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 applications. Edge environments are characterised by intermittent connectivity, bandwidth constraints, and resource limitations, making standard reliable transport protocols like TCP insufficient for application-level data integrity. This paper presents a comprehensive survey of protocols, architectural patterns, and state-of-the-art mechanisms for ensuring reliable message delivery in distributed edge-cloud systems. The article analyse key protocols, including MQTT, AMQP, CoAP, and QUIC, and evaluates their reliability mechanisms, such as Quality of Service (QoS) levels, persistent sessions, and stream multiplexing. Furthermore, the paper surveys the architectural patterns such as Store-and-Forward, application-level acknowledgements, and idempotency mechanisms that operate above the transport layer. Through a comparative analysis, the trade-offs identified among throughput, latency, and delivery guarantees provide a decision framework for system architects. Finally, emerging trends discussed 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.

I. 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 over unreliable networks (Shi et al., 2016).

A. 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.

B. 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 delivered.
- Resource Constraints: Buffering reliably on devices with limited flash storage and RAM.

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

C. 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 catalogue of architectural reliability patterns (Store-and-Forward, Outbox Pattern).
4. A comparative evaluation framework for selecting appropriate mechanisms.

II. KEY CHALLENGES IN EDGE-CLOUD MESSAGING

A. 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).

B. 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.

C. 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.

D. 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.

III. SURVEY OF COMMUNICATION PROTOCOLS

A. MQTT (Message Queuing Telemetry Transport)

MQTT is the dominant protocol for IoT connectivity due to its lightweight header (2bytes) 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 messages for a disconnected client.   76
- Limitation: QoS 2 has high latency overhead due to the 4-way handshake.

B. 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.

C. 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.

D. 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.

E. QUIC and HTTP/3

QUIC improves upon TCP by operating over UDP (Iyengar & Thomson, 2021; Contreras & 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 (switching between Wi-Fi and LTE without re-handshake) is valuable.

IV. ARCHITECTURAL RELIABILITY PATTERNS

A. *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 the oldest data only when storage is full, preserving the most critical recent context or the 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.

B. *Application-Level Acknowledgments.*
Transport ACKs (TCP ACK) only prove that the 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.

C. *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 Deduplication: 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).

D. *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.

V. 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)

VI. FUTURE DIRECTIONS

A. *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).

B. *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; Fernandezet al., 2021).

VII. 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 acknowledgements, and rigorous idempotency handling. Architects must balance the strictness of delivery guarantees (QoS) against the latency and storage costs inherent in constrained edge environments.

VIII. REFERENCES

1. Weisong Shi et al., "Edge Computing: Vision and Challenges, *IEEE Internet of Things Journal*, vol. *3*, no. 5, pp. 637 - 646, 2016. Google Scholar | Publisher Link

2. OASIS, *MQTT Version 5.0*. OASIS Standard, 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

3. OASIS, Advanced Message Queuing Protocol (AMQP) Version 1.0, 2012. [Online]. Available: https://www.oasis-open.org/standard/amqp/

4. Z. Shelby, K.Hartke, and C. Bormann, The Constrained Application Protocol (CoAP) (RFC 7252), Internet Engineering Task Force (IETF), 2014. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7252

5. J. Iyengar, and M. Thomson, *QUIC: A UDP-Based Multiplexed and Secure Transport* (RFC 9000), Internet Engineering Task Force (IETF), 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9000

6. Flavio Bonomi et al., "Fog Computing and its Role in the Internet of Things," *MCC '12: Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13-16, 2012. Google Scholar | Publisher Link

7. Biswajeeban Mishra, and Attila Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," *IEEE Access*, vol. 8, pp. 201071–201086, 2020. Google Scholar | Publisher Link

8. Dinesh Thangavel et al., "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, pp. 1-6, 2014. Publisher Link

9. J. Wang, "A Kafka-based real-time data acquisition and processing system for industrial Internet of Things," *IEEE Access*, vol. 9, pp. 156321–156333, 2021.

10. S.K. Gupta, and P.K.Singh, "Store and forward mechanism for reliable data delivery in IoT," *Proc. ICCCS,* pp. 45–52, 2019.

11. L. M. Contreras, and S. Baliosian, "QUIC as a Potential Enabler for 5G Low Latency Services," *Proc. IEEE Conference on Standards for Communications and Networking*, 2018.

12. Angelo Corsaro, and Orsini O. H. M, "Zenoh: Zero overhead pub/sub, store/query and computations," In *Proc. IEEE International Conference on Edge Computing,* 2021.

13. Li, Y et al., "Data Redundancy Elimination in Edge Storage Systems for IoT," *IEEE Transactions on Parallel and Distributed Systems*, vol. *32*, no. 1, pp. 123–136, 2021.

14. T.M. Fernandez, A. Alchieri, and A.D. Bessani, "A Survey on Consensus Protocols for Edge Computing," *ACM Computing Surveys*, vol. *54*, no. 2, pp. 1–36, 2021.

15. Mingzhe Chen et al., "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. *21*, no. 4, pp. 3039–3071, 2019. Google Scholar | Publisher Link