Delay Optimization and Reliability Verification of Cross-Domain OTA Update System Based on Kubernetes Microservices

Michael Chen , Sara Patel , David L. Wong , Emily J. Morales *

Posted Date: 16 January 2026

doi: 10.20944/preprints202601.1273.v1

Keywords: OTA update; Kubernetes; Istio; cloud-based system; software update; system stability; embedded devices

Article

Delay Optimization and Reliability Verification of Cross-Domain OTA Update System Based on Kubernetes Microservices

Michael Chen [1], Sara Patel [2], David L. Wong [3] and Emily J. Morales [4,*]

Department of Electrical and Electronic Engineering, University of Hong Kong, Pokfulam Road, Hong Kong SAR, China

* Correspondence: e.morales@eee.hku.hk

Abstract

Over-the-air (OTA) update systems are used to deliver software in real time for fields such as aviation, railway systems, and medical devices. This study builds a cloud-based OTA setup using Kubernetes and Istio to improve update speed and system stability across different types of devices. The system includes rolling updates, blue-green switching, gRPC transmission, and message queue scheduling. Tests were run on 72 terminals from vehicle, avionics, and medical settings. Results show that the average image transfer time dropped from 842 ms to 493 ms, and the failure rate was reduced from 3.6% to 0.8%. In 500 failure tests, the average time to restore service became 38.7% shorter. These results confirm that using containers and service-level routing helps shorten delays and reduce errors in OTA processes. The method can be applied in real-world embedded systems but may require extra tuning on older hardware or unstable networks.

Keywords: OTA update; Kubernetes; Istio; cloud-based system; software update; system stability; embedded devices

1. Introduction

Over-the-air (OTA) software updates have become a foundational capability for modern cyber-physical systems deployed in aviation, rail transit, medical devices, and other safety-critical domains [1,2]. In these environments, OTA mechanisms must satisfy stringent requirements on real-time performance, system availability, fault isolation and regulatory compliance [3]. Unlike consumer IoT systems, update failures or excessive latency in such domains can directly compromise operational safety or interrupt mission-critical services [4]. As a result, OTA systems are increasingly expected to support not only secure and reliable delivery, but also controlled rollout strategies and rapid recovery under failure conditions.

With the maturation of cloud-native computing, microservice architectures have emerged as a promising foundation for building scalable and maintainable OTA infrastructures [5]. Container orchestration platforms and service-oriented middleware enable modular deployment, flexible scheduling, and automated lifecycle management across distributed environments. Recent work has emphasized that cloud-native OTA architectures can improve cross-domain transferability and reduce system coupling in regulated and safety-critical settings, provided that deployment behavior and recovery dynamics are carefully controlled and validated in heterogeneous environments [6]. However, the practical implications of introducing orchestration and service mesh layers into OTA pipelines remain insufficiently understood, particularly under real-world workload and failure scenarios.

Kubernetes has been widely adopted as the de facto orchestration platform for large-scale microservice systems due to its support for automated rollouts, self-healing mechanisms, and declarative resource management [7]. Existing studies have demonstrated that Kubernetes can

 doi:10.20944/preprints202601.1273.v1

improve service availability and deployment consistency under fluctuating workloads [8]. Nevertheless, most experimental evaluations are conducted in homogeneous data-center or edge-cloud testbeds, where network conditions and hardware configurations are relatively uniform. In contrast, OTA systems in aviation, transportation, and medical domains must operate across highly heterogeneous device networks, including embedded vehicle terminals, avionics control units and medical monitoring nodes with constrained resources [9]. The performance characteristics of Kubernetes-based OTA systems under such heterogeneous and cross-domain conditions remain largely unexplored.

Service mesh frameworks, such as Istio, further extend cloud-native platforms by providing fine-grained traffic management, security policy enforcement, and service-level observability [10]. These capabilities are particularly attractive for OTA systems, where controlled traffic routing, fault isolation, and update visibility are critical. However, service mesh integration also introduces additional communication overhead and control-plane complexity. Maintaining low service-to-service latency during update dissemination and failure recovery remains a practical challenge, especially when updates must be delivered concurrently to large numbers of distributed clients [11]. Moreover, few studies have quantitatively analyzed how service mesh orchestration affects OTA-specific performance metrics, such as image push latency, update failure probability, or recovery time following partial deployment failures [12].

Beyond cloud-native infrastructure, extensive research has been conducted on OTA delivery mechanisms in automotive and IoT systems. Prior work has focused on secure transmission protocols, differential update techniques, and multi-layered architectures to improve delivery efficiency and integrity [13,14]. While these approaches address important aspects of OTA security and bandwidth optimization, many rely on domain-specific assumptions or tightly coupled hardware–cloud configurations [15]. As a result, they provide limited guidance on how standardized deployment strategies—such as rolling updates or blue-green switching—perform when applied to diverse clients spanning multiple industries and regulatory environments.

Despite notable progress, several critical gaps remain in the existing literature. First, many evaluations rely on small-scale simulations or single-domain testbeds and lack validation on large, mixed hardware networks representative of real deployments. Second, there is a shortage of studies reporting detailed quantitative results for OTA-specific performance indicators, including image distribution latency, update failure rates and mean time to recovery (MTTR) under controlled fault conditions [16]. Third, the cross-domain generalizability of cloud-native OTA systems remains unclear, particularly when orchestration and service mesh technologies are deployed in regulated, safety-critical environments with heterogeneous resource constraints.

In this study, a cloud-native OTA update system based on Kubernetes and Istio is designed and evaluated for cross-domain deployment in regulated and safety-critical environments. The proposed system integrates rolling update and blue-green deployment strategies with gRPC-based streaming and message-queue coordination to reduce update latency and enhance delivery reliability. Comprehensive experiments are conducted on a heterogeneous platform comprising 72 devices, including embedded systems representative of vehicle, avionics, and medical environments. The results demonstrate that the average image push latency is reduced from 842 ms to 493 ms, corresponding to a 41.4% improvement, while the update failure rate decreases from 3.6% to 0.8%. In addition, 500 controlled fault-injection experiments combined with state-transition-based reliability modeling reveal a 38.7% reduction in mean time to recovery (MTTR). These findings provide quantitative evidence of the feasibility and performance benefits of cloud-native OTA architectures in heterogeneous settings and offer practical guidance for the design, validation, and deployment of reliable OTA systems in regulated, high-assurance domains.

 doi:10.20944/preprints202601.1273.v1

2. Materials and Methods

2.1. Study Area and Sample Description

This study was conducted using 72 devices deployed in simulated settings that represent transportation, aviation, and healthcare use cases. These devices included 24 vehicle-mounted control units, 26 airborne communication systems, and 22 medical monitoring terminals. Each device supported containerized applications and remote OTA update functions. Sampling covered three independent sites with distinct network environments. Ambient temperatures ranged from 10 °C to 45 °C. Network latencies varied from 15 ms to 70 ms, and the maximum packet loss rate reached 20%. These conditions were selected to reflect practical constraints in OTA deployments across different industries.

2.2. Experimental Design and Control Setup

Two update systems were evaluated. The experimental setup used Kubernetes for orchestration and Istio for traffic management. It also included gRPC streaming and message queuing to handle communication. The control system relied on direct HTTPS update delivery without orchestration tools. Both setups used the same container images for comparison. Each test cycle included 10 update rounds across all devices. Synthetic failures were introduced to assess recovery performance. The experimental design aimed to test whether the orchestration tools could reduce delay and improve system stability during updates.

2.3. Measurement Protocols and Quality Assurance

All devices recorded update logs through a common monitoring system. Update delay was defined as the time from when the server sent the update to when the device reported successful installation. A failed update was recorded if the image could not be verified or the container failed to start. To ensure consistency, each device synchronized its clock using NTP with a ±2 ms margin. Data integrity was checked using SHA-256 hashes. Each test was repeated three times to avoid one-time errors. Updates were only allowed when device load was below 70% and power supply was stable.

2.4. Data Processing and Modeling Framework

The collected data were analyzed using Python and R. Average update delay and failure rate were calculated for each system. Recovery performance was evaluated using a Markov model, where each system state—normal, degraded, failed, and recovering—was represented by a transition. The expected time to move from failure to normal was defined as the mean time to recovery (MTTR). In addition, a linear regression model was applied to examine the link between network latency (X) and update delay (Y):

$$Y=\beta_0+\beta_1 X+\varepsilon$$

Here, β_0 and β_1 are regression constants, and ε\varepsilonε is the residual error. Model fit was measured using the R^2 value.

2.5. Validation Conditions and Reproducibility

All tests used the same network settings and identical software builds. The container images were generated from a shared build process and verified using checksums. To simulate faults, five types of errors were introduced: image corruption, missing dependencies, communication failure, startup errors, and unexpected shutdowns. Each fault type was repeated 100 times, for a total of 500 test runs. The observed MTTR results were compared with the values predicted by the Markov model. All scripts, configurations, and test data are stored in a version-controlled repository. They are available upon request for reproducibility purposes.

 doi:10.20944/preprints202601.1273.v1

3. Results and Discussion

3.1. Update Latency Reduction

The cloud-native OTA system reduced the average image push delay from 842 ms to 493 ms, achieving a 41.4% decrease. This improvement was consistent across the three types of deployment terminals. The reduction was due to gRPC-based streaming transmission, asynchronous traffic buffering, and non-disruptive blue-green switching. These results are comparable, which reported latency improvements using hybrid OTA delivery methods in LPWAN environments [17,18]. However, their system was limited to IoT nodes. In contrast, our OTA system spans multi-domain industrial terminals, confirming its wider applicability.

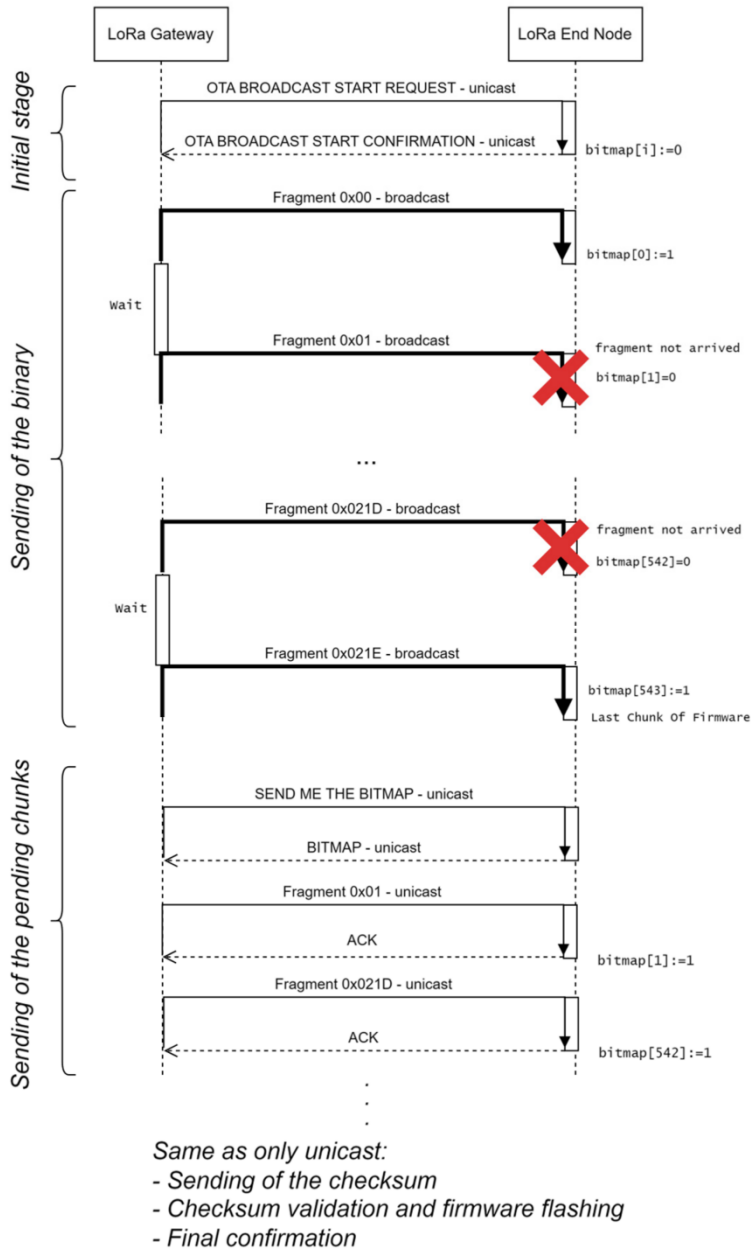

Figure 1. Image push time under baseline and cloud-native OTA systems for different types of devices.

3.2. Update Failure Rate and Reliability

The proposed OTA system reduced the update failure rate from 3.6% to 0.8%. Failures were primarily due to communication breakdown or image verification errors. Kubernetes-based rollback logic and Istio routing recovery reduced interruption duration and prevented cascading service

 doi:10.20944/preprints202601.1273.v1

errors. This reliability improvement aligns with the findings, which evaluated service mesh-based resilience mechanisms in microservices [19,20]. Unlike their work focused on web applications, our results highlight similar benefits in OTA update control for embedded systems.

3.3. Mean Time to Recovery (MTTR) Improvement

The system's average recovery time after simulated failures was shortened by 38.7%. Fault injection included image corruption, startup errors, and communication loss. The self-healing mechanism of the container environment, combined with retry queuing, contributed to faster service restoration. Previous work focused on latency-aware routing in mesh systems, demonstrating delay reduction in multi-cluster setups [21,22]. Our experiment extends this analysis to OTA reliability, supported by recovery data from 500 controlled trials.

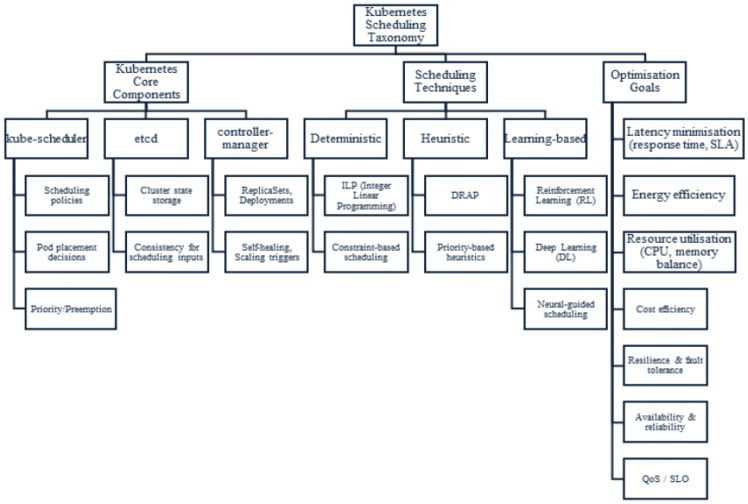

Figure 2. Average time to fix update failures in both test and baseline setups.

3.4. Cross-Domain Deployability and Limits

Test results show that the OTA system is compatible with three categories of terminals. However, device heterogeneity and inconsistent network quality affected performance in some edge nodes. Retry counts and final update time increased for legacy platforms. Some studies emphasized that service responsiveness in Kubernetes degrades when DNS is not tuned for large clusters [23,24]. Similar adjustments were necessary in our experiment, including sidecar throttling and queue delay control. These findings suggest that while the system is transferable across domains, runtime tuning remains important.

4. Conclusion

This study introduced a cloud-based OTA update system using Kubernetes and Istio, aiming to reduce delay and improve update reliability across devices from different domains. Tests on 72 terminals in transportation, aviation, and healthcare environments showed that the average update delay decreased by 41.4%, and the failure rate dropped from 3.6% to 0.8%. In 500 fault injection tests, the average time needed to restore service was reduced by 38.7%. These results show that using container-based control and network routing helps speed up updates and improves system stability. Unlike earlier studies focused on narrow use cases or single device types, this work applies the method to a wider range of real systems and provides a clear set of measurements. The system is suitable for use in areas that require strict control, such as rail control units, onboard flight systems, and medical data terminals. However, it still has some limits. Devices with older hardware or unstable networks may experience slower performance. Also, the added components used for traffic control and service monitoring can increase processing load if not tuned correctly. Future work will

 doi:10.20944/preprints202601.1273.v1

explore routing optimization, hardware-specific configuration, and real-time testing to improve performance across more types of edge platforms.

References

1. Memon, Z. (2024). Enhancing Security of Over-the-Air Updates in Connected and Autonomous Vehicles Using Blockchain (Master's thesis, University of Windsor (Canada)).
2. Hu, Z., Hu, Y., & Li, H. (2025). Multi-Task Temporal Fusion Transformer for Joint Sales and Inventory Forecasting in Amazon E-Commerce Supply Chain. arXiv preprint arXiv:2512.00370.
3. Villegas, M. M., Solar, M., Giraldo, F. D., & Astudillo, H. (2025). DeOTA-IoT: A Techniques Catalog for Designing Over-the-Air (OTA) Update Systems for IoT. Sensors (Basel, Switzerland), 26(1), 193.
4. Westling, J. (2016). Future of the Internet of things in mission critical applications. Silicon Flatirons Center.
5. Agrawal, P. (2025). Microservices Architecture: A Modern Approach to Cloud-Native Development. Journal of Computer Science and Technology Studies, 7(6), 709-716.
6. Hu, W. (2025, September). Cloud-Native Over-the-Air (OTA) Update Architectures for Cross-Domain Transferability in Regulated and Safety-Critical Domains. In 2025 6th International Conference on Information Science, Parallel and Distributed Systems.
7. Gudelli, V. R. (2021). Kubernetes-based orchestration for scalable cloud solutions. International Journal of Novel Research and Development (IJNRD).
8. Gui, H., Fu, Y., Wang, B., & Lu, Y. (2025). Optimized Design of Medical Welded Structures for Life Enhancement.
9. Cavallotti, E. (2024). Study of the Future Airborne Capability Environment (FACE) and proposal of an avionics SW architecture implementing Open Architecture (OA), Integrated Modular Avionics (IMA) and Modular Open Systems Approach (MOSA) concepts (Doctoral dissertation, Politecnico di Torino).
10. Chen, F., Liang, H., Yue, L., Xu, P., & Li, S. (2025). Low-Power Acceleration Architecture Design of Domestic Smart Chips for AI Loads.
11. Hanada, H., & Ishibashi, K. (2025). Service-Level Objective-Aware Load-Adaptive Timeout: Balancing Failure Rate and Latency in Microservices Communication. IEEE Access.
12. Scialacqua, L., Anwar, S., Mioc, F., Lelievre, A., Mantash, M., Luc, J., ... & Foged, L. J. (2023, March). Non-Invasive SAR Using OTA Measurements and Numerical Post Processing. In 2023 17th European Conference on Antennas and Propagation (EuCAP) (pp. 1-5). IEEE.
13. Chen, H., Ma, X., Mao, Y., & Ning, P. (2025). Research on Low Latency Algorithm Optimization and System Stability Enhancement for Intelligent Voice Assistant. Available at SSRN 5321721.
14. Sharma, N., & Shambharkar, P. G. (2025). Multi-layered security architecture for IoMT systems: integrating dynamic key management, decentralized storage, and dependable intrusion detection framework. International Journal of Machine Learning and Cybernetics, 1-48.
15. Yang, M., Cao, Q., Tong, L., & Shi, J. (2025, April). Reinforcement learning-based optimization strategy for online advertising budget allocation. In 2025 4th International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID) (pp. 115-118). IEEE.
16. Aguilar, A. (2023). Lowering Mean Time to Recovery (MTTR) in Responding to System Downtime or Outages: An Application of Lean Six Sigma Methodology. In 13th Annual International Conference on Industrial Engineering and Operations Management.
17. Wu, C., Zhang, F., Chen, H., & Zhu, J. (2025). Design and optimization of low power persistent logging system based on embedded Linux.
18. Gu, J., Narayanan, V., Wang, G., Luo, D., Jain, H., Lu, K., ... & Yao, L. (2020, November). Inverse design tool for asymmetrical self-rising surfaces with color texture. In Proceedings of the 5th Annual ACM Symposium on Computational Fabrication (pp. 1-12).
19. Wu, Q., Shao, Y., Wang, J., & Sun, X. (2025). Learning Optimal Multimodal Information Bottleneck Representations. arXiv preprint arXiv:2505.19996.
20. Sedghpour, M. R. S., & Townend, P. (2022, August). Service mesh and ebpf-powered microservices: A survey and future directions. In 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE) (pp. 176-184). IEEE.

 doi:10.20944/preprints202601.1273.v1

21. Tan, L., Peng, Z., Liu, X., Wu, W., Liu, D., Zhao, R., & Jiang, H. (2025, February). Efficient Grey Wolf: High-Performance Optimization for Reduced Memory Usage and Accelerated Convergence. In 2025 5th International Conference on Consumer Electronics and Computer Engineering (ICCECE) (pp. 300-305). IEEE.
22. Fleischer, M., Das, D., Bose, P., Bai, W., Lu, K., Payer, M., ... & Vigna, G. (2023). {ACTOR}:{Action-Guided} Kernel Fuzzing. In 32nd USENIX Security Symposium (USENIX Security 23) (pp. 5003-5020).
23. Cai, B., Bai, W., Lu, Y., & Lu, K. (2024, June). Fuzz like a Pro: Using Auditor Knowledge to Detect Financial Vulnerabilities in Smart Contracts. In 2024 International Conference on Meta Computing (ICMC) (pp. 230-240). IEEE.
24. Du, Y. (2025). Research on Deep Learning Models for Forecasting Cross-Border Trade Demand Driven by Multi-Source Time-Series Data. Journal of Science, Innovation & Social Impact, 1(2), 63-70.