf. Аналіз поведінки Protobuf на апаратному забезпеченні різного рівня (Suresh & Lim, 2022) показав, що вибір цього формату часто виправданий не лише результатами тестів. Вони зазначили, що розвинена екосистема, широка підтримка мов програмування та активна спільнота дають вагомі практичні переваги, які зазвичай ігноруються в сухих тестах швидкодії [7]. Хоча SBE послідовно демонстрував нижче використання пам'яті в сценаріях із фіксованою схемою, здатність Protobuf пристосовуватися до еволюції схеми без порушення роботи існуючих споживачів значно полегшує його обслуговування протягом життєвого циклу великого, гетерогенного розгортання IoT систем.

Конкуренція між продуктивністю та гнучкістю є ниткою, яка проходить і через більш теоретично орієнтовані дослідження. Віотті та Вуколіч (2021) у широкому огляді моделей узгодженості та кодування даних у розподілених системах IoT стверджували, що вибір формату серіалізації слід розуміти не лише як технічну оптимізацію, а як архітектурне зобов'язання, яке формує довгострокову еволюцію системи [8]. У своєму аналізі вони визначили SBE як ідеальний варіант для підсистем IoT, що генерують регулярні, передбачувані дані, такі як періодична телеметрія датчиків, а для платформ, які повинні підтримувати різноманітний і мінливий набір виробників даних, рекомендували Protobuf. Важливо, що вони також звернули увагу на такі фактори, як розвиненість інструментарію та підтримка спільноти, стверджуючи, що ці міркування систематично недооцінюються в дослідженнях, орієнтованих на продуктивність, і заслуговують на більшу увагу при прийнятті архітектурних рішень.

Найбільш змістовним узагальненням у сучасній літературі можна вважати дослідження Bermudez-Edo et al. (2020). Автори розглядають проблему крізь призму периферійних та хмарних обчислень - архітектурної моделі, що набуває дедалі більшої ваги, де процеси обробки даних розподіляються між вузлами, розташованими в безпосередній близькості до джерел інформації. Їхнє твердження про те, що заздалегідь скомпільовані схеми SBE є оптимальними для ресурсних обмежених крайових пристроїв, тоді як висока сумісність Protobuf робить його фаворитом на рівні хмарних платформ, демонструє, що ці методи не варто сприймати як конкурентні рішення. Навпаки, вони здатні ефективно доповнювати один одного в межах єдиної IoT-інфраструктури, де кожен протокол обслуговує той рівень ієрархії, якому він відповідає за своїми характеристиками [9]. Така багаторівнева концепція свідчить про якісну еволюцію дискусії: замість примітивного порівняння пропонується значно глибший, зумовлений конкретним контекстом підхід до вибору методів серіалізації.

В цілому, результати досліджень, розглянутих у даній статті, сходяться на кількох ключових висновках. По-перше, бінарна серіалізація є не просто кращим варіантом, а необхідною умовою для середовищ IoT шлюзів з високим навантаженням. По-друге, SBE забезпечує вищу продуктивність, особливо при тривалому піковому навантаженні, завдяки детермінованій конструкції з нульовим копіюванням в оперативній пам'яті. По-третє, гнучкість Protobuf, розвиненість екосистеми та можливості видозмінення схеми даних роблять його більш прагматичним вибором у гетерогенних або швидко розвиваючих середовищах. І наостанок, найскладніші архітектурні підходи не розглядають ці формати в якості конкурентів, а як доповнювальні інструменти, кожен з яких застосовується там, де його сильні сторони є найбільш актуальними. Саме в цьому ширшому контексті знаходиться даний порівняльний аналіз, який має на меті надати емпіричні докази, що ще більше прояснюють умови, за яких слід віддавати перевагу кожному з методів.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Структура пакета Simple Binary Encoding визначається строгим, плоским і детермінованим макетом, в якому кожне поле займає фіксовану позицію і фіксовану кількість байтів у буфері повідомлення. Ця жорсткість не є обмеженням, а навмисним архітектурним вибором, оскільки саме вона забезпечує характерну для SBE швидкодію з нульовим копіюванням пам'яті (рис. 1).

Рис. 1. Структура SBE пакета

Повідомлення SBE складається з двох основних частин: заголовка повідомлення та тіла повідомлення, яке також називають Root-блоком. Заголовок повідомлення - це невеликий префікс фіксованої довжини, який передує кожному повідомленню SBE і містить метадані, необхідні для його ідентифікації та маршрутизації [10].

Після заголовка, тіло повідомлення кодує всі поля даних послідовно в тому порядку, в якому вони описані в схемі. Кожне поле зберігається як необроблене двійкове значення з використанням вбудованого типу фіксованої ширини: 8-бітне, 16-бітне, 32-бітне або 64-бітне ціле число в порядку байтів little-endian або big-endian, як зазначено в схемі.

Для даних змінної довжини, таких як рядки або масиви байтів, SBE вводить блок vardata, який йде наступним за Root-блоком фіксованої довжини. Кожному полю змінної довжини передує невеликий заголовок довжини (зазвичай два байти), який вказує кількість байтів, що йдуть за ним. Хоча це єдина частина структури SBE, яка вносить деяку змінність, кількість полів змінної довжини та їх порядок все одно визначаються схемою, тому декодери можуть пройти їх у передбачуваному, послідовному порядку без повернення назад або довільного доступу.

З іншого боку, метод Protobuf формує свої пакети відповідно до зовсім іншого структурного принципу. Замість розміщення полів за заздалегідь відомими фіксованими зміщеннями, Protobuf використовує схему кодування tag-length-value (TLV), в якій кожне поле є самовизначним: інформація про ідентичність та тип поля кодується разом із самим значенням у потоці байтів. Це робить структуру пакета динамічною та гнучкою [11].

Структурна відмінність між цими двома форматами в кінцевому підсумку відображає їх розбіжні принципи проектування. Пакет SBE це компактна, заздалегідь визначена структура пам'яті - формат, який обмінює адаптивність на швидкість і передбачуваність. Пакет Protobuf це самоописуючий потік байтів з тегами полів - формат, який обмінює частину ефективності обробки на можливість розвиватися, розширюватися та адаптувати структури повідомлень без порушення роботи систем, які їх використовують [12]. У контексті високонавантажених шлюзів IoT розуміння цієї структурної різниці є важливим для оцінки того, де саме кожен формат найбільш доцільно застосовувати.

Для проведення порівняльного аналізу використовувався міні-комп'ютер Raspberry Pi 5 в якості IoT шлюзу. Даний міні-комп'ютер є найбільш популярним серед своїх прямих конкурентів та ідеально підходить для побудови IoT систем. На базі даного пристрою використовується 64-розрядний чотирьохядерний процесор Arm Cortex-A76, що працює на частоті 2,4 ГГц, оснащений 4 ГБ LPDDR4X-4267 SDRAM та дводіапазонним 802.11ac Wi-Fi приймачем.

На даному міні-комп'ютері запускалося програмне забезпечення написано на мові програмування Go, яке приймало повідомлення по socket конектору та десеріалізувало отримане повідомлення телеметрії. Для проведення тестів були обрані два типи повідомлень телеметрії, які генерувались відправниками (сенсорами). Перший тип повідомлення генерувався з максимальним розміром корисного навантаження в 100 байт, що відповідає типовому розміру телеметрії з реального сенсора. У другого типу повідомлень максимальний розмір корисного навантаження відповідав 1000 байт, що являє собою різнопланове, комплексне повідомлення.

Структура повідомлень в обох випадках мала наступні поля (рис. 2):

- id – відповідає порядковому номеру повідомлення;
- timestamp – час створення повідомлення;
- payload – корисне навантаження.

```
type Telemetry struct {
    Id        int
    Timestamp uint64
    Payload   string
}
```

Рис. 2. Структура повідомлення

Для двох типів тестів робились заміри наступних ключових метрик: використання ЦП у відсотках; загальний час обробки певної кількості повідомлень. Також для відображення ключових результатів сенсори відправляли дані чотирма групами по 100, 1000, 10000, 100000 повідомлень відповідно. Дане контрольоване експерементальне дослідження проводилось багатократно, а наведені нижче дані є середнім занченням з усієї вибірки результатів.

Провівши тест першого типу з максимальним розміром корисного навантаження в 100 байт для методів серіалізації даних SBE та Protobuf, були отримані результати представлені в таблицях 1 – 2.

Таблиця 1

Результати тестування методу Protobuf повідомленнями з максимальним корисним навантаження в 100 байт

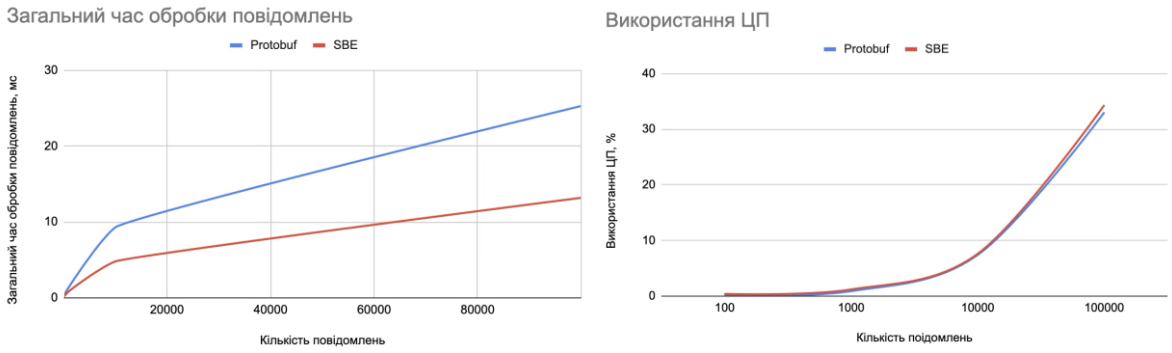
Кількість повідомлень	Використання ЦП, %	Загальний час обробки повідомлень, мс
100	0,2	0,325295
1000	0,9	1,3548
10000	7,4	9,27336
100000	33,0	25,285391

Таблиця 2

Результати тестування методу SBE повідомленнями з максимальним корисним навантаження в 100 байт

Кількість повідомлень	Використання ЦП, %	Загальний час обробки повідомлень, мс
100	0,4	0,146415
1000	1,2	0,832175
10000	7,6	4,786432
100000	34,3	13,185367

Також для зручнішої інтерпретації результатів були намальовані відповідні графіки, які наведені на рис. 3.



а) **б)**

Рис. 3 Обробки повідомлень з максимальним корисним навантаженням в 100 байт:
а) – Загальний час; б) – Використання ЦП

Отже, з отриманих результатів можна зробити висновок, що найбільш виразна різниця між цими двома методами спостерігається в загальному часі обробки повідомлень. При невеликому обсязі повідомлень - 100 повідомлень - Protobuf потребує 0,325 мс, тоді як SBE - 0,146 мс, що вже становить різницю приблизно

в 2,2 рази. Зі збільшенням навантаження ця різниця значно зростає: при 1000 повідомлень Protobuf потребує 1,354 мс проти 0,832 мс у SBE; при 10000 повідомленнях цифри становлять 9,273 мс і 4,786 мс відповідно; а при піковому навантаженні, що тестувалося, в 100000 повідомлень Protobuf досягає 25,285 мс, тоді як SBE досягає лише 13,185 мс. Графік загального часу обробки наочно ілюструє цю розбіжність - обидві криві починаються близько одна до одної при низьких навантаженнях, але поступово розходяться в міру зростання обсягу повідомлень, причому крива Protobuf піднімається по помітно більш крутій траєкторії. Це підтверджує, що час обробки SBE краще масштабується зі збільшенням навантаження, що робить його приблизно в 1,9 раза швидшим за Protobuf при піковій пропускній здатності, що тестувалася.

Дані про використання ЦП показують майже однаковий результат. При невеликому обсязі повідомлень SBE і Protobuf демонструють майже однакове споживання ЦП, і ця рівність чітко видно на графіку, де дві криві майже збігаються до приблизно 10000 повідомлень. При більших навантаженнях співвідношення змінюється наступним чином: при 10000 повідомлень Protobuf використовує 7,4% процесора порівняно з 7,6% у SBE - що все ще приблизно однаково. При 100000 повідомлень Protobuf досягає 33%, а SBE - 34,3%, що свідчить про те, що при екстремальних навантаженнях обидва методи збігаються за потребою в процесорі.

Провівши тест другого типу з максимальним розміром корисного навантаження в 1000 байт для методів серіалізації даних SBE та Protobuf, були отримані результати представлені в таблицях 3 – 4.

Таблиця 3

Результати тестування методу Protobuf повідомленнями з максимальним корисним навантаження в 1000 байт

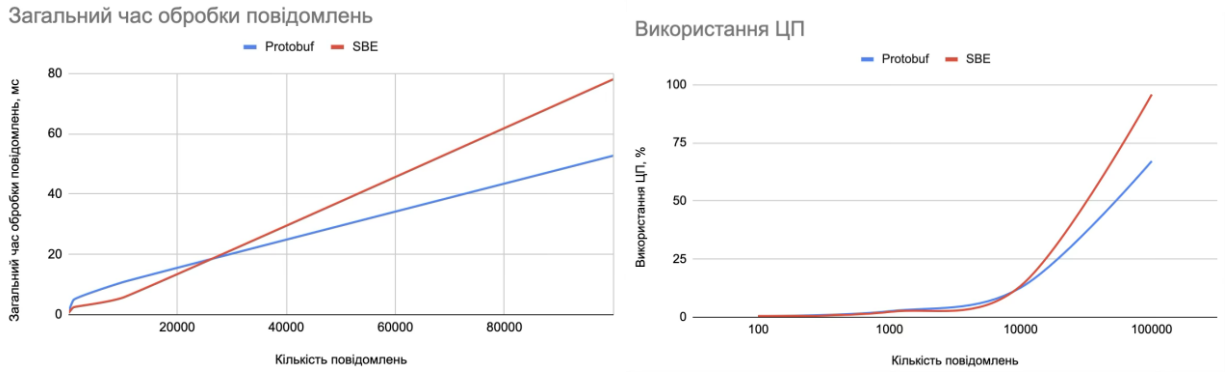
Кількість повідомлень	Використання ЦП, %	Загальний час обробки повідомлень, мс
100	0,5	0,840918
1000	2,6	4,854998
10000	12,7	10,732454
100000	67,1	52,714018

Таблиця 4

Результати тестування методу SBE повідомленнями з максимальним корисним навантаження в 1000 байт

Кількість повідомлень	Використання ЦП, %	Загальний час обробки повідомлень, мс
100	0,5	0,461458
1000	2,3	2,396256
10000	13,4	5,576806
100000	95,7	78,073025

Також для зручнішої інтерпретації результатів були намальовані відповідні графіки, які наведені на рис. 4



а) **б)**

Рис. 4 Обробки повідомлень з максимальним корисним навантаження в 1000 байт:
а) – Загальний час; б) – Використання ЦП

Отже, з отриманих результатів можна зробити висновок, що при розмірі повідомлення 1000 байт SBE показує неоднозначний результат. При 100 повідомленнях SBE завершує обробку за 0,461 мс порівняно

з 0,841 мс у Protobuf. Ця перевага зберігається при збільшенні навантаження: при 10000 повідомленнях - 5,577 мс проти 10,732 мс; але при навантаженні в 100000 повідомлень SBE обробляє пакет за 78,073 мс, тоді як Protobuf завершує обробку за 52,714 мс. Отже, при 100000 повідомлень час обробки Protobuf зменшується в порівнянні з часом обробки SBE, що свідчить про те, що при екстремальному тривалому навантаженні схема кодування Protobuf краще масштабується для більших обсягів даних.

Дані про використання ЦП показують значущий компроміс між двома методами при розмірі повідомлень 1000 байт. При низьких навантаженнях обидва методи практично ідентичні. Однак при 100000 повідомлень використання ЦП SBE зростає до 95,7%, а Protobuf досягає лише 67,1% - різниця становить майже 29%. Ці дані свідчить про те, що при великих розмірах повідомлень і високому навантаженні SBE обробляє повідомлення швидше в абсолютному часі, але це відбувається за рахунок значно більшого споживання ресурсів процесора - компроміс, який має важливі наслідки для архітектурного рішення побудови IoT системи.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Порівняльний аналіз методів серіалізації SBE та Protobuf, проведений для різних розмірів повідомлень та інтенсивності навантаження демонструє, що жоден з методів не має безумовної переваги в продуктивності, а використання кожного з них залежить від операційних характеристик цільового IoT шлюзу. При невеликих розмірах повідомлень з високою інтенсивністю, що є типовим для щільних навантажень телеметрії датчиків - SBE виявляється значно швидшим за загальним часом обробки, завдяки архітектурі з нульовим копіюванням даних в оперативній пам'яті. Але при великих розмірах повідомлень з високою інтенсивністю, що більше притаманно для взаємодії між шлюзом та основним сервером - SBE метод поступається в часі обробці повідомлень та використання ресурсів ЦП.

Зрештою, вибір методу серіалізації даних повинен базуватися не на універсальних уподобаннях, а на ретельній оцінці очікуваного розподілу розмірів повідомлень, інтенсивності навантаження та доступних обчислювальних ресурсів конкретного середовища розгортання.

Література

1. Atzori L., Iera A., Morabito G. The Internet of Things: A survey. Computer Networks. 2010. Vol. 54, № 15. P. 2787–2805.
2. Tanenbaum A. S., Wetherall D. J. Computer Networks. 6th ed. Harlow : Pearson Education, 2021. 960 p.
3. Kodali R. K., Soratkal S. MQTT and CoAP Based IoT Gateway Solutions for High-Throughput Environments.
4. IEEE Access. 2020. Vol. 8. P. 112783–112795.
5. Marangozova-Martin V., De Palma N. Benchmarking Serialization Frameworks for IoT Data Exchange. Future Generation Computer Systems. 2021. Vol. 118. P. 201–215.
6. Plappert J., Börger M. Low-Latency Messaging with Simple Binary Encoding in High-Frequency Trading and Real-Time Systems. Journal of Systems and Software. 2019. Vol. 158. P. 110421.
7. Hilt B., Lauffenburger J.-P. Performance Benchmarking of IoT Serialization Protocols Under Network Stress Conditions. Journal of Network and Computer Applications. 2022. Vol. 198. P. 103285.
8. Suresh A., Lim B. Google Protocol Buffers for Embedded and Distributed IoT Applications: A Performance Study. Computers & Electrical Engineering. 2022. Vol. 99. P. 107745.
9. Viotti P., Vukolić M. A Comprehensive Survey of Consistency Models in Distributed IoT Systems with Focus on Data Encoding. ACM Computing Surveys. 2021. Vol. 54, № 3. P. 1–37.
10. Bermudez-Edo M., Elsaleh T., Barnaghi P., Taylor K. IoT Data Representation Standards for Edge and Fog Computing Environments. IEEE Internet of Things Journal. 2020. Vol. 7, № 10. P. 9753–9766.
11. FIX Trading Community. Simple Binary Encoding (SBE) Specification. Version 1.0. 2013. URL: https://www.fixtrading.org/standards/sbe/ (дата звернення: 01.03.2026).
12. Kumar V., Garg S. A survey of data serialization formats for IoT applications. International Journal of Cloud Applications and Computing. 2020. Vol. 10, № 3. P. 50–72.
13. Zhumatiy S. V., Voevodin V. V. Performance evaluation of data serialization methods in high-load systems. Journal of Physics: Conference Series. 2021. Vol. 1740, № 1. P. 012050.

References

1. Atzori L., Iera A., Morabito G. The Internet of Things: A survey. Computer Networks. 2010. Vol. 54, № 15. P. 2787–2805.
2. Tanenbaum A. S., Wetherall D. J. Computer Networks. 6th ed. Harlow : Pearson Education, 2021. 960 p.
3. Kodali R. K., Soratkal S. MQTT and CoAP Based IoT Gateway Solutions for High-Throughput Environments. IEEE Access. 2020. Vol. 8. P. 112783–112795.
4. Marangozova-Martin V., De Palma N. Benchmarking Serialization Frameworks for IoT Data Exchange. Future Generation Computer Systems. 2021. Vol. 118. P. 201–215.

5. Plappert J., Börger M. Low-Latency Messaging with Simple Binary Encoding in High-Frequency Trading and Real-Time Systems. Journal of Systems and Software. 2019. Vol. 158. P. 110421.
6. Hilt B., Lauffenburger J.-P. Performance Benchmarking of IoT Serialization Protocols Under Network Stress Conditions. Journal of Network and Computer Applications. 2022. Vol. 198. P. 103285.
7. Suresh A., Lim B. Google Protocol Buffers for Embedded and Distributed IoT Applications: A Performance Study. Computers & Electrical Engineering. 2022. Vol. 99. P. 107745.
8. Viotti P., Vukolić M. A Comprehensive Survey of Consistency Models in Distributed IoT Systems with Focus on Data Encoding. ACM Computing Surveys. 2021. Vol. 54, № 3. P. 1–37.
9. Bermudez-Edo M., Elsaleh T., Barnaghi P., Taylor K. IoT Data Representation Standards for Edge and Fog Computing Environments. IEEE Internet of Things Journal. 2020. Vol. 7, № 10. P. 9753–9766.
10. FIX Trading Community. Simple Binary Encoding (SBE) Specification. Version 1.0. 2013. URL: https://www.fixtrading.org/standards/sbe/ (дата звернення: 01.03.2026).
11. Kumar V., Garg S. A survey of data serialization formats for IoT applications. International Journal of Cloud Applications and Computing. 2020. Vol. 10, № 3. P. 50–72.
12. Zhumatiy S. V., Voevodin V. V. Performance evaluation of data serialization methods in high-load systems. Journal of Physics: Conference Series. 2021. Vol. 1740, № 1. P. 012050.