Serdica Journal of Computing 19(1), 2025, pp. 1-15, 10.55630/sjc.2025.19.1-15
Published by Institute of Mathematics and Informatics, Bulgarian Academy of Sciences

Methodology for Selecting Communication Protocols in M2M Systems

Oleg Iliev

Laboratory of Telematics,
Institute of Mathematics and Informatics,
Bulgarian Academy of Sciences, Bulgaria
iliev.oleg@gmail.com

Abstract

This article presents a methodology for the analysis and classification of communication protocols and data stream management systems used in machine-to-machine (M2M) environments. The focus is placed on non-functional characteristics such as latency, reliability, scalability, security, and message ordering, and their relevance in protocol selection across different functional domains. A comparative evaluation of established protocols – including MQTT, HTTP/1.1-3, Kafka, Pulsar, RabbitMQ, AMQP, and CoAP – is conducted based on objective criteria and existing literature. Building on this foundation, hybrid architectures are proposed that combine multiple technologies according to the criticality and technical requirements of specific scenarios. The paper offers structured recommendations for communication strategies in M2M contexts, with empirical validation planned as the subject of future work.

Keywords: communication protocols, non-functional requirements, M2M, IoT, hybrid architecture, latency, reliability, classification

ACM Computing Classification System 2012: Networks → Network architectures → Network design principles → Layering, Computer systems organization → Architectures → Other architectures → Data flow architectures

Mathematics Subject Classification 2020: 68M10

Received: November 27, 2025, *Accepted:* December 23, 2025, *Published:* December 30, 2025
Citation: Oleg Iliev, Methodology for Selecting Communication Protocols in M2M Systems, Serdica Journal of Computing 19(1), 2025, pp. 1-15, https://doi.org/10.55630/sjc.2025.19.1-15

1 Introduction

In machine-to-machine (M2M) systems, the selection of a communication protocol is often driven by technological preferences or industry norms, without sufficient consideration of the actual application context. This frequently leads to the adoption of solutions that provide high levels of security, reliability, and message ordering - at the cost of unnecessary overhead, latency, or operational complexity in cases where such guarantees are not essential [1]. As M2M communication becomes a fundamental building block of contemporary cyber-physical systems, ensuring efficient data exchange across heterogeneous devices is critical [2].

Over the past decade, a diverse ecosystem of communication technologies has emerged, ranging from traditional request-response protocols to publish-subscribe and streaming platforms such as MQTT, RabbitMQ, Apache Kafka, and Apache Pulsar [3, 4]. Each technology exhibits distinct characteristics regarding latency, throughput, reliability, and operational complexity [5]. Despite the abundance of solutions, the selection of an appropriate protocol remains a non-trivial engineering problem, as existing studies predominantly focus on technology-centric comparisons, often evaluating protocols in isolation or under narrowly defined conditions [6].

The importance of the functional domain, defined by application semantics and operational requirements, is recognized as a decisive factor in the selection process. Many real-world M2M systems operate under constrained infrastructure conditions, where the ideal communication substrate may not be available. Legacy installations and industrial sites often rely on pre-existing automation-specific networks, which motivates the adoption of hybrid communication architectures that combine different protocols and transport technologies [7, 8].

The aim of this paper is to analyze the non-functional characteristics of communication protocols and technologies in the context of specific functional domains. The analysis is based on three realistic and diverse M2M scenarios, which represent different sets of constraints and requirements:

1. Sensor data collection: This domain encompasses a wide range of telemetry applications, from non-critical environmental monitoring (e.g., ambient temperature) to high-priority healthcare systems. In the case of medical sensors, the transmission of vital signs requires low latency and high reliability to ensure patient safety, while environmental sensors often operate on battery power and prioritize energy efficiency and resource optimization over constant connectivity [5, 9].
2. Ticket vending kiosk systems: These represent typical order-creation and trans-

actional environments. Such systems require strict message ordering to prevent logical errors in the purchasing process, high data security for handling sensitive user information, and robust delivery guarantees. In this context, fault tolerance and the ability to recover from network disruptions are paramount to maintain service availability and transactional integrity [2, 10].

3. Real-time data streaming (sports betting odds): This scenario involves the ingestion and distribution of high-frequency data from multiple sources. The primary requirements here are ultra-low latency and the capacity to handle massive throughput. Unlike transactional systems, a degree of message loss may be tolerable if subsequent updates refresh the data within milliseconds, but the system must scale dynamically to accommodate peak loads during live events [3, 8].

This paper defines and justifies the key non-functional metrics by which communication protocols are evaluated across these scenarios. The analysis includes indicators such as latency (including 99th percentile), message loss, system resource usage, security, reliability, and scalability. This methodological consolidation serves as a necessary prerequisite for subsequent empirical validation [10]. Future research is planned to involve controlled measurements and simulations across these domains to complement the analytical framework presented in this study.

2 Non-functional characteristics: definition and significance in the context of M2M functional domains

The selection of a communication protocol in machine-to-machine (M2M) systems cannot rely solely on functional requirements without considering the impact of non-functional characteristics. These characteristics, although often overlooked during design, significantly influence the performance, reliability, security, and resilience of the system, especially in scenarios where communication is a critical component.

This paper identifies and systematizes eleven key non-functional characteristics, including latency, delay tolerance, message loss tolerance, message ordering, throughput and payload size, scalability, resource usage, security, reliability, interoperability, and fault tolerance & recovery. These characteristics directly affect the adequacy of protocol selection. They have been derived based on a comprehensive literature review and an analysis of real-world functional scenarios that demonstrate different requirements and constraints. Particular attention is paid to the interaction between these characteristics and functional domains, as well as

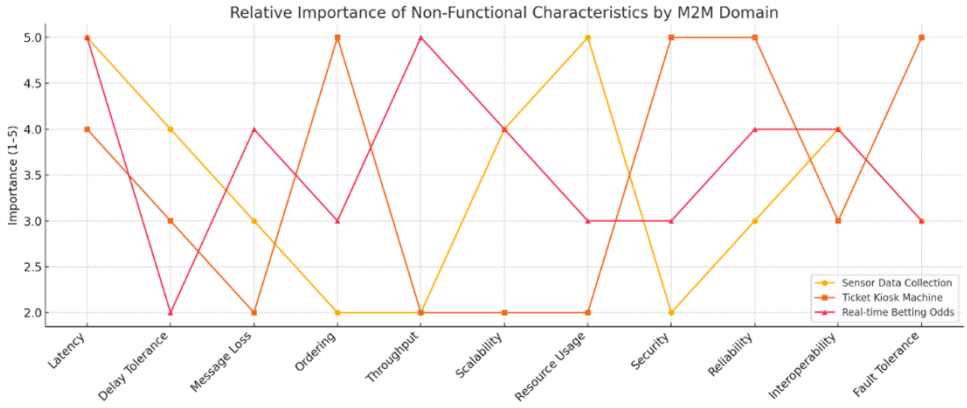

Figure 1: Relative importance of non-functional characteristics by M2M domain.

the need to measure not only average values but also extreme deviations through percentile evaluation (99p), enabling a more accurate assessment of system behavior under load or in critical situations.

The literature emphasizes the importance of non-functional characteristics for the effective implementation of M2M communication, particularly in the context of specific applications such as healthcare (e.g., vital sign monitoring), industrial automation (e.g., machine and process control), and public information services (e.g., ticket vending kiosks and terminals). [1] conduct a comparative analysis of communication protocols such as MQTT, CoAP, HTTP, AMQP, and DDS, using criteria like latency, throughput, reliability, security, and scalability. Their conclusion is that no universal protocol exists and that selection should be domain-specific. [2] investigate kiosk system design, emphasizing security, fault tolerance, and message ordering. [3] empirically compare MQTT and HTTP in real-time conditions, revealing significant differences in latency and resilience to message loss in the context of data streaming.

The following characteristics are summarized and categorized according to their relevance in three representative functional domains: 1) sensor data collection, 2) ticket vending kiosk systems, and 3) real-time collection of sports betting odds. The evaluation scale ranges from 1 (low importance) to 5 (high importance). Table 1 presents the numerical evaluation, and Figure 1 visualizes the comparison across domains. The analysis of these characteristics is not intended as a purely technical classification but emphasizes the need for contextual selection based on actual application needs.

Characteristic	Sensor Data Collection	Ticket Kiosk Machine	Real-Time Betting Odds
Latency	5	4	5
Delay Tolerance	4	3	2
Message Loss Tolerance	3	2	4
Message Ordering	2	5	3
Throughput & Payload Size	2	2	5
Scalability	4	2	4
Resource Usage / Overhead	5	2	3
Security	2	5	3
Reliability	3	5	4
Interoperability	4	3	4
Fault Tolerance & Recovery	3	5	3

Table 1: Importance of non-functional characteristics by functional domain (scale: 1–5).

2.1 Latency

Latency is defined as the time between the sending and receiving of a message. It is a key metric in applications requiring real-time responsiveness. Measuring the 99th percentile allows for the detection of rare but critical cases that can impact reliability. In sensor systems for health monitoring and in betting odds streaming, latency is crucial. In kiosk systems, while important, some degree of tolerance is allowed, especially when short performance drops are acceptable.

2.2 Delay tolerance

This characteristic describes the system's ability to function correctly despite communication delays. Non-critical sensor systems, such as temperature monitoring, can tolerate high delay, while medical or financial applications require minimal latency. In some IoT scenarios with energy-constrained devices, this becomes a key trade-off between accuracy and endurance.

2.3 Message loss tolerance

Message loss is acceptable in applications that rely on frequent updates, such as betting odds that refresh multiple times per second. In kiosk systems and healthcare applications, message loss may lead to functional failure and is unac-

ceptable. Protocols with acknowledgment or redelivery mechanisms are mandatory in such contexts.

2.4 Message ordering

Messages must be delivered in the correct sequence in systems relying on sequential processing - for example, kiosks that handle requests step by step. In sensor and streaming applications, this is less critical, unless the data represents a time series where order matters.

2.5 Throughput and payload size

Protocols must support the required transmission frequency and data volume. Betting systems require high throughput and minimal latency. Kiosk systems deal with infrequent but possibly large and structured messages. Sensor data tends to be small but frequent; optimization can be achieved through compression or aggregation.

2.6 Scalability

Scalability refers to the system's ability to support an increasing number of devices. It is essential for large-scale sensor networks and data aggregation platforms for bookmakers, where data sources are numerous and often asynchronous. In kiosk systems, where the number of terminals is predictable, this is of lower priority.

2.7 Resource usage / overhead

This is critical for devices with limited resources—such as battery-powered sensors or IoT edge devices. Heavy protocols with encryption and acknowledgment can further strain processing resources and reduce device lifetime. In server-based systems (e.g., kiosks), this is less important, though still worth considering when scaling to hundreds of devices.

2.8 Security

Includes authentication, encryption, and access control. It is critical for kiosk systems and medical data. For non-critical sensor measurements, it is of lower priority. Protocols like HTTPS, MQTT with TLS, and AMQP offer varying levels of protection; the choice depends on the sensitivity of the transmitted data.

2.9 Reliability

The ability to deliver messages without loss or duplication. It is essential in transactional environments. In streaming systems, compromises can be accepted using QoS mechanisms. Reliability assessment should be based on both real measurements and simulations under load and failure scenarios.

2.10 Interoperability

Systems must exchange data regardless of the platforms, network standards, or operating systems used. This is especially important in heterogeneous data aggregation environments, such as bookmaker platforms. Standards like REST, JSON, and multi-transport protocol support facilitate interoperability.

2.11 Fault tolerance & recovery

The ability to continue operating after network or software disruptions. This is a critical feature for reliable user-facing or automated systems. It includes message redelivery, local buffering, and automatic reconnection mechanisms.

The summary of these characteristics provides a foundation for the systematic analysis of existing communication protocols in the following section. Their features will be compared across the predefined metrics within each domain. The objective is to formulate recommendations for selecting the most suitable protocol, based on the specific non-functional requirements and constraints of the domain, including trade-off analysis, hybrid solutions, and reference architectures.

2.12 Evaluation rationale and scoring methodology

The numerical evaluations presented in this study are derived through a qualitative synthesis of technical specifications, protocol standards, and existing empirical research. A five-point scale (1 to 5) is utilized to reflect the relative capability of each protocol to satisfy a specific non-functional requirement, where 5 represents optimal support or performance and 1 indicates significant limitations or lack of inherent support.

To ensure objectivity and address the need for verifiable evidence, the scoring is based on the following criteria:

- Latency and real-time performance: Protocols like MQTT and CoAP are assigned a score of 5 due to their minimal fixed-header overhead and efficient binary encoding, which significantly reduce serialization and transmission delays [1, 5]. HTTP/3 also receives a 5 for latency, as the underlying QUIC

transport mechanism eliminates head-of-line blocking and accelerates connection establishment [2, 4].
- Resource efficiency: A score of 5 is granted to protocols designed for constrained environments, such as CoAP and MQTT, which utilize UDP or lightweight TCP connections to minimize CPU and battery consumption [1,6]. Conversely, distributed streaming platforms like Apache Kafka receive lower scores (e.g., 3) because they require a more robust infrastructure and higher memory overhead to maintain distributed logs [4].
- Reliability and ordering: Protocols that implement mandatory acknowledgment (ACK) and persistent queuing, such as AMQP (RabbitMQ), receive a score of 5 for message ordering and reliability. In contrast, CoAP's reliance on UDP and optional reliability mechanisms results in a lower score (3) for these specific metrics [1, 6].
- Scalability and throughput: Distributed platforms like Apache Kafka and Pulsar are rated 5 because their architecture allows for horizontal scaling and high-volume data ingestion, which is critical for large-scale M2M deployments [3, 4].
- Interoperability: Traditional web protocols such as HTTP/1.1 and HTTP/2 are assigned a score of 5 due to their near-universal support across different platforms and ease of integration with existing web infrastructures [2].

By grounding the evaluation in these technical distinctions and referencing established studies [1,3,5], the methodology provides a transparent and verifiable basis for the protocol-domain mapping and the subsequent proposal of hybrid architectures [8, 10].

3 Comparative Analysis of Communication Protocols Based on Non-Functional Characteristics

Selecting an appropriate communication protocol is essential for the effective implementation of M2M systems, particularly when considering the specific non-functional requirements of different functional domains. Numerous factors influence this choice—from message latency to scalability, reliability, and security of transmission. This section offers a systematic approach to a comparative analysis of leading modern protocols in the context of their applicability across various M2M scenarios.

The following protocols have been selected based on their widespread use, available documentation, and diversity in architectural and functional properties:

- MQTT – popular in IoT due to low latency and low power consumption [1, 5].

Characteristic	MQTT	HTTP/1.1	HTTP/2	HTTP/3	Kafka	Pulsar	AMQP	CoAP
Latency	5	3	4	5	4	4	3	5
Delay Tolerance	5	3	3	3	2	3	3	5
Message Loss Tolerance	4	2	2	2	3	4	4	3
Message Ordering	3	4	4	4	4	4	5	2
Throughput & Payload Size	2	3	4	5	5	5	3	2
Scalability	4	3	3	3	5	5	4	3
Resource Usage / Overhead	5	3	3	3	3	3	3	5
Security	3	4	4	4	3	4	5	2
Reliability	4	3	3	3	5	5	5	3
Interoperability	4	5	5	5	4	4	4	3
Fault Tolerance & Recovery	5	3	4	5	4	4	4	3

Table 2: Evaluation of communication protocols across key non-functional characteristics in M2M environments (scale: 1–5).

- HTTP/1.1, HTTP/2, HTTP/3 – classical web protocols that have evolved with improved performance and security [2].
- Apache Kafka – a distributed streaming platform offering high resilience and throughput [3].
- Apache Pulsar – a modern alternative to Kafka with a distributed architecture and support for multi-threaded clients [4].
- RabbitMQ and AMQP – broker-based systems widely used in enterprise settings for guaranteed message delivery.
- CoAP – a lightweight protocol designed for resource-constrained environments [6].

The evaluation of protocols is performed against the following non-functional characteristics: latency, delay tolerance, message loss and ordering, throughput and payload size, scalability, resource usage, security, reliability, interoperability, and fault tolerance. A five-point scale is used (1 = low performance/support, 5 = high), based on synthesized literature data and comparative experimental studies from both academic and industrial sources.

Figure 2 provides a visual representation of the comparative scores using a heatmap. The color gradient helps identify the strengths and weaknesses of each

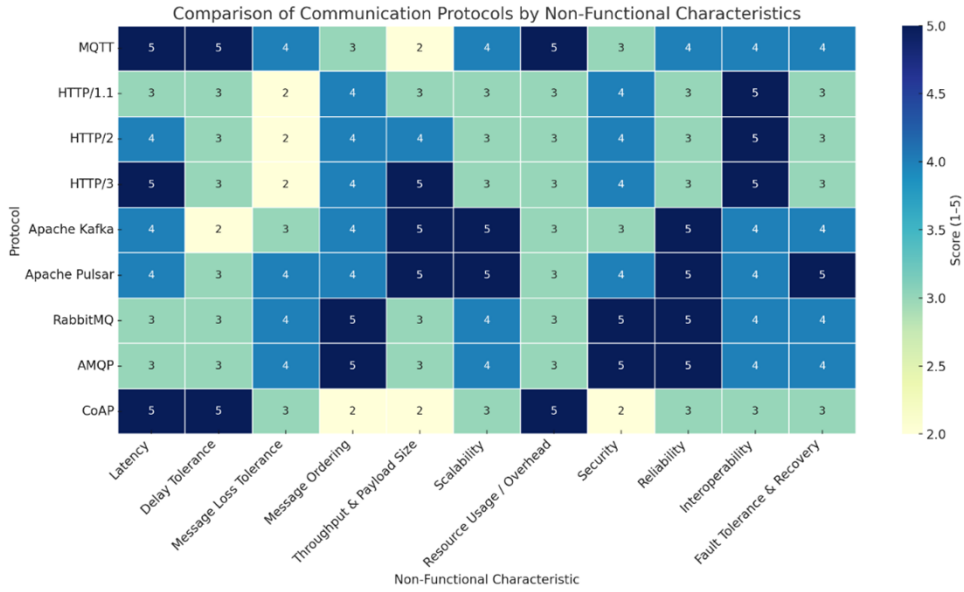

Figure 2: Heatmap – comparison of communication protocols by non-functional characteristics.

protocol at a glance. For instance, Apache Kafka and Pulsar stand out in terms of scalability, reliability, and throughput, while MQTT and CoAP excel in low resource usage and latency.

3.1 Summary analysis by protocol

- MQTT shows superior performance in low-power, low-latency applications and is particularly suitable for constrained devices and critical sensor data [1].
- HTTP/1.1-3 offer strong interoperability and web integration, with HTTP/3 significantly reducing latency through QUIC [2].
- Apache Kafka provides excellent scalability and throughput for real-time streaming but requires a stable and powerful infrastructure [3].
- Apache Pulsar builds on Kafka's architecture and introduces more flexible storage and delivery guarantees, including delayed messages [4].
- RabbitMQ and AMQP remain reliable choices for transactional scenarios, offering guaranteed delivery, message ordering, and fault tolerance [5].
- CoAP offers minimal overhead and is designed for resource-constrained environments where simplicity and low power consumption are essential [6].

The comparative scores and subsequent summary analysis are derived from the systematization of technical characteristics and literature review. These findings serve as a foundation for design-oriented recommendations and should be distinguished from empirical validation through physical measurements. The following conclusions are intended to guide architectural decisions based on the identified non-functional alignment.

3.2 Recommendations by functional domain

- Sensor networks: MQTT is considered well-aligned with the requirements of critical applications, while CoAP is viewed as appropriate for non-critical scenarios. Apache Pulsar is identified as an effective candidate for message aggregation.
- Kiosk systems: RabbitMQ or AMQP are regarded as suitable for ensuring reliable message ordering. HTTP/2 is considered a viable option for front-end interfaces due to its multiplexing capabilities.
- Real-time betting applications: Apache Kafka is seen as highly compatible with high-frequency event streaming requirements. MQTT is considered applicable for lightweight client systems where fast update cycles are prioritized.

3.3 Mapping functional requirements to protocol capabilities

The analysis of non-functional characteristics (Section 2) and the comparative evaluation of protocols (Section 3) reveal significant differences in the needs of specific functional domains and the extent to which current communication protocols can satisfy them. This subsection bridges those perspectives by mapping real-world application needs to protocol capabilities.

- Sensor systems: In sensor-based environments (e.g., telemetry, environmental monitoring, or health tracking), the key requirements include low latency, low resource consumption, and resilience to unreliable network conditions. MQTT stands out due to its lightweight nature and built-in Quality of Service (QoS) features. CoAP is even more lightweight and well-suited for non-critical data under constrained conditions, albeit with trade-offs in ordering and security. In more complex deployments, Apache Pulsar can act as a central aggregation hub with buffering and delivery management.
- Kiosk and transactional systems: For applications like ticketing or financial kiosks, key non-functional needs include high reliability, security, strict message ordering, and fault recovery. RabbitMQ and AMQP support broker-based architectures with acknowledgments and redelivery queues. HTTP/2 provides

a convenient channel for secure web communication, but additional logic is needed for reliability guarantees. A hybrid deployment that separates interface and backend processing is often preferred.
- Real-time odds streaming: Systems that ingest live betting odds require ultra-low latency, high throughput, and interoperability with external providers. Apache Kafka and Pulsar are strong candidates due to their support for large-scale, real-time event handling. MQTT can be leveraged for lightweight edge clients, where high delivery speed is more important than guaranteed ordering. Protocols such as AMQP, although reliable, are generally too slow for real-time requirements.

This mapping confirms that no single protocol universally satisfies all M2M communication scenarios. Optimal implementations are usually the result of trade-offs between latency, reliability, resource constraints, and scalability. These findings justify the exploration of hybrid architectures where the communication stack is tailored to specific functional requirements, allowing for a balance between latency, reliability, and resource efficiency.

The presented analysis highlights that no protocol offers a one-size-fits-all solution for M2M applications. Aligning non-functional characteristics with the specific functional domain is essential. In future work, we will empirically validate the performance of the leading protocols in real or simulated M2M environments.

4 Hybrid architecture: combining protocols according to domain and non-functional requirements

Selecting a single communication protocol that satisfies the full spectrum of non-functional requirements across all M2M application domains is rarely feasible. As demonstrated in the previous sections, protocols differ significantly in their ability to support low latency, message ordering, scalability, reliability, and minimal resource consumption. A practical and increasingly adopted strategy is to design hybrid architectures that combine the strengths of different protocols depending on the context. This section presents such architectures, supported by contemporary scientific literature.

4.1 Scenario 1: Sensor network with varying criticality

An environmental monitoring system composed of both non-critical sensors (e.g., temperature, humidity) and highly critical sensors (e.g., vital signs in medical settings).

Hybrid architecture:

- CoAP for non-critical sensors: minimal overhead, low energy consumption, multicast capabilities.
- MQTT with QoS 2 for critical sensors: guaranteed delivery and resilience to connection loss.
- Apache Pulsar as a central broker for buffering and analytical integration.

This architecture aligns with the model proposed by [7], where hybrid MAC protocols achieve higher efficiency through a combination of contention-based and scheduled access. Similarly, separating traffic by criticality at the transport layer improves reliability and resource utilization.

4.2 Scenario 2: Ticket kiosk with web interface and transactional backend

A ticket vending kiosk that provides a web-based interface and requires secure communication, message ordering, and durable transaction processing. Hybrid architecture:

- HTTP/2 with TLS for the front-end interface: secure communication, multiplexing, and web compatibility.
- AMQP or RabbitMQ for backend logic: guaranteed delivery, ordered message queues, and recovery mechanisms.
- Redis or Kafka as a buffer for temporary storage in case of failure.

This model reflects the approach advocated by [9], where hierarchical access and prioritization enable effective handling of heterogeneous message traffic. RabbitMQ is commonly used in systems requiring transactional reliability and durability.

4.3 Scenario 3: Real-time sports betting odds aggregation

A system that aggregates betting odds from multiple external sources and distributes them to user interfaces and analytics platforms. Hybrid architecture:

- MQTT or WebSocket for ingesting odds from external providers.
- Apache Kafka for streaming and high-throughput event consumption.
- Apache Pulsar for prioritization, delayed delivery, and long-term retention.

In latency-sensitive and high-frequency environments, streaming platforms such as Kafka and Pulsar ensure scalability and reliability. Studies by [1] and [3] support the application of such models in real-time IoT scenarios.

4.4 Benefits and challenges of hybrid architectures

Benefits:

- Context-driven optimization of communication behavior.
- Increased architectural flexibility and modularity.
- Improved fault tolerance and latency management.

Challenges:

- Increased complexity in deployment and integration.
- Higher demand for advanced monitoring and orchestration.
- Potential interoperability risks across protocol layers.

Hybrid architectures represent a promising design paradigm for M2M systems operating in heterogeneous environments. By tailoring the communication stack to specific functional requirements, better alignment with latency, reliability, and efficiency objectives can be achieved. Future research will involve empirical validation of the proposed models through simulations and real-world measurements, following methodologies such as those of [7, 9] and aligned with industry best practices.

5 Conclusion

This paper presented a systematic evaluation of communication protocols and data stream management systems in M2M environments, with a focus on non-functional characteristics and functional domains. Based on well-defined and justified metrics, drawn from scientific and industrial sources, we compared widely used protocols such as MQTT, HTTP/1.1-3, Kafka, Pulsar, RabbitMQ, AMQP, and CoAP.

The comparative analysis clearly demonstrates that no single protocol can meet all requirements on its own, especially in the context of heterogeneous functional scenarios—from sensor networks and kiosk terminals to real-time streaming applications. This underscores the need for developing hybrid architectures in which different technologies are strategically combined to address specific non-functional requirements, based on domain, environment, and data criticality.

Within the proposed methodology, we formulated hypothetical models for communication architecture whose advantages and limitations were supported by literature and logical analysis. This approach enables more flexible and efficient M2M system design, particularly in contexts with dynamic or conflicting requirements.

Empirical validation of the proposed classification and hybrid architectural models will be the subject of a follow-up publication, based on both simulation and real-world measurements. That work will apply quantitative metrics such as average values and 99th percentiles for latency, message loss, performance, and resilience. Additionally, we plan to explore the possibility of constructing an automated decision-support model for protocol selection, based on specified functional requirements and constraints.

References

[1] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, C. K.-Y. Tan, Performance evaluation of MQTT and CoAP via a common middleware, *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Singapore, 2014.

[2] I. Lee, K. Lee, The Internet of Things (IoT): Applications, investments, and challenges for enterprises, *Business Horizons*, 58:431–440, 2015.

[3] K. Mekki, E. Bajic, F. Chaxel, F. Meyer, A comparative study of LPWAN technologies for large-scale IoT deployment, *ICT Express*, 5:1–7, 2019.

[4] S. Tallberg, *A comparison of data ingestion platforms in real-time stream processing pipelines*, Master thesis, Mälardalen University, Sweden, 2020.

[5] P. Thota, Y. Kim, Implementation and Comparison of M2M Protocols for Internet of Things, *2016 4th Intl Conf on Applied Computing and Information Technology (ACIT-CSII-BCD)*, Las Vegas, USA, 2016.

[6] Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), Internet Engineering Task Force (IETF), RFC 7252, 2014.

[7] Y. Liu, C. Yuen, J. Chen, X. Cao, A scalable Hybrid MAC protocol for massive M2M networks, *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, 2013.

[8] B. Alojaiman, A Multi-Criteria Decision-Making Process for the Selection of an Efficient and Reliable IoT Application, *Processes*, 11:1313, 2023.

[9] Y. Liu, C. Yuen, X. Cao, N. Ul Hassan, J. Chen, Design of a Scalable Hybrid MAC Protocol for Heterogeneous M2M Networks, *IEEE Internet of Things Journal*, 1:99–111, 2014.

[10] O. A. Khashan, N. M. Khafajah, Efficient hybrid centralized and blockchain-based authentication architecture for heterogeneous IoT systems, *Journal of King Saud University – Computer and Information Sciences*, 35:726–739, 2023.