High-Performance Cloud-Based Architecture for Real-Time Patient Monitoring and Alerting in IoMT Environments

Vahagn Aleksanyan
Yerevan State University
Yerevan, Armenia
e-mail: vahagn.aleksanyan@gmail.com

Narek Petrosyan
Yerevan State University
Yerevan, Armenia
e-mail: pnarek375@gmail.com

Anna Hovakimyan
Yerevan State University
Yerevan, Armenia
e-mail: ahovakimyan@ysu.am

***Abstract*—Growing demand for real-time patient monitoring in modern healthcare has accelerated the adoption of Internet of Medical Things (IoMT) technologies. This paper presents a high-performance, cloud-based architecture that supports the continuous collection of biometric data, intelligent processing, and real-time alerting during critical events. The proposed system utilizes lightweight communication protocols, such as MQTT and WebSocket, for low latency and effective data transfer between wearable medical devices and a cloud server. The backend is optimized using modular C++ components, containerized services, and TimescaleDB for scalable storage of time-series data. Dynamic integration of real-time risk assessment plugins enables the analysis of patient data streams and the issuance of emergency alerts. Mobile applications for patients and doctors offer responsive interfaces for visualizing historical data, sending real-time notifications, and enabling secure doctor-patient messaging. Altogether, this system provides a practical and scalable solution to the performance and reliability challenges of remote health monitoring in IoMT environments.**

***Keywords*—Internet of Medical Things, IoMT, real-time monitoring, cloud computing, MQTT, TimescaleDB, patient alerting, high-performance architecture.**

I. Introduction

Real-time monitoring of patient health is a key aim of modern healthcare, and is increasingly enabled by the Internet of Medical Things (IoMT)—interconnected medical devices and systems that continuously collect and transmit biometric data [1]. IoMT-based remote patient monitoring (RPM) lets clinicians track vitals in real time, improving convenience for patients with chronic conditions and enabling earlier intervention; studies report benefits for clinical decision-making, timely responses, and long-term cost reduction [2]–[4].

To process continuous data streams at scale, IoMT architectures commonly rely on cloud computing for elastic storage and compute [5]. Cloud-connected sensor networks improve quality of service and reliability, while aggregated vital signs can be analyzed to detect risks and trigger alerts; recent platforms increasingly apply machine learning for anomaly detection and prediction [2], [6].

Low-latency communication is critical. MQTT provides lightweight, publish/subscribe messaging with configurable QoS, suiting resource-constrained devices and real-time vital streaming [7]. For clinician- and patient-facing apps, WebSockets maintain bidirectional connections, so alerts and live data can be pushed without polling, achieving sub-second updates in practice [8]. Together, MQTT for device-to-cloud ingestion and WebSockets for cloud-to-user delivery enable end-to-end responsiveness.

Finally, modularity and scalability are addressed via a microservice design deployed in containers (e.g., Docker), allowing independent development, scaling, and updates of services for ingestion, analytics, storage, and notification [5]. This cloud-backed, containerized approach underpins the real-time performance and extensibility targeted in this work. The following sections review related efforts and detail the proposed architecture.

II. Related Work

IoMT-based patient monitoring has been widely studied, with surveys cataloging applications, common architectures, and challenges such as data heterogeneity, mobility, and QoS constraints [4]. Early efforts connected wearable sensors to cloud services for remote review and disease-specific management, reporting benefits for clinical decisions and reduced readmissions [2], [3], [6].

A key theme is communication efficiency. MQTT is frequently adopted for device-to-cloud streaming thanks to its lightweight pub/sub model and QoS options; studies recommend mid-level QoS to balance latency and reliability in healthcare traffic [7]. For cloud-to-user delivery, event-driven push (e.g., WebSockets) replaces HTTP polling, achieving sub-second updates in telehealth dashboards and enabling instant alerting [8], [9]. The literature increasingly converges on MQTT plus real-time push as a pragmatic foundation for time-critical IoMT [7], [8].

Architecturally, multi-tier designs (device, edge/fog, cloud) are used to trade latency vs. capacity [4], while cloud-centric systems expose standardized APIs and leverage NoSQL or time-series stores for high-volume streams [5]. Microservices and containerization (e.g., Docker) improve scalability and evolvability across ingestion, analytics, storage, and notification services [5]. Interoperability through gateways or standards (e.g., HL7 FHIR) supports integration with hospital systems. Beyond thresholds, recent work applies ML to detect anomalies and predict adverse events from IoMT data; other efforts combine IoMT with blockchain for trustworthy event logging and alerts [2], [6], [10].

Despite this progress, gaps remain: many solutions emphasize connectivity or analytics in isolation, struggle to scale under high-frequency, multi-sensor workloads, or hard-wire alert logic into monoliths [5]. To address these limitations, we adopt a plugin-based architecture that lets new analysis modules be integrated dynamically—unlike fixed-function designs [1], [4]—and we employ event-driven, real-time alerting that uses MQTT for ingestion and WebSocket push instead of periodic polling [3], [7], [8]. At the core is a high-performance, modular C++ backend engineered for non-blocking I/O and horizontal scaling, sustaining low-latency processing under load. The resulting platform is general-purpose and extensible—capable of incorporating new devices, analytics methods, and even add-ons such as blockchain-backed logging [2]—while preserving end-to-end responsiveness.

III. System Architecture

A. Overview of the System

The system provides a high-performance, scalable platform for real-time remote monitoring in IoMT settings. Wearable devices, mobile apps, and cloud services work together to capture, normalize, transmit, analyze, and store biometric data, and to issue emergency notifications. The architecture is modular for extensibility and maintainability. At the core is a cloud backend with a message broker, processing server, database layer, and attachable risk-assessment plugins. The patient app connects to wearables via BLE/GATT, normalizes readings, and publishes them via MQTT. The physician app provides real-time views, historical data, alerts, and secure messaging. Services are deployed in containers/VMs for horizontal scaling and fault tolerance. Time-series data use TimescaleDB; relational data (users, roles, alerts) use PostgreSQL.

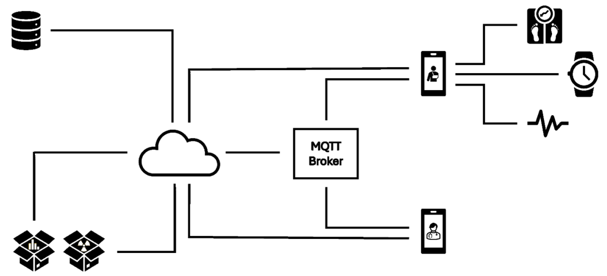

Fig. 1. The Architecture of the System

B. Cloud Infrastructure

The backend is implemented in modern C++ (C++26) with attention to performance, memory, concurrency, and I/O. Components run independently in containers. **Key components:**

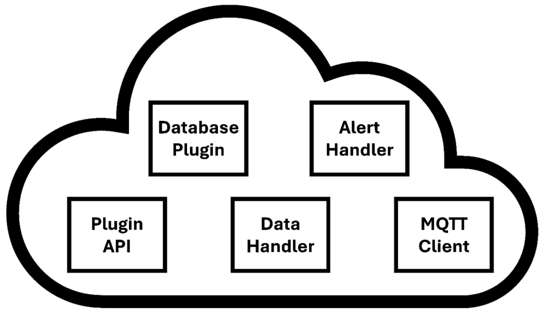

Fig. 2. The Architecture of the Cloud Infrastructure

- **MQTT Broker** – Low-latency publish/subscribe ingress for device telemetry.
- **Main Processing Server** – Subscribes to topics, validates/decodes messages, manages auth/sessions, and routes work.
- **Database Plugin Layer** – Decoupled writers and analysis hooks; plugins can attach to specific streams.
- **Risk Assessment Modules** – Real-time rules/heuristics over single or combined signals; emit alerts and persist results.

All services are containerized (Docker) and deployed on VMs with load balancing, centralized logging/monitoring, and rolling updates. The modular design eases incremental expansion.

C. Mobile Application

The mobile client targets Android and iOS using native stacks (Java/Kotlin, Swift) and a shared C++ core for common logic and performance. BLE integration uses platform APIs with attention to power and intermittent connectivity; device-specific adapters normalize payloads without affecting server contracts. **Main layers:**

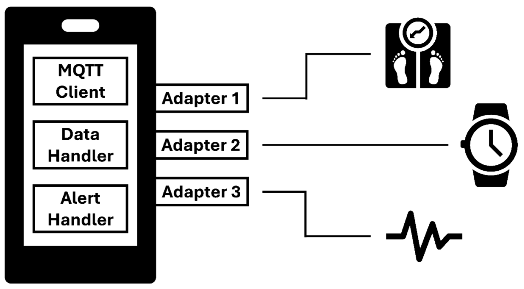

Fig. 3. The Architecture of the Mobile Application

- **Device Interface** – BLE discovery, pairing, and GATT subscriptions (heart rate, glucose, etc.).
- **Adapters** – Per-device normalization to an internal JSON schema.
- **Communication** – MQTT for uplink telemetry; WebSocket for real-time updates/alerts.

- **UI** – Live and historical charts, alerts, and secure chat for patient/physician roles.
- **Caching/Offline** – Local buffering and later sync on poor connectivity.
- **Role-Based Views** – Conditional features and layouts for patients vs. clinicians.

D. Data Storage and Time-Series Management

Adapters produce standardized JSON published to MQTT; the backend (as subscriber) timestamps, authenticates, and prepares data for storage. PostgreSQL provides relational storage, while TimescaleDB handles time-series ingestion and querying via hypertables (automatic time partitioning). Partitioning by patient/time and indexes on IDs/timestamps enable fast filters and trends. Non-biometric data (alerts, chat, events) reside in separate schemas. Retention, compression, and archival policies are managed by Timescale background jobs to balance performance, cost, and lifecycle needs.

E. Data Flow

Wearables stream via BLE; the app discovers services (GATT), subscribes to characteristics, normalizes readings, and publishes to MQTT topics. The backend processes messages in near real time, persists them, and runs plugin analysis (e.g., combined heart-rate/glucose spikes). On detection, an alert is stored and pushed via MQTT; patient and physician apps receive it immediately. Secure WebSocket channels support bidirectional messaging and acknowledgments. The pipeline minimizes latency while isolating failures across components.

F. Extensibility, Modularity, and Performance Considerations

C++26 services provide low-latency, multi-threaded paths for parsing, routing, and analysis. Containerized microservices scale horizontally; MQTT supports high concurrency with low overhead; TimescaleDB sustains high read/write rates using hypertables and compression. VM-backed deployments with containers allow elastic scaling behind load balancers. Fault tolerance includes retries, redundant instances, and database replication.

Stress testing (qualitative). We simulated rising device counts and high-frequency telemetry. Horizontal scaling (additional service replicas) maintained throughput without noticeable latency growth; the asynchronous C++ backend handled thousands of concurrent streams, and redundancy/replication avoided data loss during failover. Collectively, these measures indicate readiness for real-world IoMT loads while preserving real-time responsiveness.

IV. System Operation Flow

To better illustrate the practical operation of the system, consider the following real-life scenario involving a patient with diabetes and a cardiovascular condition.

The patient uses a body-worn glucose monitoring device alongside a heart rate monitor, both of which communicate with the patient's smartphone via Bluetooth Low Energy (BLE) technology. At 08:17 AM, the measured heart rate is 132 beats per minute, with the blood glucose level recorded as 210 mg/dL. Both readings are transmitted automatically to the mobile app, where device-specific adapters are used to decode the data and normalize it into structured JSON payloads.

Once the transformation process is complete, the application publishes the data to the respective MQTT topics—namely, `/patient/{id}/heart_rate` and `/patient/{id}/glucose`—using the smartphone's internet connectivity. The cloud-based backend receives messages within milliseconds because it is subscribed to the above topics and designed to handle thousands of concurrent data streams.

Once the data reaches the backend, it undergoes authentication, timestamping, and validation processes. Afterward, it is stored in the TimescaleDB database to enable longitudinal monitoring. At the same time, the system sends the readings to a specifically designed real-time risk estimation plugin to detect dangerous multi-signal patterns.

This plugin is aware that an increase in heart rate concurrently with blood glucose levels can indicate a serious health threat, such as cardiac stress due to hyperglycemia. Therefore, this plugin immediately triggers an emergency notification, logs the incident in the database, and the server publishes a message to the `/alerts/patient_id` topic. Both the patient receiving medical treatment and the physician in charge of their care have signed up for this notification service via their mobile applications. As a result, both are given a push message in real time. The patient's application has a prominent red alert banner added to a list of emergency recommendations, including rest, fluids, and the acquisition of emergency medications that may be required. In contrast, the doctor is given a complete overview of the patient's vital signs, trend history, and a structured list of recent measurements.

Realizing the urgency of the situation, the doctor initiates a secure messaging session via the application's inbuilt messaging system, which is based on an ongoing WebSocket connection. The doctor advises that the patient take a dose of fast-acting insulin and continue to be monitored. The patient acknowledges receiving the message, and both continue to maintain their connection until the readings stabilize.

In this interaction, the system continuously collects, stores, and evaluates data in real-time. Any significant future changes trigger additional alerts, keeping both the patient and the physician adequately updated on time. This scenario underscores how the system seamlessly bridges data collection, analysis, and clinical response. The entire flow—from sensor to server to human action—completes in seconds, demonstrating how high-performance cloud infrastructure and modular architecture can directly support timely, life-saving interventions in an IoMT environment.

V. Future Work

While the proposed system provides a strong foundation for real-time patient monitoring in IoMT environments, several opportunities for future enhancement remain.

First, the current system supports specialized plugin modules for risk assessment purposes. However, they operate independently and utilize manually defined heuristics. Future research may focus on the development of more sophisticated, cooperative plugin systems that can handle multiple signal trends and apply dynamic rule learning based on historical data sets. These improvements could significantly enhance the early detection of complex medical events.

Second, the system's scope can be extended to support a broader range of wearable devices, including smartwatches and multi-sensor health kits. To facilitate this, a standardized plugin SDK for device vendors may be introduced, allowing third-party developers to contribute data normalization adapters and expand the supported device ecosystem with minimal integration effort.

The mobile application, although built with offline support and caching, can be further improved by integrating advanced synchronization techniques, data prioritization algorithms, and local trend analysis capabilities. These improvements would be especially beneficial for residents of rural or underprivileged communities where network access is usually poor.

The interoperability with health information systems is a significant area for future research studies. The use of standardized protocols, including HL7 FHIR and DICOM, is promising because it can enable open communication between the intended system and hospital Electronic Medical Records (EMRs), improving the care loop and data transparency.

Implementing these guidelines would enhance the system, resulting in a more comprehensive and intelligent healthcare system that can efficiently manage various medical situations while being flexible, secure, and easy to maintain.

VI. Conclusion

This paper presents a high-performance, cloud-focused architecture for real-time patient monitoring in IoMT environments, addressing the need for responsive, adaptive, and modular healthcare solutions. The combination of lightweight communication protocols, such as MQTT and WebSocket, with modern C++ services and TimescaleDB optimized for time series, enables the proposed system to achieve low-latency data transmission, efficient processing, and a reliable alerting mechanism during medical emergencies.

The use of modules for real-time risk assessment, containerized deployment strategies, and dedicated mobile applications for both patients and medical professionals enables seamless information exchange, dynamic alerting, and continuous communication across the care loop. The proposed system not only ensures early detection of important health events but also provides a flexible framework for future growth and compliance with evolving medical guidelines.

Through its modular design, platform independence, and extensibility, the proposed solution demonstrates how cloud technologies can be effectively leveraged to meet the demands of next-generation digital health platforms. It lays the groundwork for further innovation in IoMT systems, promoting safer, smarter, and more connected healthcare delivery.

References

[1] C. Huang *et al.*, "Internet of Medical Things: A systematic review," *Neurocomputing*, vol. 557, pp. 34–48, Nov. 2023, doi: 10.1016/j.neucom.2023.126719.

[2] S. B. Othman and M. Getahun, "Leveraging blockchain and IoMT for secure and interoperable electronic health records," *Scientific Reports*, vol. 15, no. 1, Art. no. 12358, Apr. 2025, doi: 10.1038/s41598-025-95531-8.

[3] L. P. Serrano *et al.*, "Benefits and challenges of remote patient monitoring as perceived by health care practitioners: A systematic review," *The Permanente Journal*, vol. 27, no. 4, pp. 100–111, 2023, doi: 10.7812/TPP/23.022.

[4] K. Boikanyo, M. Adamu, B. Sigweni, A. Yahya, and C. Lebekwe, "Remote patient monitoring systems: Applications, architecture, and challenges," *Scientific African*, vol. 19, Art. no. e01638, 2023, doi: 10.1016/j.sciaf.2023.e01638.

[5] J. Mateo-Fornés *et al.*, "An Internet of Things platform based on microservices and cloud paradigms for livestock," *Sensors*, vol. 21, no. 17, Art. no. 5949, Sep. 2021, doi: 10.3390/s21175949.

[6] A. Ahila *et al.*, "A smart IoMT-based architecture for E-healthcare patient monitoring system using artificial intelligence algorithms," *Frontiers in Physiology*, vol. 14, Art. no. 1125952, Jan. 2023, doi: 10.3389/fphys.2023.1125952.

[7] F. Shahid *et al.*, "Adaptive lightweight security for performance efficiency in critical healthcare monitoring," in *Proc. 14th IEEE Int. Symp. Medical Information and Communication Technology (ISMICT)*, pp. 1–6, May 2024, doi: 10.1109/ISMICT61996.2024.90738175.

[8] F. Nehru and A. Yudertha, "A proposed design of real-time patient monitoring system using WebSocket as a basis of telemedicine," in *Proc. SICONIAN 2019 – Sriwijaya Int. Conf. on Information Technology and Its Applications*, Adv. Intell. Syst. Res., vol. 172, pp. 263–268, 2020, doi: 10.2991/aisr.k.200424.040.

[9] N. Kirilov and M. Dugas, "Evaluation of technical approaches for real-time data transfer from electronic health record systems," *Computer Methods and Programs in Biomedicine*, vol. 255, Art. no. 108347, 2024, doi: 10.1016/j.cmpb.2024.108347.

[10] M. A. Khan *et al.*, "Smart Steering Wheel: Design of IoMT-based non-invasive driver health monitoring system to enhance road safety," *IET Intelligent Transport Systems*, vol. 19, no. 1, Art. no. e70012, Jan. 2025, doi: 10.1049/itr2.70012.