NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 7 Issue 4, Pg. No. 103-106, AIJCST-V7I4P109, 2025
https://doi.org/10.63282/3117-5481/AIJCST-V7I4P109

Original Article

Reliable Message Delivery in Distributed Edge-Cloud Systems: A Comprehensive Survey

***Manojava Bharadwaj Bhagavathula**

Independent Researcher, USA.

Abstract:

The reliable delivery of messages from edge devices to distributed cloud infrastructure is a foundational requirement for modern industrial, automotive, and consumer IoT ap- plications. Edge environments are characterized by intermittent connectivity, bandwidth constraints, and resource limitations, which make standard reliable transport protocols like TCP insufficient for application-level data integrity. This paper presents a comprehensive survey of the protocols, architectural patterns, and state-of-the-art mechanisms used to ensure reliable message delivery in distributed edge-cloud systems. I analyze key protocols including MQTT, AMQP, CoAP, and QUIC, evaluating their reliability mechanisms such as Quality of Service (QoS) levels, persistent sessions, and stream multiplexing. Furthermore, I survey architectural patterns such as Store-and-Forward, application-level acknowledg-ments, and idempotency mechanisms that operate above the transport layer. Through a comparative analysis, I identify the trade-offs between throughput, latency, and delivery guarantees, providing a decision framework for system architects. Finally, I discuss emerg-ing trends in AI-driven connectivity management and decentralized consensus for message integrity.

Keywords:

Edge Computing, Reliable Messaging, Distributed Systems, MQTT, AMQP, Store-And-Forward, Iot, Network Resilience.

Article History:

Received: 01.06.2025

Revised: 04.07.2025

Accepted: 15.07.2025

Published: 26.07.2025

1. Introduction

The proliferation of edge computing has shifted data generation from centralized datacenters to distributed edge locations—factories, vehicles, and remote installations. While this shift reduces latency for local processing, it creates a significant challenge for data aggregation: reliably delivering critical telemetry and event data to the cloud overunreliable networks (Shi et al., 2016).

1.1. Motivation

In mission-critical systems such as industrial automation or healthcare monitoring, message loss is not an option. A missing sensor reading could indicate a missed failure precursor; a lost command could delay safety-critical actuation. However, edge networks often rely on cellular (LTE/5G), satellite, or Wi-Fi connections that are prone to packet loss, high jitter, and extended outages.

1.2. Problem Statement

Standard transport layers like TCP provide reliability only while a connection is active. They do not handle:

- Connection Persistence: Queuing messages during extended outages (hours or days).
- Application Semantics: Ensuring messages are processed exactly once, not just deliv-ered.
- Resource Constraints: Buffering reliably on devices with limited flash storage and RAM.

Thus, reliability must be engineered at the application and protocol layers.39

1.3. Scope and Contributions

This survey contributes:

1. A taxonomy of reliability challenges in edge-cloud systems.
2. A detailed analysis of standard protocols (MQTT, AMQP, Kafka, QUIC) regarding their reliability guarantees.
3. A catalog of architectural reliability patterns (Store-and-Forward, Outbox Pattern).
4. A comparative evaluation framework for selecting appropriate mechanisms.

2. Key Challenges in Edge-Cloud Messaging

2.1. Network Unreliability and Partitioning

Edge devices frequently experience "flapping" connections or total partitions. Systems must distinguish between transient failures (retriable immediately) and long-term outages (requiring disk buffering). The CAP theorem implies that during these partitions, systems must choose between consistency (strong ordering) and availability (accepting new local writes).

2.2. Resource Constraints

Edge gateways often run on ARM-based hardware with limited RAM (e.g., 2-8 GB) and flash storage. Infinite in-memory buffering is impossible. Writing to disk (Store-and-Forward) introduces I/O bottlenecks and flash wear concerns. Reliability mechanisms must be lightweight.

2.2. Throughput vs. Reliability Trade-offs

Mechanisms like synchronous acknowledgments (ACKs) and transactions reduce throughput significantly. Achieving high throughput (e.g., for video analytics or vibration data) while guaranteeing delivery requires batching, compression, and asynchronous reliability patterns.

2.3. Data Consistency and Ordering

Ensuring First-In-First-Out (FIFO) processing is difficult when utilizing multi-path routing or parallel ingestion consumers. Duplicate delivery (At-Least-Once) is common during retries; handling this requires idempotent consumers.

3. Survey of Communication Protocols

3.1. MQTT (Message Queuing Telemetry Transport)

MQTT is the dominant protocol for IoT connectivity due to its lightweight header (2 bytes) and publish-subscribe model (Mishra & Kertesz, 2020).

- Reliability: Offers three QoS levels. QoS 1 (At-least-once) and QoS 2 (Exactly-once) utilize persistent storage on both client and broker to ensure delivery across reconnects (Thangavel et al., 2014).
- Session Persistence: The CleanSession=false flag allows the broker to queue mes-sages for a disconnected client.
- Limitation: QoS 2 has high latency overhead due to the 4-way handshake.

3.2. AMQP (Advanced Message Queuing Protocol)

AMQP 1.0 is a binary wire protocol focused on enterprise reliability (OASIS, 2012).

- Reliability: Supports strong transaction guarantees and flexible routing. Messages can be settled (ACKed) independently of transfer.
- Flow Control: Built-in link credit system manages backpressure natively.
- Suitability: Excellent for edge gateways aggregating data, but heavy for constrained sensors.

3.3. CoAP (Constrained Application Protocol)

Designed for UDP-based constrained networks (Shelby et al., 2014; Thangavel et al., 2014).

- Reliability: Uses "Confirmable" (CON) messages to implement a lightweight ACK layer over UDP.
- Suitability: Best for extremely low-power networks (LoRaWAN, NB-IoT) where TCP overhead is prohibitive.

3.4. Kafka / Pulsar (Log-Based Streaming)

While primarily data center technologies, lightweight adaptors allow edge deployment (Wang et al., 2021).

- Reliability: Durability is achieved via disk logs. Consumers track their own offsets, allowing replayability.
- Pattern: Edge devices often use a lightweight forwarder (e.g., vector, fluentd) to buffer locally and produce to a central Kafka cluster.

3.5. QUIC and HTTP/3

QUIC improves upon TCP by operating over UDP (Iyengar & Thomson, 2021; Contr-eras & Baliosian, 2018).

- Reliability: Eliminates Head-of-Line (HoL) blocking. Loss of one packet in a stream doesn't stall other streams.
- Suitability: Ideal for unreliable cellular networks where connection migration (switch-ing between Wi-Fi and LTE without re-handshake) is valuable.

4. Architectural Reliability Patterns

4.1. Store-and-Forward

The most fundamental pattern for edge reliability (Bonomi et al., 2012; Gupta & Singh, 2019). Messages are written to local persistent storage (SQLite, embedded key-value store, or disk queue) before transmission is attempted.

- Edge Queue: A local circular buffer ensures new data overwrites oldest data only when storage is full, preserving the most critical recent context or oldest critical context depending on policy.
- Forwarder: A separate thread reads from the queue and sends to the cloud, deleting local copies only upon receiving an application-level ACK.

4.2. Application-Level Acknowledgments

Transport ACKs (TCP ACK) only prove data reached the OS kernel. Application ACKs prove the cloud service processed (or persisted) the data.

- Mechanism: The edge device retains the message until it receives a specific HTTP 200 OK or MQTT PUBACK packet containing the message ID.

4.3. *Idempotency and Deduplication*

To support "Exactly-Once" processing with "At-Least-Once" delivery infrastructure:

- Unique IDs: Every message is assigned a UUID or monotonic sequence number at the source (edge).
- Cloud De-duplication: The cloud ingestion layer caches seen IDs (e.g., in Redis) and discards duplicates, ensuring downstream analytics see accurate data counts (Li et al., 2021).

4.4. Dual-Path Routing

For critical alarms, devices may utilize multiple network interfaces simultaneously (e.g., LTE and Satellite).

- Pattern: Send message clones over all interfaces. The cloud accepts the first arrival and discards the rest.
- Cost: Increases data usage but minimizes latency and maximizes delivery probability.

5. Comparative Analysis

Table 1 summarizes the trade-offs between protocols.

Table 1. Protocol Comparison for Reliable Edge Messaging

Protocol	Reliability	Overhead	Persistence	Edge Fit
MQTT	QoS 1/2	Low	Session	High
AMQP	Transactions	High	Queue	Medium

CoAP	CON Msgs	Very Low	None	High (Sensor)
HTTP/REST	App Retry	Medium	None	Low
Kafka	Log Commit	High	Disk Log	Medium (Gateway)
QUIC	Stream	Medium	None	High (Mobile)

6. Future Directions

6.1. AI-Driven Connectivity Management

Future edge agents will use ML models to predict connectivity outages based on geolocation (e.g., knowing a vehicle is entering a tunnel) and preemptively buffer data or aggressively upload critical packets (Chen et al., 2019).

6.2. Decentralized Consensus

Blockchain and DAG (Directed Acyclic Graph) technologies are being explored to guarantee message integrity and ordering across decentralized edge nodes without a single central authority, useful for multi-party supply chains (Corsaro & Orsini, 2021; Fernandez et al., 2021).

7. Conclusion

Reliable message delivery in distributed edge-cloud systems requires a multi-layered approach. While protocols like MQTT and QUIC provide transport optimization, true reliability is achieved through architectural patterns: local persistence (Store-and-Forward), application-level acknowledgments, and rigorous idempotency handling. Architects must balance the strictness of delivery guarantees (QoS) against the latency and storage costs inherent in constrained edge environments.

Acknowledgments:

I acknowledge the use of Google's Gemini 3 Pro language model to assist with English grammar and clarity edits. The model was used only for language polishing; all technical content, claims, and conclusions remain my own responsibility.

References

[1] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646.

[2] OASIS. (2019). *MQTT Version 5.0*. OASIS Standard.

[3] OASIS. (2012). *Advanced Message Queuing Protocol (AMQP) Version 1.0*.

[4] Shelby, Z., Hartke, K., & Bormann, C. (2014). The Constrained Application Protocol (CoAP) (RFC 7252).

[5] Iyengar, J., & Thomson, M. (2021). *QUIC: A UDP-Based Multiplexed and Secure Transport* (RFC 9000).

[6] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proc. MCC Workshop on Mobile Cloud Computing*.

[7] Mishra, B., & Kertesz, A. (2020). The use of MQTT in M2M and IoT systems: A survey. *IEEE Access*, *8*, 201071–201086.

[8] Thangavel, D., Ma, X., Valera, A., Tan, H. X., & Tan, C. K. (2014). Performance analysis of MQTT and CoAP protocols in constrained wireless networks. In *Proc. IEEE Sensors* (pp. 167–170).

[9] Wang, J., Wu, G., Lin, X., & Chang, M. (2021). A Kafka-based real-time data acquisition and processing system for industrial Internet of Things. *IEEE Access*, *9*, 156321–156333.

[10] Gupta, S. K., & Singh, P. K. (2019). Store and forward mechanism for reliable data delivery in IoT. In *Proc. ICCCS* (pp. 45–52).

[11] Contreras, L. M., & Baliosian, S. (2018). QUIC as a potential enabler for 5G low latency services. In *Proc. IEEE Conference on Standards for Communications and Networking*.

[12] Corsaro, A., & Orsini, O. H. M. (2021). Zenoh: Zero overhead pub/sub, store/query and computations. In *Proc. IEEE International Conference on Edge Computing*.

[13] Li, Y., Chen, W., Wang, Z., & Liu, Y. (2021). Data redundancy elimination in edge storage systems for IoT. *IEEE Transactions on Parallel and Distributed Systems*, *32*(1), 123–136.

[14] Fernandez, T. M., Alchieri, A., & Bessani, A. D. (2021). A survey on consensus protocols for edge computing. *ACM Computing Surveys*, *54*(2), 1–36.

[15] Chen, M., Challita, U., Saad, W., Yin, C., & Debbah, M. (2019). Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, *21*(4), 3039–3071.