International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.134, August 2026

Experimental AHP-TOPSIS Evaluation of Communication Technologies for Real-Time M2M Data Streaming Systems

Oleg Iliev
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
Acad. Georgi Bonchev Str., Block 8
Sofia 1113, Bulgaria

ABSTRACT

This paper presents an experimental evaluation of communication technologies for real-time Machine-to-Machine (M2M) data streaming systems based on an integrated Analytic Hierarchy Process (AHP) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) decision model. Four communication technologies - HTTP/1.1, HTTP/2, RabbitMQ, and Apache Kafka - were evaluated under controlled LAN conditions using a sustained stress-test workload of up to 50,000 generated messages per second. The evaluation considered six criteria: average latency, 99th-percentile latency, message loss, ordering consistency, transaction success, and throughput. The results indicate that Apache Kafka achieves the highest overall suitability with a TOPSIS closeness coefficient of $C_i = 0.946$, followed by RabbitMQ ($C_i = 0.777$), HTTP/2 ($C_i = 0.607$), and HTTP/1.1 ($C_i = 0.000$). The findings demonstrate that multi-criteria decision models provide a practical framework for objective communication technology selection in real-time M2M data streaming systems by simultaneously considering latency, reliability, throughput, and domain-specific operational requirements.

Keywords

Machine-to-Machine (M2M), Real-Time Data Streaming, Communication Technology Selection, Multi-Criteria Decision Making (MCDM), Analytic Hierarchy Process (AHP), Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)

1. INTRODUCTION

Machine-to-Machine (M2M) systems increasingly rely on continuous data streaming infrastructures to support real-time information exchange between distributed devices, gateways, and backend processing platforms. Such systems are widely deployed in industrial automation, smart buildings, intelligent transportation, energy management, and large-scale event-driven applications. As the volume and frequency of generated events continue to grow, communication infrastructures are required to provide low latency, high throughput, reliable delivery, and predictable behavior under varying workload conditions [4, 9].

The selection of communication technologies for real-time M2M environments has become a complex engineering problem. Modern distributed systems may employ a variety of communication approaches, including traditional request–response protocols such as HTTP/1.1 and HTTP/2, as well as message-oriented middleware platforms including RabbitMQ and Apache Kafka. Each technology provides different trade-offs regarding latency, throughput, scalability, fault tolerance, message durability, and operational complexity [1, 8, 2]. Consequently, identifying the most appropriate communication solution cannot be reduced to the evaluation of a single performance metric.

The challenge becomes even more significant in environments where existing network infrastructure imposes practical constraints on technology deployment. Many industrial and enterprise systems operate in heterogeneous environments that combine wired and wireless communication channels, legacy fieldbus technologies, cloud services, and modern event-streaming platforms. In such scenarios, communication architectures are often required to satisfy conflicting operational requirements. The optimal solution therefore depends not only on the characteristics of individual communication technologies but also on the interaction between technologies, deployment infrastructure, and domain-specific requirements.

Several studies have proposed methodologies for evaluating communication protocols in M2M and Internet of Things (IoT) environments. Existing approaches range from qualitative assessments and protocol-specific benchmarks to formal multi-criteria decision-making frameworks [5, 6]. Recent research demonstrates that multi-criteria methodologies provide a more systematic and reproducible basis for technology selection than purely performance-oriented comparisons. In particular, the Analytic Hierarchy Process (AHP) and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) have been successfully applied to technology evaluation problems involving multiple conflicting criteria [7, 3].

Despite these advances, many published evaluations remain focused on individual protocols or isolated deployment scenarios. Limited attention has been devoted to the combined evaluation of communication technologies and communication architectures under realistic workload conditions. Furthermore, existing studies frequently prioritize average latency while providing insufficient consideration of throughput, message reliability, delivery ordering, and domain-specific operational constraints.

This study addresses these limitations through an experimental evaluation of communication technologies and communication ar-

chitectures for real-time M2M data streaming systems. The evaluation considers HTTP/1.1, HTTP/2, RabbitMQ, and Apache Kafka under controlled high-volume workloads representative of transactional event-streaming environments. An integrated AHP–TOPSIS decision model is employed to combine empirical performance measurements with domain-specific importance weights. The objective is not to identify a universally optimal technology, but rather to demonstrate a structured methodology for selecting the most appropriate communication solution according to the requirements and constraints of a particular application domain.
The obtained results provide empirical evidence regarding the strengths and limitations of different communication approaches and demonstrate the applicability of multi-criteria decision-making techniques for supporting technology selection in real-world M2M deployments.

2. RELATED WORK

The problem of selecting communication technologies for Machine-to-Machine (M2M) and Internet of Things (IoT) systems has attracted significant research interest due to the increasing complexity of distributed data processing environments. Modern M2M systems frequently operate under strict requirements regarding latency, reliability, scalability, and fault tolerance, while simultaneously facing deployment constraints imposed by existing network infrastructure and operational environments [4, 9].
Early studies in this domain primarily focused on the technical characteristics of individual communication protocols. Particular attention was devoted to the evaluation of lightweight protocols designed for constrained devices and low-bandwidth environments. As IoT deployments expanded, protocol selection became increasingly dependent on application-specific requirements rather than on protocol specifications alone. Consequently, researchers began investigating systematic approaches for comparing alternative communication solutions across multiple performance dimensions.
Message-oriented middleware technologies have emerged as important components of modern distributed systems due to their ability to decouple message producers and consumers while providing improved scalability and reliability. Contemporary middleware platforms support asynchronous communication models that enable efficient handling of high-volume event streams. Comparative studies have demonstrated that middleware-based solutions offer significant advantages over traditional request–response communication models in scenarios involving large numbers of concurrent events and distributed processing components [1].
Among the most widely adopted communication technologies, HTTP-based protocols remain attractive due to their simplicity, broad interoperability, and extensive support across software platforms. The introduction of HTTP/2 introduced several performance improvements, including multiplexing, header compression, and more efficient connection utilization, resulting in lower communication overhead compared with HTTP/1.1 [8]. Nevertheless, request–response communication models continue to exhibit limitations when applied to large-scale event-driven systems requiring persistent message delivery and high throughput.
Event-streaming platforms such as Apache Kafka and message-oriented middleware systems such as RabbitMQ have received considerable attention as alternatives to traditional communication approaches. These technologies provide features including asynchronous processing, message durability, horizontal scalability, and fault tolerance. Comparative evaluations have shown that Kafka typically excels in high-throughput streaming scenarios, whereas RabbitMQ often provides advantages in routing flexibility and operational simplicity [2]. The selection between these technologies therefore depends heavily on workload characteristics and operational requirements.
The growing diversity of communication technologies has stimulated research into formal technology selection methodologies. Traditional protocol comparisons based solely on latency or throughput measurements have gradually been replaced by multi-criteria decision-making approaches that incorporate both technical and non-technical factors. Such approaches enable systematic evaluation of competing alternatives while accounting for conflicting requirements and domain-specific priorities.
A comprehensive overview of communication protocol selection methodologies has recently been presented through a systematic literature review and multidimensional taxonomy of protocol selection approaches in M2M and IoT environments [6]. The review demonstrates a transition from heuristic and experience-based technology selection toward structured decision-support frameworks capable of incorporating contextual information, deployment constraints, and multiple evaluation criteria.
Among the available multi-criteria decision-making techniques, the Analytic Hierarchy Process (AHP) and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) have become particularly popular due to their transparency, computational efficiency, and practical applicability. AHP provides a structured mechanism for determining criterion importance through pairwise comparisons, while TOPSIS enables the ranking of alternatives according to their relative proximity to an ideal solution [7, 3]. These methods have been successfully applied in numerous engineering decision problems, including technology selection, network design, resource allocation, and infrastructure planning.
Previous work introduced a domain-oriented methodology for selecting communication protocols in M2M systems and subsequently extended the approach through an integrated AHP–TOPSIS framework capable of incorporating multiple technical and operational criteria [4, 5]. However, most existing studies focus either on protocol-level comparisons or on theoretical evaluation frameworks. Limited attention has been devoted to the experimental validation of technology selection models using realistic high-volume workloads and communication architectures.
The present study extends existing research by combining empirical performance evaluation with multi-criteria decision analysis. Unlike previous investigations that focus on individual protocols or isolated deployment scenarios, the proposed approach evaluates both communication technologies and communication architectures under realistic real-time data streaming conditions. The objective is to provide a reproducible framework for selecting communication solutions that best satisfy the operational requirements of a particular M2M application domain.

3. AHP–TOPSIS EVALUATION FRAMEWORK

Selecting communication technologies for real-time M2M systems requires balancing multiple conflicting criteria that cannot be represented by a single performance indicator. Technologies providing minimal latency may exhibit lower scalability, whereas solutions optimized for throughput may introduce additional processing overhead. Consequently, a structured multi-criteria decision-making framework is required for objective technology evaluation.
This study employs an integrated Analytic Hierarchy Process (AHP) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) framework. AHP determines the relative importance of the evaluation criteria, while TOPSIS ranks the alternative

communication technologies using experimentally measured performance indicators.

3.1 Decision Criteria

Six quantitative criteria were selected to characterize the performance of real-time M2M data streaming systems:

—Average end-to-end latency (L_{avg});
—99th percentile latency (L_{p99});
—Message loss rate ($Loss$);
—Ordering consistency ($Ordering$);
—Successful transaction processing rate ($TxnSuccess$);
—Throughput ($Tput$).

Average latency reflects overall system responsiveness, whereas the 99th-percentile latency captures infrequent but operationally significant delays. Message loss represents communication reliability, while throughput characterizes the capability to process large event volumes.
Ordering consistency measures the percentage of messages received in their original sequence, an important property for event-driven processing. The successful transaction processing rate represents the percentage of messages that are successfully received, processed, and acknowledged. Together, these criteria characterize the correctness and reliability of communication beyond simple message delivery.
The selected criteria were derived from previous communication protocol evaluation studies and from the operational requirements of real-time M2M streaming environments [4, 5, 6].

3.2 AHP Weight Determination

The Analytic Hierarchy Process (AHP) determines the relative importance of the evaluation criteria using pairwise comparisons represented by the matrix

$$A = a_{11}a_{12}\dots a_{1n}a_{21}a_{22}\dots a_{2n}\vdots\ddots\vdots a_{n1}a_{n2}\dots a_{nn} \quad (1)$$

where a_{ij} denotes the relative importance of criterion i with respect to criterion j.
The resulting priority vector is

$$W = (w_1, w_2, \dots, w_n) \quad (2)$$

where w_i is the normalized weight of criterion i.
For real-time data streaming systems, the adopted weight vector is

$$W = (w_{L_{avg}} = 0.25,\ w_{L_{p99}} = 0.15,\ w_{Loss} = 0.15, w_{Ordering} = 0.10,\ w_{TxnSuccess} = 0.15,\ w_{Tput} = 0.20) \quad (3)$$

This weighting scheme emphasizes latency while simultaneously accounting for reliability and scalability requirements.

3.3 TOPSIS Ranking Method

Following weight determination, TOPSIS ranks the evaluated communication technologies.
The normalized decision matrix is obtained using vector normalization:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \quad (4)$$

where x_{ij} is the measured value of criterion j for alternative i.
The weighted normalized matrix is then computed as

$$v_{ij} = w_j r_{ij} \quad (5)$$

where w_j denotes the corresponding criterion weight.
The ideal (A^+) and anti-ideal (A^-) solutions are defined as

$$A^+ = \{\max(v_{ij}) \mid j \in J_{benefit}, \min(v_{ij}) \mid j \in J_{cost}\} \quad (6)$$

$$A^- = \{\min(v_{ij}) \mid j \in J_{benefit}, \max(v_{ij}) \mid j \in J_{cost}\} \quad (7)$$

Latency and message loss are treated as cost criteria, whereas ordering consistency, transaction success, and throughput are considered benefit criteria.
The Euclidean distances from the ideal and anti-ideal solutions are

$$S_i^+ = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^+)^2} \quad (8)$$

$$S_i^- = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^-)^2} \quad (9)$$

The relative closeness coefficient is finally calculated as

$$C_i = \frac{S_i^-}{S_i^+ + S_i^-} \quad (10)$$

where larger values indicate alternatives closer to the ideal solution.

3.4 Evaluation Procedure

The proposed evaluation procedure consists of the following steps:

(1) Experimental performance measurements under controlled workloads;
(2) Construction of the decision matrix;
(3) Determination of criterion weights using AHP;
(4) Normalization of the performance indicators;
(5) Computation of the weighted normalized matrix;
(6) Identification of the ideal and anti-ideal solutions;
(7) Calculation of TOPSIS closeness coefficients and final ranking.

The resulting ranking provides an objective basis for selecting communication technologies according to the operational requirements of real-time M2M application domains.

4. EXPERIMENTAL ENVIRONMENT

4.1 Domain Requirements and Experimental Motivation

Real-time Machine-to-Machine (M2M) data streaming systems are widely deployed in industrial automation, smart buildings, intelligent transportation, telemetry, and large-scale event-driven applications. These systems require communication infrastructures capable of delivering high-frequency event streams with low latency,

reliable message delivery, preserved message ordering, and predictable behavior under sustained workloads.
The experimental evaluation represents a typical real-time streaming environment in which large numbers of updates are distributed simultaneously to multiple consumers. Such applications share several common requirements:

—low communication latency;
—predictable response times under heavy load;
—reliable message delivery;
—preservation of event ordering;
—scalability under increasing workloads;
—compatibility with existing network infrastructure.

Although latency is often the primary evaluation metric, practical deployments are also influenced by infrastructure availability, deployment complexity, operational constraints, and technology interoperability. Consequently, the communication technology with the lowest latency is not necessarily the most suitable solution for a given application domain.
The objective of the experimental evaluation is therefore to validate the proposed AHP–TOPSIS decision framework by identifying the communication technology that best satisfies the operational requirements of real-time M2M streaming systems.

4.2 Communication Technologies Under Evaluation

Four communication technologies representing different architectural approaches were selected for evaluation.

(1) HTTP/1.1 implements the traditional request–response communication model and remains widely used because of its simplicity and interoperability. However, synchronous communication and connection management introduce additional protocol overhead.
(2) HTTP/2 extends HTTP/1.1 through multiplexing, header compression, and improved connection utilization, reducing communication overhead while preserving compatibility with existing web infrastructures [8].
(3) RabbitMQ is a message-oriented middleware platform implementing AMQP. It provides asynchronous messaging, message persistence, flexible routing, and reliable delivery, making it suitable for enterprise integration and event-driven applications.
(4) Apache Kafka is a distributed event-streaming platform optimized for high-throughput data processing, horizontal scalability, durability, and fault tolerance in large-scale streaming environments [2].

Together, these technologies provide representative alternatives for comparative evaluation.

4.3 Experimental Architecture

The experimental platform consists of two processing nodes connected through a Gigabit Ethernet LAN. The first node generates event streams through an Odds Feed Push API, while the second node receives and processes the transmitted messages.
The application logic remains identical throughout all experiments; only the communication technology is varied. Consequently, observed performance differences originate from the communication layer rather than application-specific behavior.
InfluxDB is used for time-series storage of performance metrics, while Grafana provides visualization and monitoring.

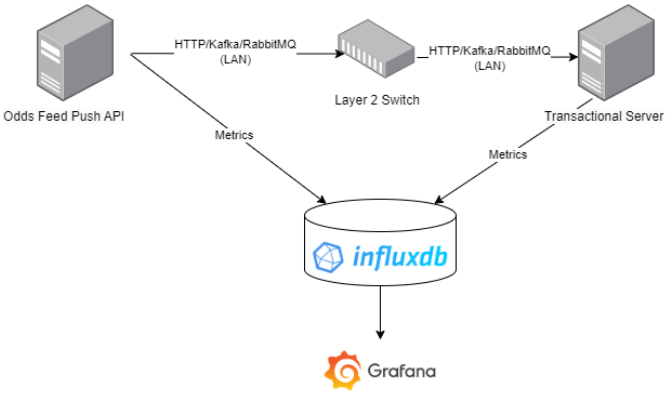

Fig. 1. Experimental architecture used for evaluating communication technologies in a real-time M2M data streaming environment.

Figure 1 illustrates the experimental architecture.
The experiments were performed under controlled LAN conditions in order to minimize external network variability and isolate protocol-level performance characteristics.

4.4 Workload Generation

A software workload generator simulates continuous event streams instead of physical sensor devices, ensuring repeatability and precise control of experimental parameters.
Each generated message contains a unique identifier, sequence number, and timestamp, enabling end-to-end traceability throughout the communication pipeline.
The workload generator supports configurable message generation rates, concurrent producers, payload sizes, and experiment durations, allowing evaluation under varying communication intensities.

4.5 Performance Metrics

Six quantitative performance indicators were measured:

—Average end-to-end latency (L_{avg});
—99th-percentile latency (L_{p99});
—Message loss rate ($Loss$);
—Ordering consistency ($Ordering$);
—Successful transaction processing rate ($TxnSuccess$);
—Throughput ($Tput$).

These metrics correspond directly to the evaluation criteria defined in the AHP–TOPSIS framework presented in Section 3.

4.6 Experimental Scenarios

Although three workload profiles were designed, this paper reports only the sustained stress-test scenario because it exposes communication bottlenecks most clearly under high-volume operating conditions.

—Normal Operation – continuous event generation under typical workloads.
—Peak Load – short-duration bursts with rapidly increasing message rates.
—Sustained Stress Test – prolonged high-volume traffic for evaluating scalability, latency stability, and throughput.

Table 1. Empirical performance measurements obtained during the sustained stress-test scenario.

Tech.	Avg.	P99	Loss	Order	Success	Tput
HTTP/1.1	224.5	445.2	1.12	98.6	98.7	3950
HTTP/2	128.4	264.3	0.58	98.9	99.1	7650
RabbitMQ	135.3	250.6	0.30	99.2	99.6	9450
Apache Kafka	140.7	230.4	0.18	99.7	99.8	12800

Avg. = average end-to-end latency (ms); P99 = 99th-percentile latency (ms); Loss = message loss rate (%); Order = ordering consistency (%); Success = successful transaction processing rate (%); Tput = throughput (messages/s).

The combination of representative communication technologies, controlled workloads, and a common experimental architecture provides a consistent basis for comparative evaluation using the proposed AHP–TOPSIS methodology.

5. EXPERIMENTAL RESULTS

This section presents the empirical performance measurements obtained from the comparative evaluation of HTTP/1.1, HTTP/2, RabbitMQ, and Apache Kafka under sustained real-time M2M data streaming workloads. These measurements constitute the input data for the AHP–TOPSIS evaluation framework presented in Section 3.

5.1 Empirical Performance Measurements

Table 1 summarizes the measured performance indicators obtained during the sustained stress-test scenario. The experiments were conducted under controlled LAN conditions using workloads of up to 50 000 generated messages per second.

The empirical measurements reveal clear performance differences among the evaluated communication technologies. HTTP/1.1 consistently exhibits the highest communication latency and the lowest throughput due to its synchronous request–response model and repeated connection management. HTTP/2 reduces latency through multiplexing and persistent connections but remains constrained by its client–server communication paradigm. In contrast, RabbitMQ and Apache Kafka benefit from asynchronous message handling, resulting in substantially higher throughput and improved delivery reliability under increasing workload. These observations indicate that communication architecture has a direct influence on overall system performance beyond the characteristics of the underlying transport protocol.

The measurements also indicate that each evaluated technology exhibits distinct performance advantages depending on the considered criterion. HTTP/2 provides the lowest average latency, whereas Apache Kafka achieves the highest throughput together with the lowest message loss and strongest ordering consistency. RabbitMQ offers the most balanced performance across all criteria, while HTTP/1.1 consistently ranks lowest under sustained workloads.

5.2 Latency Analysis

Latency is a key performance indicator for real-time data streaming applications because delayed updates reduce the operational value of transmitted information.

Figure 2 compares the average and 99th-percentile latency values. HTTP/2 achieves the lowest average latency (128.4 ms), whereas Apache Kafka provides the lowest 99th-percentile latency (230.4 ms), indicating superior stability under sustained load. RabbitMQ also demonstrates competitive latency performance, while

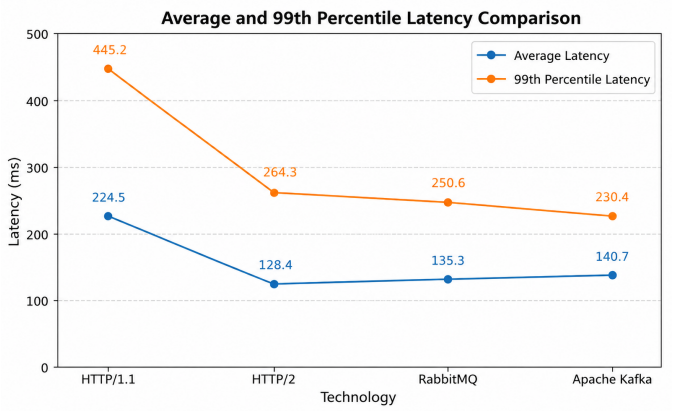

Fig. 2. Average and 99th-percentile latency comparison.

HTTP/1.1 exhibits considerably higher latency because of synchronous request–response communication overhead.

These results indicate that average latency alone is insufficient for characterizing communication performance. Tail-latency measurements provide additional insight into worst-case communication delays.

The observed latency behaviour reflects the communication models employed by the evaluated technologies. While HTTP-based protocols require synchronous request processing before each response is returned, broker-based systems decouple message production from message consumption. Consequently, RabbitMQ and Apache Kafka maintain more stable latency under increasing workload despite the additional middleware layer. This behaviour becomes particularly evident under sustained high-volume workloads, where communication efficiency and scalability become increasingly important.

5.3 Reliability and Ordering Analysis

Reliable message delivery and preservation of event ordering are essential for maintaining consistency in event-driven systems.

Figures 3 and 4 summarize the measured message loss and ordering consistency.

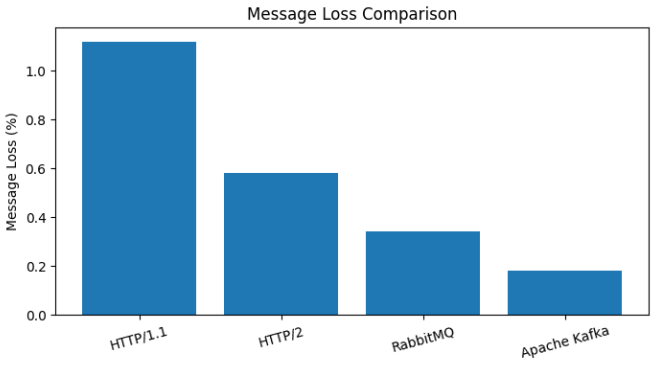

Fig. 3. Message loss comparison.

Message reliability is particularly important in real-time M2M applications where missing events may lead to incomplete system state information or delayed control actions. The experiments

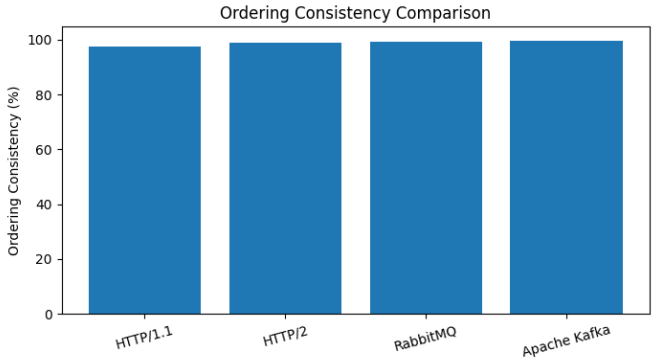

Fig. 4. Ordering consistency comparison.

demonstrate that persistent message storage and acknowledgement mechanisms implemented by broker-based platforms significantly reduce message loss compared to synchronous HTTP communication, especially under elevated communication load.

Although all evaluated technologies preserve message ordering under normal operating conditions, differences become visible under higher traffic intensity. Broker-based platforms provide stronger ordering guarantees through queue- and partition-based processing mechanisms, whereas HTTP-based communication relies primarily on sequential request execution. The observed behaviour confirms that ordering performance depends not only on the protocol itself but also on the underlying communication architecture.

Apache Kafka achieves both the lowest message loss (0.18%) and the highest ordering consistency (99.7%). RabbitMQ also provides strong reliability characteristics, whereas HTTP-based communication exhibits increased message loss under sustained workloads.

5.4 Throughput Analysis

Figure 5 compares the throughput achieved by the evaluated technologies.

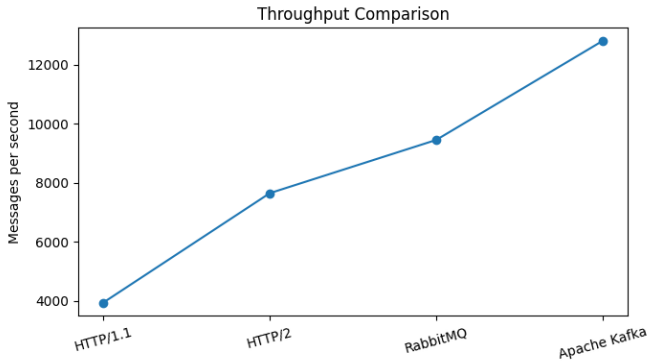

Fig. 5. Throughput comparison.

Apache Kafka achieves the highest throughput (12 800 msg/s), followed by RabbitMQ (9 450 msg/s), HTTP/2 (7 650 msg/s), and HTTP/1.1 (3 950 msg/s).

Throughput measurements clearly demonstrate the scalability advantages of asynchronous communication platforms. Apache Kafka consistently achieves the highest throughput owing to its distributed log architecture and sequential disk access model, while RabbitMQ follows closely through efficient message queuing mechanisms. HTTP-based approaches remain suitable for moderate communication rates but exhibit reduced scalability as message frequency increases.

Table 2. Normalized decision matrix.

Tech.	L_{avg}	L_{p99}	Loss	Order	Success	Tput
HTTP/1.1	0.6933	0.7178	0.8562	0.4970	0.4972	0.2184
HTTP/2	0.3960	0.4271	0.4427	0.4985	0.4992	0.4228
RabbitMQ	0.4179	0.4043	0.2290	0.5000	0.5017	0.5223
Kafka	0.4345	0.3716	0.1375	0.5025	0.5027	0.7078

Table 3. Weighted decision matrix.

Tech.	v_{avg}	v_{p99}	v_{Loss}	v_{Ord}	v_{Succ}	v_{Tput}
HTTP/1.1	0.1733	0.1077	0.1284	0.0497	0.0746	0.0437
HTTP/2	0.0990	0.0641	0.0664	0.0499	0.0749	0.0846
RabbitMQ	0.1045	0.0609	0.0343	0.0501	0.0752	0.1044
Kafka	0.1085	0.0558	0.0207	0.0503	0.0754	0.1416

5.5 Summary of Experimental Findings

The experimental results demonstrate that communication technology selection cannot rely on a single performance indicator. While HTTP/2 offers favorable latency characteristics, broker-based technologies provide substantially better reliability, ordering consistency, and throughput. These empirical measurements constitute the quantitative basis for the AHP–TOPSIS evaluation presented in the following section. Since no technology consistently dominates across all evaluated performance criteria, a structured multi-criteria decision-making procedure is required to determine the most suitable overall solution. The following AHP–TOPSIS analysis integrates these competing performance characteristics into a single quantitative ranking.

6. AHP–TOPSIS ANALYSIS

This section applies the integrated AHP–TOPSIS framework to the empirical measurements presented in Section 5. The objective is to transform the measured performance indicators into a quantitative ranking of the evaluated communication technologies.

6.1 Decision Matrix Construction

The empirical measurements in Table 1 form the initial decision matrix for the TOPSIS analysis. The criteria weights were determined using AHP, as described in Section 3. The decision matrix was normalized using the procedure defined in Section 3, and the resulting values were multiplied by the AHP-derived criterion weights to obtain the weighted decision matrix.

6.2 Normalized Decision Matrix

6.3 Weighted Decision Matrix

6.4 Ideal and Anti-Ideal Solutions

6.5 Closeness Coefficients and Ranking

The TOPSIS ranking results are summarized in Table 5. The closeness coefficient C_i expresses the relative proximity of each alternative to the ideal solution; higher values indicate greater suitability for the considered real-time M2M streaming environment.

Table 4. Ideal and anti-ideal solutions.

Criterion	Ideal (A^+)	Anti-Ideal (A^-)
L_{avg}	0.0990	0.1733
L_{p99}	0.0558	0.1077
Loss	0.0207	0.1284
Ordering	0.0503	0.0497
$TxnSuccess$	0.0754	0.0746
Tput	0.1416	0.0437

Table 5. TOPSIS distance and ranking results.

Tech.	S_i^+	S_i^-	C_i	Rank
Kafka	0.009	0.168	0.946	1
RabbitMQ	0.040	0.140	0.777	2
HTTP/2	0.074	0.114	0.607	3
HTTP/1.1	0.171	0.000	0.000	4

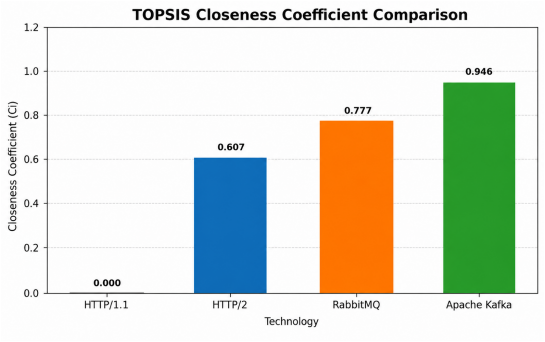

Fig. 6. TOPSIS closeness coefficient comparison.

The TOPSIS analysis integrates all measured performance indicators together with the AHP-derived criterion weights into a single suitability score for each communication technology.
Figure 6 visualizes the closeness coefficients obtained for the evaluated technologies.
The results confirm that Apache Kafka achieves the highest closeness coefficient ($C_i = 0.946$), indicating the strongest overall performance across the evaluated criteria. RabbitMQ ranks second ($C_i = 0.777$), followed by HTTP/2 ($C_i = 0.607$). HTTP/1.1 obtains the lowest ranking due to its comparatively high latency,higher message loss, lower throughput, and reduced reliability under sustained load conditions.

6.6 Comparative Evaluation and Practical Recommendations

The obtained TOPSIS ranking provides quantitative decision support for selecting communication technologies according to the operational requirements of real-time M2M data streaming systems. Although Apache Kafka achieves the highest overall score, the experimental results indicate that each evaluated technology remains appropriate for specific deployment scenarios.
HTTP/2 is particularly suitable for latency-sensitive applications requiring synchronous request–response communication while maintaining compatibility with existing web infrastructures. RabbitMQ provides a balanced compromise between throughput, reliability, and deployment complexity, making it appropriate for enterprise integration and moderate-scale event processing. Apache Kafka is the preferred solution for large-scale streaming applications where throughput, scalability, and reliable event delivery represent the primary operational requirements. HTTP/1.1 remains applicable for lightweight systems with limited communication intensity despite its lower overall performance.
Consequently, the proposed AHP–TOPSIS methodology should not be interpreted as identifying a universally optimal communication technology. Instead, it provides a structured framework for selecting the technology that best satisfies the priorities and constraints of a particular M2M application domain.
Table 6 summarizes the recommended deployment scenarios derived from the obtained experimental results and the corresponding TOPSIS ranking.

7. DISCUSSION

The presented results highlight that communication technology selection in real-time M2M systems is inherently a multi-dimensional engineering decision rather than a simple comparison of protocol performance. Although latency is frequently considered the dominant performance indicator, the experiments demonstrate that communication reliability, throughput, message ordering, and successful transaction processing collectively determine the operational effectiveness of a communication infrastructure.
The experimental results also demonstrate that communication technology evaluation based on a single performance metric may lead to misleading conclusions. For example, HTTP/2 achieves the lowest average latency, whereas Apache Kafka dominates in throughput, reliability, and ordering consistency. The integrated AHP–TOPSIS framework resolves these conflicting observations by simultaneously considering all evaluation criteria according to their domain-specific importance.
Apache Kafka achieved the highest overall TOPSIS ranking primarily because its architecture is optimized for continuous high-volume event streaming. Its distributed log architecture, persistent storage model, and partition-based processing provide excellent scalability while maintaining reliable message delivery and ordering consistency. These characteristics explain why Kafka remains the preferred choice for large-scale telemetry platforms, industrial monitoring systems, and real-time event processing infrastructures.
RabbitMQ demonstrated consistently balanced performance across all evaluated criteria. Although its maximum throughput is lower than Kafka's, RabbitMQ offers greater routing flexibility, mature AMQP support, and comparatively simpler deployment in enterprise environments. Consequently, RabbitMQ may represent the more appropriate solution in systems where integration flexibility and operational simplicity outweigh the need for maximum throughput.
The comparison between HTTP/1.1 and HTTP/2 confirms the importance of modern transport optimizations. HTTP/2 significantly reduces communication overhead through multiplexing and header compression, producing noticeable improvements in both latency and throughput. Nevertheless, the experiments also illustrate that request–response communication models remain fundamentally less suitable than asynchronous messaging platforms for sustained real-time event streaming.
An important implication of this study is that communication technology selection should not focus exclusively on protocol characteristics. Real-world deployments are constrained by existing infrastructure, operational policies, implementation cost, and integration complexity. In many industrial systems, introducing a technically superior technology may not be economically or operationally justified. Therefore, the objective of technology selection is often to identify the solution that provides the best balance between perfor-

Table 6. Comparative summary and practical recommendations for the evaluated communication technologies.

Technology	Main Strength	Main Limitation	Typical Application
HTTP/1.1	Simplicity and universal compatibility	High latency and limited scalability	Legacy web services and lightweight client–server applications
HTTP/2	Low latency and improved communication efficiency	Limited scalability for sustained high-volume streaming	REST APIs, latency-sensitive web services, and moderate real-time systems
RabbitMQ	Balanced performance and reliable message delivery	Lower maximum throughput than Apache Kafka	Enterprise messaging, distributed applications, and event-driven systems
Apache Kafka	High throughput, scalability, and reliability	Higher deployment and operational complexity	Large-scale event streaming, telemetry platforms, and industrial monitoring

mance and deployment constraints rather than the highest benchmark score.

The proposed AHP–TOPSIS methodology addresses this challenge by combining objective experimental measurements with domain-specific priorities. Unlike traditional benchmarking approaches that compare technologies using a single metric, the proposed framework enables decision makers to adjust criterion weights according to the requirements of different application domains. For example, latency-sensitive industrial control systems, large-scale telemetry platforms, and cloud-native event processing architectures may all produce different optimal rankings despite evaluating the same communication technologies.

Although the present evaluation considered four widely adopted communication technologies under controlled LAN conditions, the proposed methodology is not limited to these technologies. The same decision framework can be applied to evaluate hybrid communication architectures, cloud-native messaging platforms, edge-computing deployments, or geographically distributed systems, provided that appropriate evaluation criteria and experimental measurements are available.

Overall, the study demonstrates that combining empirical experimentation with structured multi-criteria decision-making provides a practical and reproducible framework for communication technology selection. The methodology supports engineering decisions that extend beyond protocol benchmarking and better reflect the complexity of modern real-time M2M environments.

8. CONCLUSION

This paper presented an experimental evaluation of communication technologies for real-time Machine-to-Machine (M2M) data streaming systems using an integrated AHP–TOPSIS decision-support framework. Unlike conventional protocol benchmarking studies that compare technologies primarily through isolated performance indicators, the proposed approach combines empirical measurements with structured multi-criteria decision-making to support objective communication technology selection based on domain-specific operational requirements.

The experimental evaluation demonstrated clear performance differences among HTTP/1.1, HTTP/2, RabbitMQ, and Apache Kafka under sustained high-volume workloads. Apache Kafka achieved the highest overall ranking owing to its combination of high throughput, low message loss, strong ordering consistency, and predictable latency behaviour. RabbitMQ also demonstrated competitive performance, while HTTP/2 provided significant improvements over HTTP/1.1 but remained less suitable than asynchronous broker-based communication for continuous event streaming.

Beyond the experimental comparison itself, the principal contribution of this work is the proposed decision-support methodology. The integration of empirical performance measurements with the AHP–TOPSIS framework enables multiple performance criteria to be evaluated simultaneously while explicitly accounting for domain-specific priorities. Consequently, communication technology selection is transformed from a subjective engineering decision into a transparent, reproducible, and quantitatively supported evaluation process.

The presented methodology is not limited to selecting individual communication technologies. It can also be applied to evaluate alternative communication architectures, including hybrid solutions that combine multiple communication mechanisms while taking deployment constraints, existing infrastructure, and operational objectives into account. This capability considerably extends the applicability of the framework beyond traditional protocol benchmarking and reflects the practical complexity of contemporary M2M deployments. Although the present study focused on four widely adopted communication technologies operating under controlled LAN conditions, the methodology itself is technology-independent. Future research will investigate additional communication platforms, cloud-native messaging systems, edge-computing environments, geographically distributed deployments, and adaptive criterion weighting mechanisms capable of reflecting dynamically changing operational priorities.

Overall, the obtained results demonstrate that communication technology selection should be approached as a domain-oriented multi-criteria optimization problem rather than as a comparison of individual protocol performance metrics. The proposed framework provides a practical, extensible, and reproducible methodology that can support communication technology and architecture selection across a broad range of real-time M2M applications.

9. REFERENCES

[1] Wael Al-Manasrah et al. Message-oriented middleware systems: Technology overview. *arXiv preprint arXiv:2602.17774*, 2026.

[2] Philippe Dobbelaere and Kyumars Sheykh Esmaili. Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations. In *Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems*, pages 227–238, 2017.

[3] Ching-Lai Hwang and Kwangsun Yoon. *Multiple Attribute Decision Making: Methods and Applications – A State-of-the-Art Survey*. Springer, Berlin, 2012.

[4] Oleg Iliev. Methodology for selecting communication protocols in m2m systems. *Serdica Journal of Computing*, 19(1):1–15, 2025.

[5] Oleg Iliev. Multi-criteria methodology for selecting communication protocols in m2m environments. *International Journal of Advanced Computer Science and Applications*, 17(2), 2026.

[6] Oleg Iliev. Systematic review and taxonomy of communication protocol selection methodologies in m2m and iot systems. *International Journal For Multidisciplinary Research*, 2026.

[7] Thomas L. Saaty, Kirti Peniwati, and Jen S. Shang. The analytic hierarchy process and human resource allocation: Half the story. *Mathematical and Computer Modelling*, 46(7–8):1041–1053, 2007.

[8] Martin Thomson and Mark Benfield. Hypertext transfer protocol version 2 (http/2). RFC 7540, Internet Engineering Task Force (IETF), 2015.

[9] Sridhar Raj Sankara Vadivel, V. Karthikeyan, and K. Gopalakrishnan. Introduction to the internet of things (iot). In *Internet of Things Security*, pages 3–31. Elsevier, 2026.