УДК 004.738.5

Целуєв В.О., здобувач,
Савіцький Р.С., аспірант, ст. викладач
Державний університет «Житомирська політехніка»

REAL-TIME КОМУНІКАЦІЯ У ІГРОВІЙ ПЛАТФОРМІ ЗАСОБАМИ WEBSOCKET

Сучасні ігрові соціальні мережі ставлять принципово нові вимоги до комунікаційної інфраструктури, а саме миттєва доставка повідомлень, сповіщення про ігрові події в реальному часі, одночасна підтримка десятків тисяч активних з'єднань. Традиційна модель HTTP-запитів за схемою «запит-відповідь» принципово не здатна задовольнити ці вимоги через надмірні накладні витрати на встановлення з'єднання та неможливість серверної ініціації передачі даних [1].

Порівняльний аналіз технологій real-time комунікації, таких як HTTP Long Polling, Server-Sent Events та WebSocket, засвідчив беззаперечну перевагу протоколу WebSocket (RFC 6455) для сценаріїв ігрової платформи [2]. WebSocket встановлює постійне повнодуплексне з'єднання між клієнтом і сервером поверх єдиного TCP-з'єднання, що забезпечує двосторонню передачу даних з мінімальними накладними витратами на рівні 2–10 байт на фрейм порівняно з кількома сотнями байт HTTP-заголовків [3].

Серверна частина запропонованого модуля реалізована на платформі Node.js з використанням бібліотеки Socket.IO версії 4.x, яка надбудовує над нативним WebSocket додатковий рівень абстракції. Існує автоматичний fallback до HTTP Long Polling для клієнтів без підтримки WebSocket, кімнати для групової адресації повідомлень та вбудований механізм повторного підключення з експоненційним відступом [4, 5].

Для горизонтального масштабування серверного кластера застосовується Redis Pub/Sub як розподілена шина повідомлень між вузлами. Кожен вузол Node.js підписується на канали Redis, що відповідають активним кімнатам чату, та ретранслює повідомлення підключеним до нього клієнтам. Така архітектура забезпечує прозору маршрутизацію повідомлень між користувачами, підключеними до різних вузлів кластера.

Розроблений модуль реалізує повний спектр комунікаційних сценаріїв ігрово платформи, зокрема особисті повідомлення між гравцями з підтвердженням доставки та прочитання, групові чати ігрових спільнот із підтримкою до 10 000 учасників, push-сповіщення

про ігрові, трансляція оновлень стрічки активності та індикація онлайн-статусу друзів у реальному часі.

Безпека з'єднань забезпечується обов'язковою JWT-автентифікацією при встановленні WebSocket-з'єднання, зокрема токен передається у параметрі запиту на апгрейд, перевіряється на сервері до прийняття з'єднання, а термін його дії обмежений 15 хвилинами з автоматичним оновленням через механізм refresh-токенів. Усі повідомлення передаються виключно через захищене з'єднання WebSocket Secure поверх TLS 1.3.

Вимірювання ключових метрик продуктивності проводилось за допомогою інструменту Artillery із симуляцією реалістичних сценаріїв навантаження. При 10 000 одночасних з'єднань на один вузол середня затримка доставки повідомлення становила 18 мс, 99-й перцентиль – 47 мс. Після горизонтального масштабування до трьох вузлів система стабільно обслуговує 30 000 активних підключень із пропускною здатністю до 85 000 повідомлень на секунду.

Реалізований модуль real-time комунікації демонструє ефективність протоколу WebSocket як основи для побудови комунікаційної інфраструктури геймерських платформ. Запропонована архітектура є масштабованою та може бути розширена для підтримки більших навантажень шляхом додавання нових вузлів до кластера без зміни програмного коду. Перспективним напрямом подальших досліджень є інтеграція протоколу WebRTC для організації голосового та відеозв'язку між гравцями безпосередньо в межах платформи.

Список використаних джерел:

1. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455. – IETF, 2011. – URL: https://www.rfc-editor.org/rfc/rfc6455.
2. Grigorik I. High Performance Browser Networking. – O'Reilly Media, 2013. – 396 p.
3. Casciaro M., Mammino L. Node.js Design Patterns. – Packt Publishing, 2020. – 660 p.
4. Socket.IO Documentation. – URL: https://socket.io/docs/v4/.
5. Lubbers P., Albers B., Salim F. Pro HTML5 Programming. – Apress, 2011. – 308 p.