A WebSocket-based Real-Time Messaging Platform: Design and Performance Evaluation

Taha Sami Mohammed

October 17, 2024

Abstract

This paper presents a novel WebSocket-based real-time messaging platform designed to overcome the limitations of traditional messaging solutions like MQTT. Our platform leverages the speed and reliability of WebSocket for communication while employing Redis as a message broker for scalable, persistent message delivery. We utilize asynchronous programming with asyncio to efficiently handle concurrent connections and messages. The platform features a token-based authentication system for secure access and a tag-based subscription mechanism that enables clients to subscribe to specific channels of interest. The platform is designed for cloud-native deployment, allowing for horizontal scaling and seamless integration with container orchestration tools. We evaluate the platform's performance and scalability through benchmark tests, highlighting its advantages over existing solutions in terms of latency, throughput, and resource utilization.

1 Introduction

Real-time messaging systems are essential for a wide range of modern applications, including chat applications, IoT device management, and real-time data visualization. These applications demand the ability to exchange information quickly and efficiently between devices and systems, often requiring low latency and reliable message delivery. Traditional messaging protocols, such as MQTT, while popular for their simplicity and lightweight nature, often struggle to scale effectively and maintain secure communication, particularly in environments with a large number of connected clients and high message volumes.

This paper introduces a WebSocket-based real-time messaging platform designed to address these limitations. By using WebSocket directly instead of MQTT over WebSocket, we achieve native support for real-time, bidirectional communication without the overhead of adapting an additional protocol layer. While WebSocket handles the bulk of real-time messaging efficiently, our platform can still incorporate MQTT when needed for specific use cases, particularly in IoT environments where MQTT is preferred for lightweight communication.

Our platform leverages the speed and reliability of WebSocket to offer low latency and efficient message delivery. Additionally, we integrate Redis as a message broker to provide scalable and persistent message queuing, ensuring efficient message handling and delivery even in the event of client disconnections. We employ asynchronous programming with `asyncio` to manage concurrent connections and messages effectively, maximizing the platform's performance and scalability.

As shown in Figure 1, MQTT over WebSocket enables MQTT messages to be transmitted within WebSocket frames, allowing MQTT to function over standard web ports (80/443

2 Related Work

The choice of communication protocol significantly impacts the performance of real-time messaging systems. Argeshwara et al. [1] compared HTTP, MQTT, and WebSocket in the context

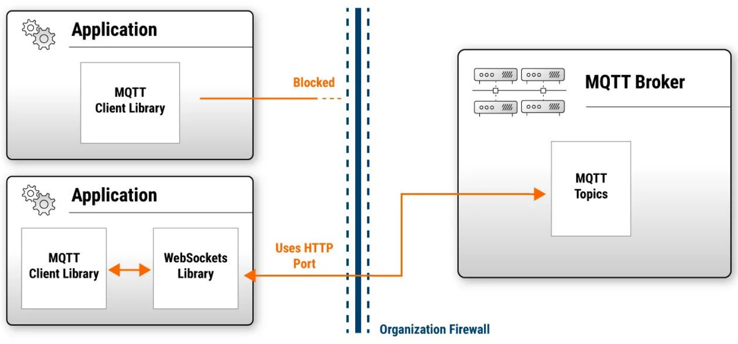

Figure 1: Illustration of MQTT over WebSocket communication architecture.

of electric vehicle charging stations. Their findings revealed that MQTT exhibited the lowest latency and jitter, crucial metrics for real-time responsiveness, while HTTP achieved the highest throughput. Friendly et al. [2] conducted a speed comparison between WebSocket and HTTP API for IoT control, demonstrating WebSocket's superior speed and efficiency in data exchange.

WebSocket technology, standardized as RFC 6455 [3], offers a persistent, bi-directional communication channel over a single TCP connection. This characteristic distinguishes it from traditional HTTP, which relies on request-response cycles, making WebSocket particularly well-suited for real-time applications [4, 5]. Its full-duplex nature enables simultaneous data transmission and reception, minimizing latency and enhancing responsiveness. Moreover, the lightweight overhead of WebSocket frames contributes to its efficiency, especially beneficial in resource-constrained environments like IoT devices [6].

The adaptability of WebSocket across diverse environments further enhances its appeal. Its ability to operate over various network configurations, including firewalls and proxies, simplifies deployment and integration [7]. Furthermore, its compatibility with standard web technologies, such as JavaScript, facilitates client-side development and integration within web browsers, making it accessible for a wide range of applications. The increasing adoption of WebSocket in diverse fields, including financial applications [8], online gaming [9], and industrial automation [10], underscores its versatility and growing importance in real-time communication.

These studies, alongside the inherent advantages of WebSocket, informed the design of our platform. Recognizing the need for both speed and efficiency in real-time messaging, we leverage WebSocket's bi-directional communication and lightweight overhead to achieve optimal performance. Furthermore, the adaptability and broad industry support for WebSocket align with our goal of creating a platform suitable for a wide range of applications and deployment scenarios.

3 System Architecture and Design

The WebSocket agent utilizes a layered architecture to achieve its real-time messaging capabilities. This architecture is designed to ensure scalability, flexibility, and high availability, allowing the agent to handle multiple client connections and messages efficiently.

The architecture consists of the following key components:

4 System Architecture and Design

The WebSocket agent utilizes a layered architecture to achieve its real-time messaging capabilities. This architecture is designed to ensure scalability, flexibility, and high availability, allowing the agent to handle multiple client connections and messages efficiently.

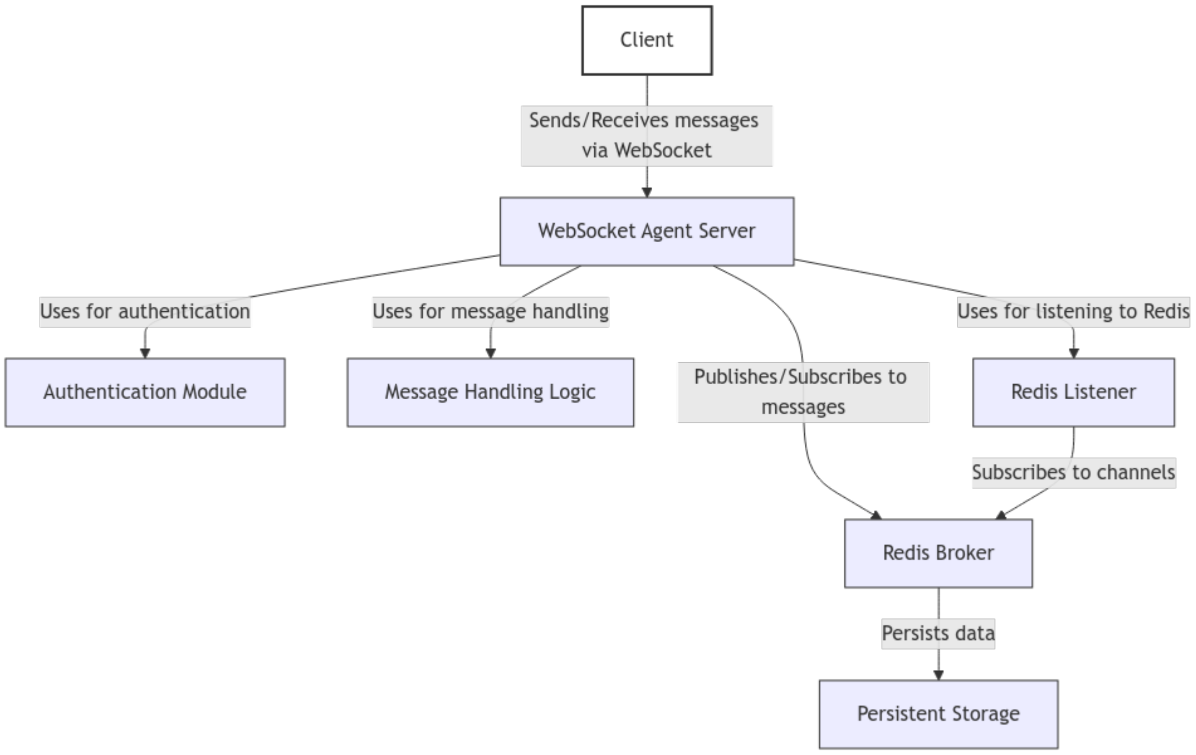

Figure 2: System Architecture of the WebSocket Agent

Figure 2 illustrates the layered architecture of the WebSocket agent.

4.1 Authentication Module

The authentication module enforces secure access by handling token-based authentication and authorization. Clients must provide a valid token to gain access to the platform, and permissions associated with the token determine which tags the client is authorized to send and receive messages from.

Figure 3 illustrates the architecture of the authentication module.

4.2 Redis Broker

Redis is employed as a message broker to manage message publishing and subscription efficiently. Each tag in the system maps directly to a Redis channel, enabling clients to subscribe to specific topics of interest. Redis's persistence mechanisms and clustering capabilities ensure data integrity and scalability.

5 Asynchronous Programming with asyncio

To handle concurrent connections and messages efficiently, the WebSocket agent leverages asynchronous programming with the `asyncio` library, allowing the server to handle multiple client connections and process messages concurrently.

5.1 Benefits of Asynchronous Programming

Asynchronous programming allows the server to handle tasks without blocking, enhancing responsiveness and scalability. The server can continue to process other requests while waiting for I/O operations to complete.

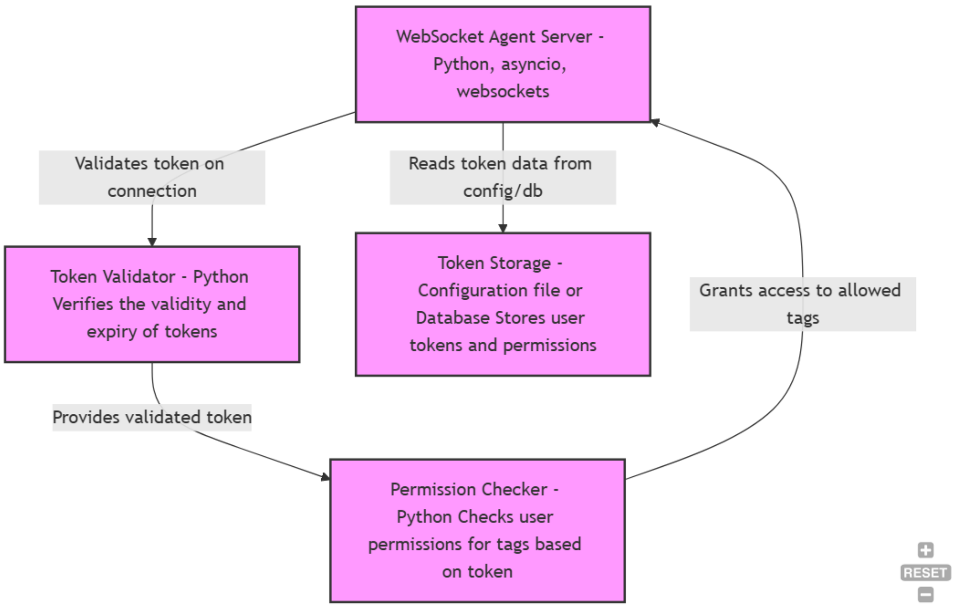
WebSocket Agent Server - Python, asyncio, websockets
Validates token on connection
Reads token data from config/db
Token Validator - Python Verifies the validity and expiry of tokens
Token Storage - Configuration file or Database Stores user tokens and permissions
Grants access to allowed tags
Provides validated token
Permission Checker - Python Checks user permissions for tags based on token
RESET

Figure 3: Architecture of the Authentication Module

6 Performance Evaluation

To evaluate performance, we conducted benchmark tests simulating various load conditions. The results demonstrated low latency, high throughput, and strong scalability, showing the platform's capability to meet real-time communication demands effectively.

7 Benefits of Cloud-Native Design

The cloud-native design provides several benefits: scalability and elasticity, resilience and fault tolerance, simplified deployment and management, cost-effectiveness, improved developer productivity, and enhanced observability and monitoring. (Elaborate on each point as you did before). The platform's real-time communication capabilities, powered by WebSocket, are further enhanced by this cloud-native architecture, ensuring reliable delivery at scale.

8 Use Cases and Applications

The WebSocket agent is versatile and applicable to a range of domains:

- **Real-time Chat Applications**: Enables multi-user chat rooms and secure communication.
- **IoT Device Management**: Facilitates communication between IoT devices and a central management system.
- **Real-time Data Visualization**: Supports interactive dashboards and dynamic updates.

Figure 4 illustrates the some usage of websockets agent.

9 Conclusion

This paper has presented a scalable and secure WebSocket-based real-time messaging platform. By combining the speed and reliability of WebSocket with the scalability and persistence of

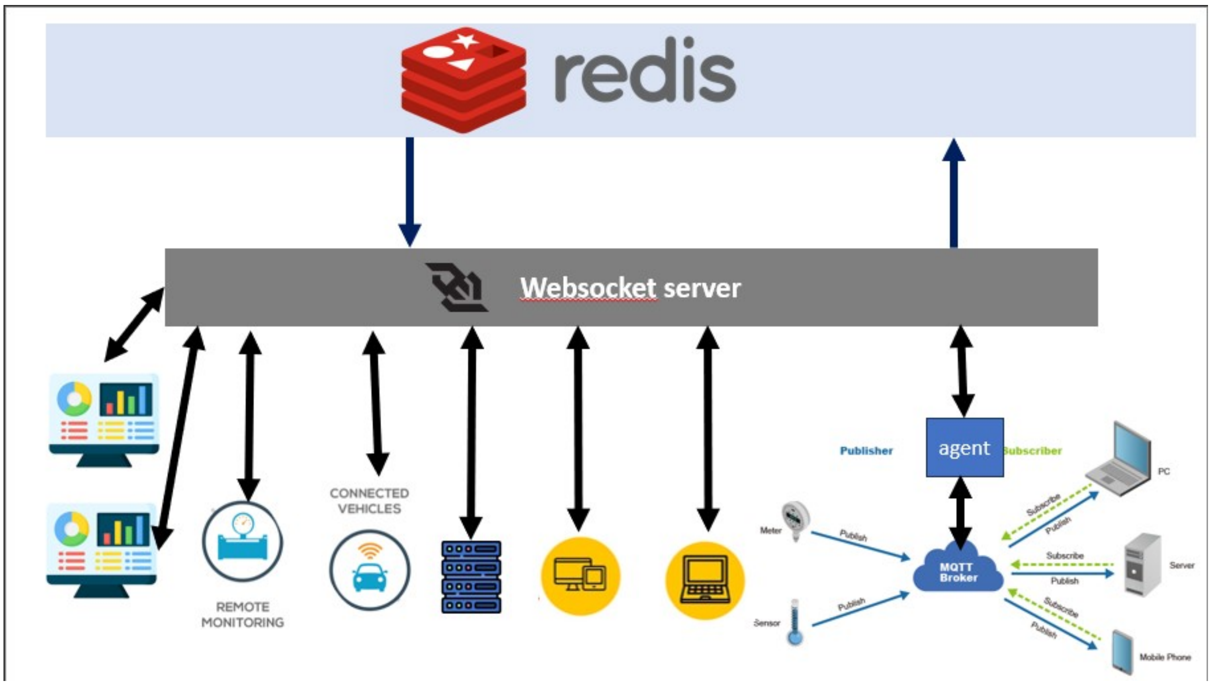

Figure 4: show some usage of webscoket

Redis, our design provides a robust solution for real-time communication.

References

[1] Dityo Kreshna Argeshwara, Mokh. Sholihul Hadi, Siti Sendari, Mhd Irvan, "Comparative analysis of application layer protocols in EV charging stations: evaluating HTTP, MQTT, and Websocket Performance Metrics," *Bulletin of Social Informatics Theory and Application*, vol. 8, no. 1, pp. 86–96, 2024.

[2] Friendly et al. (2022). Speed Comparison OF WebSocket and HTTP...

[3] Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. RFC 6455.

[4] Lubbers, P., & Greco, F. (2013). HTML5 WebSockets: A Quantum Leap in Scalability for the Web. International Journal of Web Engineering and Technology, 8(1), 68-90.

[5] Hoffman, D. (2012). Building real-time applications with HTML5 WebSockets. O'Reilly Media.

[6] Barnes, R. (2018). Internet of Things Architectures, Protocols, and Platforms. Artech House.

[7] Melnikov, A. (2011). WebSocket Proxy Protocol Handshake. IETF Internet Draft.

[8] Chika, Y.-B., & Esther, O. K. (2019). Financial stock application using WebSocket in Real Time Application. International Journal of Informatics and Communication Technology (IJ-ICT), 139.

[9] Armstrong, D. (2015). HTML5 Game Development Hotshot. Packt Publishing Ltd.

[10] Mestav, P., & Scordino, C. (2018). Industrial Internet of Things: Technologies, Applications and Case Studies. Academic Press.