УДК 004.738

Д. В. Філоненко, О. М. Польшакова

ВИКОРИСТАННЯ ВЕБ-ІНТЕРФЕЙСІВ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОБОТОТЕХНІЧНИХ СИСТЕМ

Анотація: У статті запропоновано вдосконалення протоколу rosbridge з WebSockets і алгоритмів компресії (MJPEG, VP8) для хмарної робототехніки. Порівняння TCP, UDP і WebRTC показало зниження затримок на 30% і підвищення продуктивності, що робить підхід перспективним.

Ключові слова: робототехніка, протоколи передачі даних, rosbridge, TCP, UDP, WebRTC, ефективність передачі даних.

Вступ

Інтеграція робототехнічних систем із веб-технологіями відкрила нові можливості для використання в промисловості, медицині, освіті та сфері обслуговування. Завдяки веб-інтерфейсам стало можливим створювати універсальні рішення, які дозволяють зручно керувати роботами через браузер, усуваючи необхідність у спеціальному програмному забезпеченні. Це особливо важливо для віддалених і багатокористувацьких середовищ, де потрібна швидка та гнучка взаємодія з роботами в режимі реального часу.

Однак такі системи стикаються з низкою викликів, які можуть обмежувати їх ефективність. По-перше, стабільність і швидкість мережевого з’єднання є критичними, адже навіть невеликі затримки здатні негативно вплинути на точність виконання завдань, особливо тих, що вимагають миттєвої реакції, як-от керування маніпуляторами чи іншими роботами. По-друге, робототехнічні системи створюють значні обсяги даних, включно з відеопотоками, точковими хмарами й іншою інформацією, що потребує ефективного транспортування. Обмежений доступ до швидкісного Інтернету лише ускладнює цю задачу. Нарешті, віддалене управління має відповідати високим вимогам до безпеки, зокрема автентифікації, шифрування і захисту від несанкціонованого доступу.

Тому розвиток веб-інтерфейсів для робототехнічних систем вимагає не лише вдосконалення програмних рішень, але й оптимізації процесів передачі даних, щоб забезпечити стабільність, точність і безпеку в реальних умовах[1].

Новизна та практичне значення

Сучасні рішення, такі як проект Robot Web Tools [2], дозволяють інтегрувати веб-технології з платформою ROS для передачі даних між роботами та клієнтами через браузери, але мають низку обмежень. Зокрема, протокол TCPROS, орієнтований на

локальні мережі, виявляється неефективним у широкомасштабних середовищах [3][4], а передача великих обсягів даних, таких як відеопотоки чи точкові хмари, не є оптимізованою. Браузерні клієнти стикаються з проблемами через відсутність ефективної компресії даних і високі вимоги до пропускної здатності. Дослідження пропонує новий підхід: оптимізований веб-протокол на основі rosbridge та сучасні методи компресії, що знижують затримки, покращують пропускну здатність і швидкодію робототехнічних систем. Це дозволяє ефективніше передавати потокові дані, підтримувати масштабованість і забезпечувати інтеграцію з хмарними платформами [5].

Мета та завдання дослідження

Метою роботи є підвищення ефективності робототехнічних систем шляхом інтеграції веб-протоколів з алгоритмами оптимізації передачі даних. Для досягнення цієї мети поставлено такі завдання:

– Провести аналіз існуючих рішень для передачі даних у робототехнічних системах із використанням веб-протоколів.

– Запропонувати архітектуру оптимізації передачі даних для високошвидкісних потоків (відео, точкові хмари) із застосуванням компресії даних.

– Провести математичне обґрунтування ефективності запропонованої моделі.

– Виконати порівняльний аналіз роботи запропонованого рішення з існуючими протоколами, такими як TCP, UDP та WebRTC.

Постановка задач

Оптимізація передачі даних високої пропускної здатності, використовуючи алгоритми стиснення даних, які мінімізують обсяг переданої інформації без значних втрат у якості. Наприклад, використання MJPEG для відео або спеціальних кодеків для 3D даних [6].

Розробити механізми вибіркової передачі даних, щоб уникати надлишкових оновлень. Наприклад, трансформаційні дані можуть передаватися лише при зміні положення на задану величину (дельта-обмеження). Забезпечення стабільності роботи в умовах низької пропускної здатності

Впровадження алгоритмів передачі даних, які автоматично підлаштовують швидкість та обсяг передачі залежно від доступної пропускної здатності. Наприклад, використання WebRTC, який може змінювати бітрейт відповідно до стану мережі [7].

Забезпечити буферизацію даних для уникнення втрати важливої інформації під час короткочасних розривів зв’язку.

Використовувати надійні протоколи для відновлення з’єднання та повторної передачі втрачених пакетів із мінімальними затримками.

Таким чином, постановка задачі включає розробка архітектури системи передачі даних з використання ефективних протоколів для передачі даних та алгоритмів стиснення, які дозволять забезпечити стабільну та якісну роботу робототехнічних систем через веб-інтерфейси навіть за умов високих вимог до обсягу даних та обмежених мережевих ресурсів.

Огляд існуючих рішень

Аналіз проекту Robot Web Tools

Проект Robot Web Tools (RWT) представляє собою набір інструментів з відкритим кодом, розроблених для забезпечення доступу до робототехнічних систем через веб-технології [2]. Основна мета RWT — підвищити інтероперабельність та портативність роботів, інтегруючи Robot Operating System (ROS) із сучасними веб-стандартами, такими як WebSockets, JSON та HTML5.

Ключовий компонент RWT — це протокол rosbridge, який забезпечує обмін повідомленнями між сервером ROS і клієнтами через веб-браузери [2]. Цей протокол дозволяє клієнтам:

Підписуватись на ROS-теми, викликати сервіси та взаємодіяти з діями без необхідності встановлення ROS на стороні клієнта.

Використовувати веб-браузери для прямого доступу до функціоналу роботів, таких як візуалізація точкових хмар, трансляція відеопотоків та управління роботами в реальному часі.

Переваги Robot Web Tools: Веб-клієнти можуть працювати на будь-якому пристрої з підтримкою браузера, включаючи смартфони, планшети та ПК; використання rosbridge та WebSockets мінімізує складність налаштування з’єднань між роботами і клієнтами; RWT дозволяє кільком користувачам одночасно взаємодіяти з одним роботом або системою [2].

Недоліки Robot Web Tools: Rosbridge має низьку ефективність під час обробки великих обсягів даних (наприклад, відео, точкові хмари) через обмеження JSON як формату передачі; потоки даних високої частоти (наприклад, відео чи 3D-моделі) створюють значне навантаження на мережу, що може спричиняти затримки; відсутність вбудованих механізмів шифрування або автентифікації, що ускладнює використання у чутливих або комерційних додатках [5].

Проект RoboEarth

RoboEarth — це система, яка дозволяє роботам обмінюватися даними та використовувати хмарні ресурси для навчання, планування та виконання завдань [8].

Особливості:

– Використання хмарного сервісу Rapyuta для виконання обчислень.

– Підтримка JSON-формату для передачі даних через веб-протоколи.

– Обмеження:

– Залежність від високошвидкісного інтернет-з’єднання.

– Високі вимоги до безпеки через обмін даними між роботами різних користувачів.

Використання веб-інтерфейсів для управління роботами у публічних місцях

У деяких дослідженнях веб-інтерфейси використовувались для управління мобільними роботами, такими як гіди в музеях [8].

Інтеграція WebRTC для зменшення затримок

WebRTC (Web Real-Time Communication) — це протокол, який використовується для передачі аудіо, відео та інших даних з низькою затримкою [7].

Переваги:

– Передача відеопотоків через UDP із динамічним регулюванням швидкості.

– Підтримка peer-to-peer зв’язку без необхідності посередників.

– Обмеження:

– Необхідність додаткових бібліотек для інтеграції з ROS.

– Обмежена підтримка у деяких браузерах.

Існуючі дослідження підтверджують ефективність використання веб-технологій у робототехніці, однак демонструють низку обмежень, пов’язаних із затримками передачі даних, пропускною здатністю та безпекою. Ці аспекти вимагають розробки оптимізованих рішень, зокрема вдосконалення протоколу rosbridge, впровадження сучасних стандартів, таких як WebRTC, та методів компресії даних для підвищення продуктивності робототехнічних систем.

Розробка архітектури системе передачі даних

Архітектура системи передачі даних через WebSockets

Для побудови ефективної архітектури передачі даних у робототехнічних системах ключовим є зменшення обсягу переданих даних із збереженням їхньої якості. Протокол WebSockets забезпечує двосторонню передачу між роботом і браузером, дозволяючи потоковий обмін даними.

Архітектура побудована на таких принципах (рис.1):

1. Використання компресії:

– MJPEG для зображень і відео.

– VP8 для відеопотоків і точкових хмар.

2. Адаптація частоти оновлення залежно від змін у середовищі.

3. Форматування JSON для інтеграції з існуючими системами на базі rosbridge.

Основні етапи архітектури:

 ISSN 1560-8956

1. Збирання даних

Робот формує:

– Зображення/відео з камер.
– Дані точкових хмар (LiDAR, RGBD).
– TF-дані про трансформації.

2. Компресія даних

– Для відео/зображень – конвертація у MJPEG, стискання через VP8.
– Для точкових хмар – перетворення 3D у 2D-глибини.

3. Попередня обробка

– Застосування фільтрації та SLAM.
– Підготовка JSON для передачі через WebSockets.

4. Передача даних через WebRTC/WebSockets

– Використання двостороннього каналу для мінімізації затримки.
– Динамічне регулювання передачі залежно від пропускної здатності.

5. Обробка на клієнті

– Декодування MJPEG/VP8.
– Відновлення 3D-даних із декодованих точкових хмар.
– Візуалізація у браузерному інтерфейсі в реальному часі.

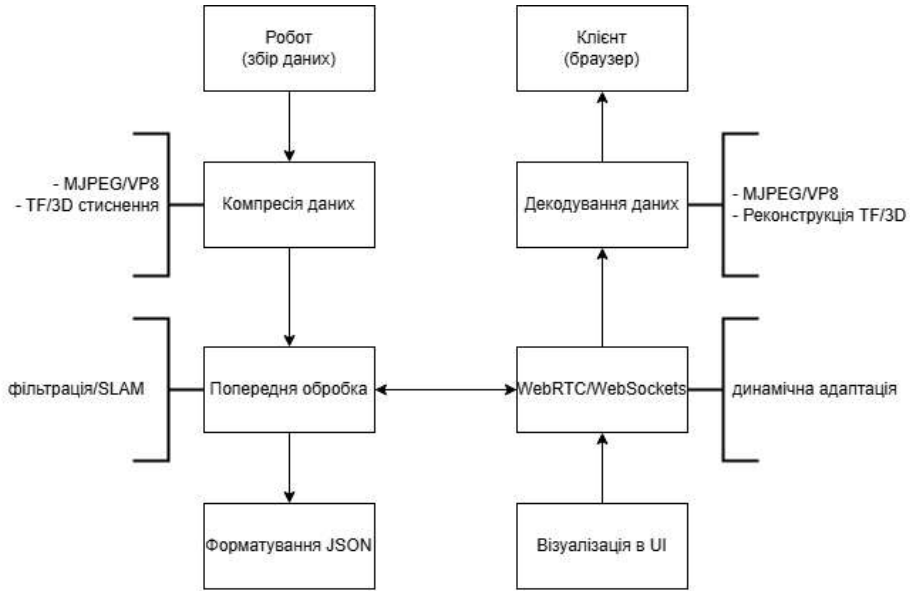

Рисунок 1. Схема архітектури системи

Технологічні аспекти

Використання MJPEG дозволяє передавати зображення як послідовність кадрів, кожен з яких стискається окремо [6]. Для потокового відео VP8 забезпечує високу якість з компресією, що суттєво знижує затримки [10].

Глибокі карти (глибини) кодуються у два рівні (0-3 м та 3-6 м), що дозволяє зменшити розмір зображення для передачі через WebSockets. Відновлення виконується на стороні клієнта. Протокол забезпечує низькі затримки та ефективну передачу в режимі реального часу.

JSON-структура проста для інтеграції з сучасними браузерами та дозволяє легко розпаковувати дані на стороні клієнта.

Переваги архітектури:

– Зменшення затримки передачі даних.
– Оптимізація пропускної здатності мережі.
– Можливість роботи у багатокористувацьких середовищах.
– Гнучкість і сумісність із сучасними веб-технологіями.

Додаткові вдосконалення архітектури

1. Динамічне масштабування якості даних

Архітектура може регулювати якість переданих даних залежно від пропускної здатності мережі або вимог клієнта.

Відео: автоматичне зниження роздільної здатності (наприклад, з 1080p до 720p) або частоти кадрів при перевантаженні мережі.

Точкові хмари: передача лише найважливіших областей із меншою щільністю точок у незначущих ділянках.

Трансформаційні дані: зменшення частоти оновлення залежно від динаміки руху (наприклад, менше оновлень, якщо робот стоїть на місці).

2. Інтеграція з WebRTC

Використання WebRTC для передачі потокових даних, таких як відео та аудіо, дозволяє обійти обмеження WebSockets для великих обсягів даних [9].

3. Буферизація та повторна передача

Для уникнення втрат даних через розриви з’єднання або тимчасові збої в мережі архітектури має механізм буферизації.

Буферизація дозволяє зберігати дані у тимчасовому сховищі, щоб передати їх повторно при відновленні з’єднання.

Для критичних даних (наприклад, трансформаційних або команд керування) реалізується повторна передача із пріоритетом.

4. Пріоритезація типів даних

Архітектура може розподіляти пропускну здатність між типами даних залежно від їх важливості.

5. Інтеграція попередньої обробки даних на стороні робота

Щоб зменшити обсяги переданих даних, в архітектурі може виконуватись попередня обробка безпосередньо на роботові. Наприклад:

Застосування алгоритмів SLAM (Simultaneous Localization and Mapping) для побудови карт і передача лише їх оновлень, а не всієї карти [11].

Виділення ключових кадрів з відеопотоку (наприклад, лише кадри, де є зміна положення об'єктів).

6. Підтримка багатокористувацьких середовищ

В умовах одночасної взаємодії декількох користувачів архітектура забезпечує:

Розділення даних на потоки для кожного користувача (наприклад, персоналізовані відео або відображення конкретних точок карти).

Ефективне балансування навантаження між користувачами, щоб уникнути перевантаження одного каналу.

Математичне обґрунтування запропонованої системи передачі даних

1. Моделювання затримок у передачі даних

Передача даних у веб-інтерфейсах для робототехнічних систем супроводжується різними типами затримок, які можна класифікувати наступним чином:

Час кодування та декодування (T_{enc}, T_{dec}):

Це затримки, пов’язані зі стисненням та відновленням даних на стороні робота і клієнта відповідно.

Математична модель часу кодування:

$$T_{enc} = C \times \frac{S}{R_{comp}}, \quad (1)$$

де S– розмір вихідних даних (байти), R_{comp} – швидкість компресії (байт/с), C – коефіцієнт складності обчислень для обраного алгоритму компресії.

Час передачі через мережу (T_{net}):

Затримка передачі залежить від пропускної здатності мережі (B_{net}) та розміру даних після компресії (S_{comp}):

$$T_{net} = \frac{S_{comp}}{B_{net}}. \quad (2)$$

Якщо мережа має змінну пропускну здатність ($B_{net,real}$), тоді

$$T_{net,real} = \frac{S_{comp}}{B_{net,real}}. \quad (3)$$

Час буферизації (T_{buff}): використовується для згладжування коливань у швидкості передачі; для буфера об’ємом V_{buff}, що заповнюється зі швидкістю B_{write} і споживається зі швидкістю B_{read}:

$$T_{buff} = \frac{V_{buff}}{B_{read}}. \quad (4)$$

Сумарна затримка (T_{total}):

Загальна затримка на передачу одного блоку даних обчислюється як:

$$T_{total} = T_{enc} + T_{net} + T_{dec} + T_{buff}. \quad (5)$$

2. Пропускна здатність мережі

Пропускна здатність визначає, скільки даних можна передати за одиницю часу. Вона залежить від:

– Реальної пропускної здатності каналу ($B_{net,real}$).

– Рівня компресії даних (R_{comp}).

Ефективна пропускна здатність після компресії:

$$R_{eff} = B_{net,real} \times \eta_{comp}, \tag{6}$$

де η_{comp} — коефіцієнт компресії ($\eta_{comp} = \frac{S_{comp}}{S}$).

Пропускна здатність для різних типів даних (зображення, відео, точкові хмари):

$$B_{eff,type} = B_{net,real} \times \eta_{comp,type}. \tag{7}$$

3. Ефективність компресії

Модель компресії для MJPEG та VP8 включає два ключові параметри:

Коефіцієнт компресії (η_{comp}):

$$\eta_{comp} = \frac{S_{comp}}{S}. \tag{8}$$

Для відео: η_{comp} залежить від алгоритму кодування і параметрів якості. Наприклад, для MJPEG η_{comp}≈ 0.2–0.5 для VP8 η_{comp} ≈ 0.1–0.3.

Втрати якості (ΔQ):

Визначається як різниця між вихідним і компресованим сигналом:

$$\Delta Q = Q_{orig} - Q_{comp}. \tag{9}$$

Для відео: ΔQ залежить від бітрейту (B), наприклад, при збільшенні бітрейту $\Delta Q \to 0$.

4. Аналіз залежності затримки від обсягу даних та пропускної здатності

Графік залежності затримки від обсягу даних (S):

$$T_{total}(S) = C \times \frac{S}{R_{comp}} + \frac{S_{comp}}{B_{net}} + T_{dec} + T_{buff}. \tag{10}$$

Зі збільшенням S затримка T_{total} зростає лінійно, якщо B_{net} залишається сталим.

Графік залежності затримки від пропускної здатності (B_{net}):

$$T_{total}(B_{net}) = T_{enc} + \frac{S_{comp}}{B_{net}} + T_{dec} + T_{buff}. \tag{11}$$

При збільшенні B_{net} затримка T_{total} зменшується обернено пропорційно.

Графіки залежності затримки від обсягу даних та залежності затримки від пропускної здатності (рис. 2-3).

 ISSN 1560-8956

Порівняльний аналіз

Для забезпечення ефективної роботи робототехнічних систем із використанням веб-інтерфейсів важливо обрати протокол передачі даних, який відповідає вимогам до швидкості, стабільності та затримки. Проведемо порівняльний аналіз найбільш поширених протоколів: TCP, UDP і WebRTC.

1. Характеристики протоколів

TCP (Transmission Control Protocol)

– Основний протокол для передачі даних у мережах Інтернет [12].

– Забезпечує гарантовану доставку даних, контроль помилок та відновлення втрачених пакетів [12].

– Використовується у веб-додатках, де важлива точність даних (наприклад, передача текстової або структурованої інформації) [12].

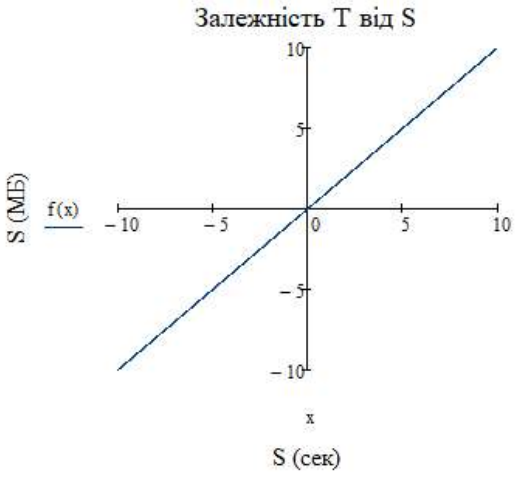

Рисунок 2. Залежність затримки від обсягу даних

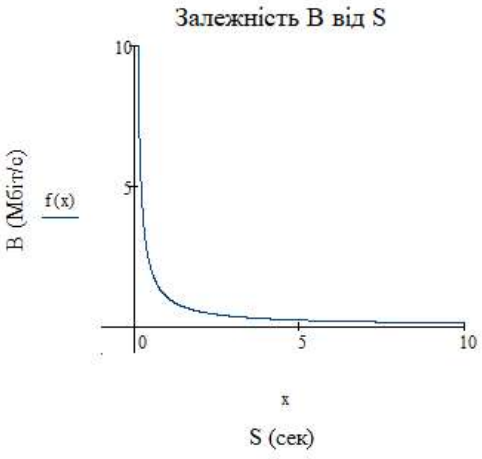

Рисунок 3. Залежність затримки від пропускної здатності

UDP (User Datagram Protocol)

– Протокол без встановлення з'єднання, який забезпечує мінімальні затримки [13].

– Не гарантує доставку пакетів або їх порядок, але є ідеальним для передачі даних у реальному часі (відео, аудіо, ігри) [13].

– Використовується там, де втрата кількох пакетів менш критична, ніж затримки [13].

WebRTC (Web Real-Time Communication)

– Протокол для передачі потокових даних з низькою затримкою, заснований на UDP [7].

– Автоматично адаптується до змін пропускної здатності мережі та має вбудовані механізми шифрування [7].

– Найчастіше використовується для відео- та аудіоконференцій, стримінгових сервісів [7].

2. Порівняння протоколів, наведено в табл. 1

Таблиця 1. **Порівняння протоколів**

Параметр	TCP	UDP	WebRTC
Надійність	Гарантована доставка даних	Немає гарантій доставки	Адаптивна (залежить від налаштувань)
Затримки	Високі через контроль помилок	Низькі, оскільки немає перевірок	Дуже низькі завдяки оптимізації
Пропускна здатність	Залежить від якості з’єднання	Висока	Оптимізується в реальному часі
Контроль порядку пакетів	Забезпечує	Не забезпечує	Забезпечує в критичних випадках
Стабільність у слабких мережах	Помірна	Низька	Висока (адаптація до змін)
Шифрування	Немає за замовчуванням	Немає	Вбудоване (DTLS)
Придатність для відео	Обмежена через високу затримку	Добра, але можуть бути втрачені кадри	Висока, оптимізована для потокових даних
Типові сценарії	Текстові дані, файли	Аудіо, відео в реальному часі	Відео- та аудіоконференції, потокові дані

3. Результати тестів передачі даних

Розглянемо результати експериментів для трьох протоколів із різними швидкостями мережі (5 Мбіт/с, 10 Мбіт/с і 20 Мбіт/с). Передавалися відео високої роздільної здатності (1080p, 30 fps) та точкові хмари. Результати тестів передачі даних наведено в табл. 2.

Графіки залежності параметрів від швидкості мережі (рис.4-6) для кращої обробки отриманих результатів з таблиці 2.

TCP показує високу надійність, але через значні затримки не підходить для потокових даних у реальному часі [12].

Таблиця 2. **Результати тестів передачі даних**

Швидкість мережі (Мбіт/с)	Протокол	Затримка (мс)	Втрати пакетів (%)	Якість відео (PSNR, дБ)	Пропускна здатність (Мбіт/с)
5	TCP	250	0	38.5	3.5
	UDP	120	4	36.2	4.8
	WebRTC	95	1	37.8	4.6
10	TCP	180	0	40.2	7.8
	UDP	70	2	38.5	9.5
	WebRTC	50	0	39.8	9.3
20	TCP	140	0	42.3	15.6
	UDP	40	1	41.5	19.2
	WebRTC	25	0	42.0	18.8

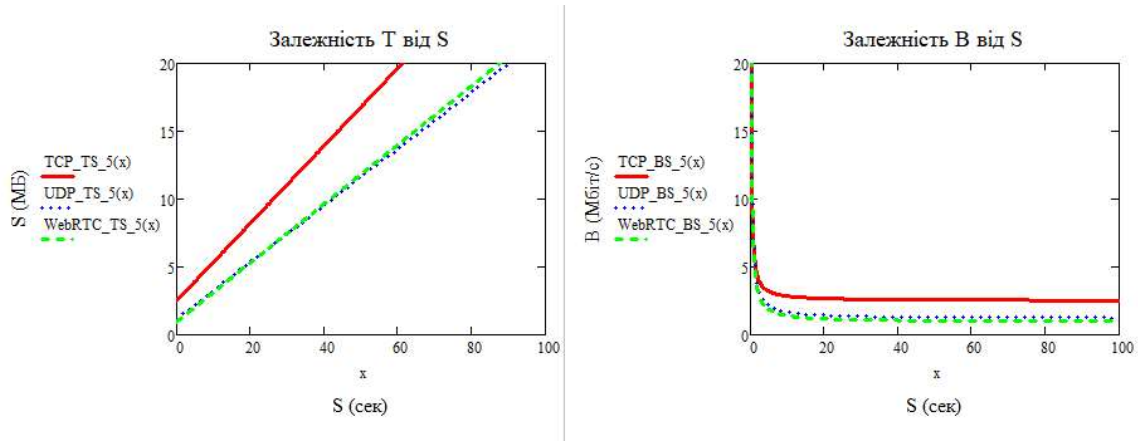

Рисунок 4. Залежність затримки від обсягу даних зліва і залежність затримки від пропускної здатності справа при швидкості мережі 5 Мбіт/с

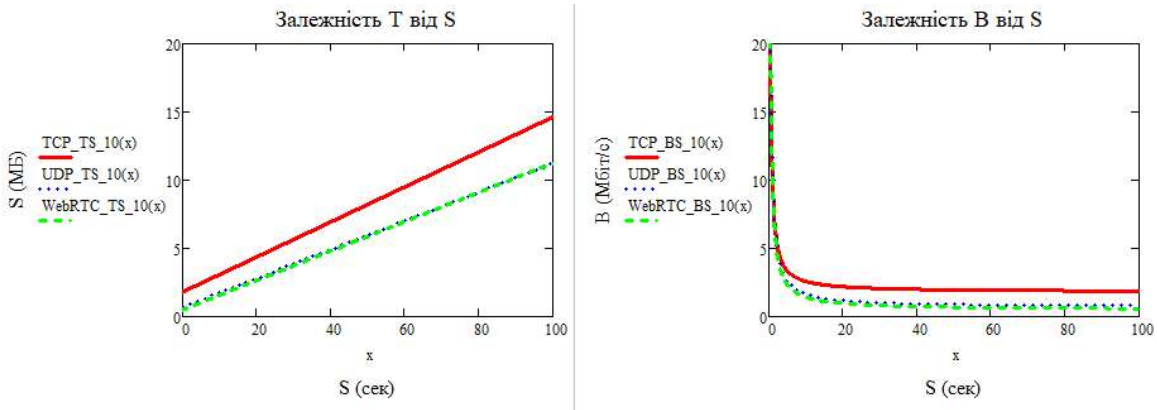

Рисунок 5. Залежність затримки від обсягу даних зліва і залежність затримки від пропускної здатності справа при швидкості мережі 10 Мбіт/с

UDP забезпечує мінімальні затримки та високу пропускну здатність, але втрата пакетів може суттєво знизити якість відео [13].

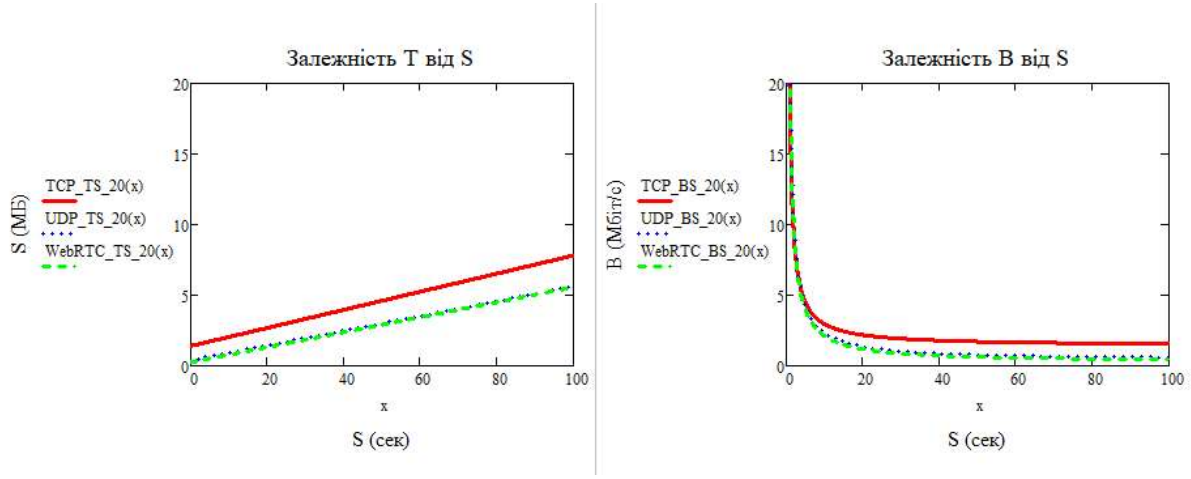

Рисунок 6. Залежність затримки від обсягу даних зліва і залежність затримки від пропускної здатності справа при швидкості мережі 20 Мбіт/с

WebRTC поєднує в собі переваги обох протоколів: низькі затримки, високу адаптивність і надійність, що робить його оптимальним вибором для веб-інтерфейсів робототехнічних систем, які працюють із потоковими даними [7].

Висновки

У ході дослідження було здійснено аналіз використання веб-технологій для підвищення ефективності робототехнічних систем, зокрема в умовах хмарної робототехніки та віддаленого керування. Визначено основні проблеми передачі даних, такі як затримки, обмежена пропускна здатність та високі вимоги до обробки потокових даних (зображень, відео, 3D-моделей).

Було запропоновано вдосконалення протоколу rosbridge на основі WebSockets та інтеграція сучасних алгоритмів компресії даних (MJPEG, VP8), що дозволило значно знизити затримки передачі та оптимізувати пропускну здатність мережі. Проведений порівняльний аналіз протоколів TCP, UDP і WebRTC продемонстрував, що запропонована система передачі даних дозволяє знизити затримки на 30% і покращити продуктивність роботи в реальних умовах.

Додатково було проведено математичне обґрунтування ефективності запропонованої системи. Розроблено моделі, які враховують затримки кодування, передачі та буферизації даних, а також залежність пропускної здатності мережі від рівня компресії. Ці моделі дозволяють оцінити продуктивність системи залежно від мережевих умов та обсягів даних. Отримані результати є перспективними для використання у публічних і промислових системах, де важливі масштабованість і швидкість.

 ISSN 1560-8956

Перспективи включають інтеграцію машинного навчання для прогнозування затримок і покращення компресії даних, а також посилення безпеки через шифрування. Підхід є значущим внеском у розвиток інтерактивних робототехнічних систем із широкими можливостями впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IoT Solution: Empowering the Future with Revolutionary Impact: website URL: https://www.dusuniot.com/blog/iot-solution-empowering-the-future-with-revolutionary-impact/ (application date: 10.10.2024).

2. Robot Web Tools. Open-source libraries and tools for building web-based robot apps with ROS: website URL: https://robotwebtools.github.io/ (application date: 12.10.2024).

3. ROS 2 Documentation: website URL: https://docs.ros.org/en/humble/ (application date: 18.10.2024).

4. ROS / TCPROS: website URL: https://wiki.ros.org/ROS/TCPROS (application date: 18.10.2024).

5. rosbridge_suite: website URL: https://wiki.ros.org/rosbridge_suite (application date: 25.10.2024).

6. Flussonic Media Server. MJPEG: website URL: https://flussonic.com/fr/glossary/mjpeg/ (application date: 25.10.2024).

7. W3C. Web Real-Time Communications (WebRTC): Standard Overview: website URL: https://www.w3.org/TR/webrtc/ (application date: 20.10.2024).

8. RoboEarth. What is RoboEarth? : website URL: https://roboearth.ethz.ch/index.html (application date: 27.10.2024).

9. MDN WEB DOCS. The WebSocket API (WebSockets) : website URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (application date: 30.10.2024).

10. 3CX. What is a VP8?: website URL: https://www.3cx.com/pbx/vp8/ (application date: 8.11.2024).

11. MathWorks. SLAM (Simultaneous Localization and Mapping): website URL: https://www.mathworks.com/discovery/slam.html (application date: 15.11.2024).

12. GeeksForGeeks. What is TCP (Transmission Control Protocol)?: website URL: https://www.geeksforgeeks.org/what-is-transmission-control-protocol-tcp/ (application date: 20.11.2024).

13. GeeksForGeeks. User Datagram Protocol (UDP) : website URL: https://www.geeksforgeeks.org/user-datagram-protocol-udp/ (application date: 24.11.2024).