УДК 316.324

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ ТЕХНОЛОГІЇ ВЕБ-СОКЕТІВ ДЛЯ АСИНХРОННОЇ КЛІЄНТ-СЕРВЕРНОЇ ВЗАЄМОДІЇ ВЕБ-ПРОГРАМ ПРОМИСЛОВОЇ АВТОМАТИЗЦІЇ

Шило Н. Ю., Сидоренко А. В., Буць Д. Є.
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: nazar.shylo@nure.ua, dmytro.buts@nure.ua

Анотація: Проаналізовано сучасні підходи до реалізації асинхронної клієнт-серверної взаємодії веб-програм. Розглянуто ефективність та недоліки основних реалізацій. Для основних алгоритмів наведено принципи їх функціонування. Запропоновано підхід до застосування технології веб-сокетів для взаємодії в реальному часі.

Ключові слова: веб-програма, AJAX, асинхронна взаємодія, технологія опитування, черга запитів очікування, веб-сокети.

FEATURES OF APPLICATION OF WEB SOCKET TECHNOLOGY FOR ASYNCHRONOUS CLIENT-SERVER INTERACTION OF INDUSTRIAL AUTOMATION WEB-PROGRAMS

N. Shylo, A. Sydorenko, D. Buts
Kharkiv National University of Radioelectronics
Ukraine, 61166, Kharkiv, Nauky av., 14
E-mail: nazar.shylo@nure.ua, dmytro.buts@nure.ua

Abstract: This paper analyzed existed approaches of implementation an asynchronous clientserver interaction between/for application. There were considered efficiency and disadvantages of main algorithms and described the principles of their functioning. For web socket technology is described approach for real-time interaction.

Key words: web application, AJAX, asynchronous interaction, Pull technology, Push technology, Long pooling, WebSockets.

АКТУАЛЬНІСТЬ РОБОТИ. Протокол WebSocket («веб-сокет»), описаний у специфікації RFC 6455, забезпечує можливість обміну даними між браузером та сервером через постійне з'єднання. Дані передаються в обох напрямках у вигляді «пакетів», без розриву з'єднання і додаткових HTTP-запитів.

WebSocket особливо хороший для сервісів, які потребують постійного обміну даними, наприклад онлайн ігри, торгові майданчики, що працюють у реальному часі, і т.д.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. Дана стаття описує особливості роботи сучасних веб-застосунків, обмін даними між користувачами в режимі реального часу за допомогою технології WebSocket.

Вебсокети (WebSockets) це просунута і відносно нова технологія, що дозволяє відкрити постійне двонаправлене мережеве з'єднання між браузером користувача та сервером. За допомогою його API ви можливо відправити повідомлення на сервер і отримати відповідь без виконання HTTP запиту, причому цей процес буде подієво-керованим. Для доведення унікальності вебсокетів – розглянемо ситуацію.

Припустимо, що необхідно написати додаток для розсилки. По-перше, треба зробити «клієнт», який буде перевіряти, чи є нові повідомлення кожну хвилину. Проте, більшу частину часу не було ніяких нових повідомлень, а «клієнт» щохвилини посилатиме нові запити, викликаючи величезне навантаження на сервер. Цей метод був дуже популярний, і називався Polling. Ним і зараз користуються, навіть великі компанії.

Логічно, що єдиним правильним рішенням буде зробити так, що сервер відсилав повідомлення на «клієнт» наскільки швидко, наскільки це можливо. Тобто «клієнт 2» не має ініціалізувати запит, цим має займатися сервер. Це було неможливо впродовж довгого часу, але як тільки була представлена технологія вебсокетів, це, нарешті, стало можливим [1].

Розробникові здається, що єдиним рішенням є направлення повідомлення «клієнта», як тільки пошта є доступною. «Клієнт» не повинен ініціювати запит, а сервер робить це. Це було неможливо протягом тривалого часу, але з винаходом WebSockets це стало майже ідеальним рішенням.

WebSockets є протоколом і API JavaScript, протокол є дуже низькорівневим, повністю двостороннім протоколом, який означає, що повідомлення можуть бути відправлені в обох напрямках одночасно. Це дало можливість серверу відсилати дані на «клієнт», замість того, щоб робити протилежне. Polling і Long Polling стали менш затребуваними.

Оскільки WebSockets забезпечують спосіб спілкування в обох напрямках, то вони часто використовуються для додатків реального часу. Якщо, наприклад, для відбувається стороння зміна даних, то є можливість безпосередньо оновити візуалізовані дані для всіх користувачів за допомогою WebSockets [2].

Як тільки сторінка сайту відчиняє WebSocket на сервер – створюється спеціальний Javascript-об'єкт. Все відбувається так само як в звичайному HTTP-запиті.

Браузер підключається по протоколу TCP на 80-й порт сервера, але дає незвичайний GET-запит, і, якщо сервер підтримує WebSocket – відповідає. Якщо браузер це «влаштовує», то залишається TCP-з'єднання відкритим. Все – встановлення зв'язку відлбулося, канал обміну даними готовий [3].

«Один або кілька байтів» – досить незвичний спосіб вказівки довжини «тіла» повідомлення [1]. Щоб не створювати обмежень на довжину переданого повідомлення і в той же час не витрачати байти нераціонально, розробники протоколу використали нижче наведений алгоритм.

Коефіцієнт корисної дії такого протоколу наближається до 95% [2]. Різниця буде особливо помітною, якщо виконується частий обмін невеликих блоків даних. Швидкість обробки також наближається до швидкості чистого TCP-сокета [4].

Але, необхідно відмітити як недоліки, так і переваги. Швидкість і ефективність передачі забезпечує малий розмір переданих даних, який, іноді навіть буде поміщатися в один TCP-пакет, що залежить від логіки розробника [3]. При необхідності враховувати, що з'єднання готове і не потрібно витрачати час і трафік на його встановлення.

Комплексні веб-додатки – в HTTP передбачено обмеження на число одночасних відкритих сесій до одного сервера[2]. Тому для безлічі різних асинхронних блоків на сторінці доводиться робити не тільки серверний, а й клієнтський мультиплексор. Це обмеження не поширюється на протокол WebSocket – відкривається стільки, скільки необхідно.

Ще одна особливість: в якості єдиного дозволеного кодування обрана UTF-8.

Якщо приділити увагу недолікам, то необхідно виділити нижче наведене. В одному з проектів, що переводяться на цей протокол, з'ясувалося, що всі користувачі, які використовувли антивірус Avast в стандартній конфігурації і не могли коректно працювати з додатком. У Avast за замовчуванням включений режим так званого «Веб-екрану», і прийняв рішення, що WebSocket протокол є некоректним і , як наслідок, «різав» його.

Значення WebSocket для розробників та архітекторів:

1. Незалежний протокол на основі TCP, але він призначений для підтримки будь-якого іншого протоколу, який традиційно, працює лише над TCP-з'єднанні.

2. Транспортний рівень, з яким може працювати будь-який інший протокол. WebSocket API підтримує можливість визначення суб-протоколів: бібліотек протоколів, з можливістю ь інтерпретування певних протоколів.

3. Приклади таких протоколів включають XMPP, STOMP та AMQP. Розробникам більше не потрібно думати з погляду парадигми HTTP запит-відповідь.

4. Єдина вимога на стороні браузера – це запуск бібліотеки JavaScript, яка може інтерпретувати з’єднання WebSocket, встановлювати та підтримувати з'єднання WebSocket.

5. На стороні сервера промисловим стандартом є використання існуючих бібліотек протоколів, що працюють над TCP, та використання шлюзу WebSocket.

На рис. 1 наведено функціональні можливості WebSocket.

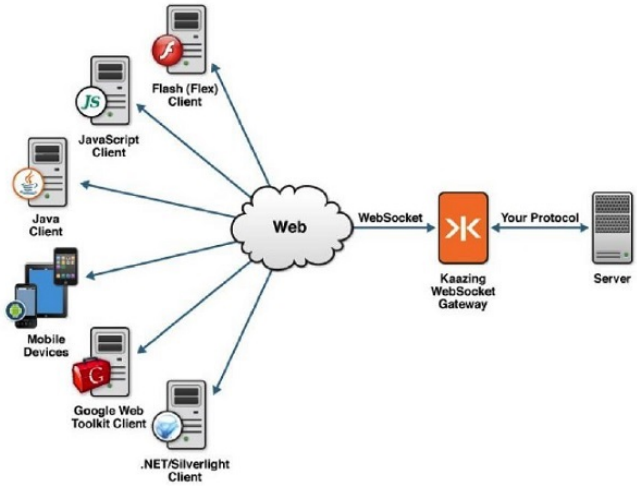

Рисунок 1 – Функціональні можливості WebSocket

З'єднання через Web Socket ініціюються через HTTP; HTTP-сервери, зазвичай, інтерпретують «рукостискання» Web Socket як запит на оновлення.

Веб-сокети забезпечують з'єднання між веб-сервером і клієнтом таким чином, що обидві сторони можуть розпочати відправлення даних.

Кроки для встановлення з'єднання WebSocket такі:

1. Клієнт встановлює з'єднання через процес, відомий як рукостискання Web Socket.

2. Процес починається з того, що клієнт надсилає на сервер звичайний HTTP-запит.

3. Потрібне оновлення заголовка. У цьому запиті він «повідомляє» серверу, що запит на підключення до веб-сокету.

URL-адреси веб-сокетів використовують схему ws. Вони також використовуються для безпечних з'єднань через веб-сокети, еквівалентні HTTP [5].

Простий приклад заголовка запиту зображений на рис. 2.

```
GET ws://websocket.example.com/ HTTP/1.1
Origin: http://example.com
Connection: Upgrade
Host: websocket.example.com
Upgrade: websocket
```

Рисунок 2 – Приклад заголовка запиту

Як сучасний протокол, зв'язок між джерелами відбувається прямо в WebSocket. WebSocket дозволяє спілкуватися між сторонами у будь-якому домені. Сервер вирішує, чи зробити свою службу доступною для всіх клієнтів, або тільки для тих, хто знаходиться в наборі чітко визначених доменів.

Кожне повідомлення, що передається між сервером WebSocket та клієнтом WebSocket, містить певний ключ, який називається «маскуючим» ключем і дозволяє будь-яким посередникам, сумісним із WebSocket, знімати «маску» та перевіряти повідомлення.

Якщо посередник не сумісний із WebSocket, повідомлення не може бути порушено. Браузер, який формує протокол WebSocket, обробляє «маскування» [6].

ВИСНОВКИ. Технологія WebSocket широко використовується майже в усіх веб-застосунках, де потрібний швидкий обмін інформацією, моніторинг та відображення статистики в режимі «реального часу».

Веб-сокети є досить гнучкими і користуються попитом не тільки у веб-застосунках, але і в інших програмах різноманітного призначення.

Технологія, що була розглянута, дозволяє створювати інтерактивне з'єднання між клієнтом (браузером) та сервером для обміну повідомленнями у режимі реального часу. Веб-сокети, на відміну від HTTP, дозволяють працювати з двонаправленим потоком даних, що робить цю технологію унікальною та, такою. що може бути успішно використаною в промислових автоматизованих ситемах [5–12].

ЛІТЕРАТУРА

1. Berners-Lee T., Cailliau R., WorldWideWeb: Proposal for a HyperText Project - Tech, rep., CERN, 1990. 18 p.
2. Jesse James Garrett. Ajax: A new Approach to Web Application. AdaptivePath, 44p.
3. Specification of the XMLHttpRequest object from the Level 1 W3C Working Draft released on April 5th, 2006. W3.org. 67 p.
4. Vanessa Wang, Frank Salim, Marcelo Jabali, The Definitive Guide to HTML5 - WebSocket, 2012, 200p.
5. Pener Lubbers, Brian Alers, Frank Sailm, Pro HTML5 Programming. Second edition. Apress, 2011. 353 p.
6. Шило Н. Ю. Зв'язок промислової автоматизації і контролюючих систем / Н. Ю. Шило // Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2020) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол. : І.Ш. Невлюдов та ін.]. Харків : ХНУРЕ, 2020. Вип. 2. – С. 129–133.
7. Шило Н. Ю. Засоби захисту систем промислової автоматизації та управління / Н. Ю. Шило // Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2020) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол. : І.Ш. Невлюдов та ін.]. Харків : ХНУРЕ, 2020. Вип. 2. С. 140–144.
8. Невлюдов І. Ш. Трансфер технологій у сучасній науці, освіті та виробництві в умовах четвертої промислової революції «ІНДУСТРІЯ 4.0» / І. Ш. Невлюдов, О. О. Чала, Ю. М. Олександров // Сучасний рух науки: тези доп. VIII міжнародної науково-практичної інтернетконференції, 3-4 жовтня 2019 р. Дніпро, 2019. Т.2 С.: 604–608.
9. Невлюдов И.Ш., Палагин В.А., Чалая Е.А. «Технологиии микросистемной техники (часть II)», НТЖ «Технология приборостоения». Х., 2015. №2.
11. Nevliudov, I., Yevsieiev, V., Maksymova, S., & Filippenko, I. (2020). Development of an architectural-logical model to automate the management of the process of creating complex cyber-physical industrial systems. Eastern-European Journal of Enterprise Technologies, 4(3 (106), 44–52. DOI: 10.15587/1729-4061.2020.210761.
12. Bortnikova, V., Nevliudov, I., Botsman, I., & Chala, O. (2019, June). Search Query Classification Using Machine Learning for Information Retrieval Systems in Intelligent Manufacturing. In ICTERI (pp. 460–465).

Науковий керівник: *Чала Олена Олександрівна, к.т.н., доцент кафедри КІТАМ Харківського національного університету радіоелектроніки.*