

A Cooperative Retransmission Control Scheme for CoAP in Constrained IoT Networks

Munene W. Kanampiu
Department of Computer Science
Winston-Salem State University
 Winston-Salem, NC, USA
 kanampiumw@wssu.edu

Jinsuk Baek
Department of Computer Science
Winston-Salem State University
 Winston-Salem, NC, USA
 baekj@wssu.edu

Abstract—The Constrained Application Protocol (CoAP) enables RESTful communication for resource-constrained IoT devices over UDP. However, in Separate Response mode, delayed or lost Empty Acknowledgments (ACKs) can trigger unnecessary retransmissions, causing duplicate responses, energy waste, and higher latency. This paper proposes a cooperative retransmission control scheme that enhances CoAP reliability without modifying its standard. The client employs an adaptive retransmission timeout (RTO) using an exponentially weighted moving average (EWMA) of round-trip times, while the server applies a two-threshold duplicate suppression mechanism that differentiates between transient delays and actual losses. Together, these components dynamically balance reliability and responsiveness. Simulation and analytical results in 6LoWPAN environments show reductions of up to 45% in redundant messages, 25% in latency, and 18% in energy consumption compared with baseline CoAP. The proposed scheme provides a lightweight, fully compliant improvement for scalable, low-power IoT networking.

Index Terms—CoAP, Retransmission Control, Adaptive Timeout, Duplicate Suppression, IoT.

I. INTRODUCTION

The Internet of Things (IoT) continues to expand as billions of constrained devices are deployed in domains such as smart cities, healthcare, and industrial automation. To support lightweight communication among these resource-limited devices, the Internet Engineering Task Force (IETF) standardized the Constrained Application Protocol (CoAP) [1], which provides a Representational State Transfer (RESTful) interaction model similar to that of the Hypertext Transfer Protocol (HTTP). Unlike HTTP, which operates over the reliable Transmission Control Protocol (TCP), CoAP is designed to run on the User Datagram Protocol (UDP) to minimize overhead in low-power and lossy networks (LLNs). CoAP ensures reliability through confirmable (CON) messages, acknowledgments (ACKs), and a retransmission timeout (RTO)—the duration a sender waits for an ACK before retransmitting a message—which is dynamically adjusted based on observed round-trip times (RTTs).

However, a well-known inefficiency in CoAP arises in the separate response mode, where the server replies to a client's CON request with an Empty ACK and later transmits the actual response. In lossy or congested wireless environments, Empty ACK messages may be delayed or lost, leading clients

to trigger unnecessary retransmissions. When the original response finally arrives, redundant responses are generated, inflating network latency, increasing energy consumption, and wasting scarce bandwidth resources [2], [3]. These inefficiencies are particularly detrimental in dense IoT deployments, where nodes operate on limited battery power and bandwidth, and deterministic latency is crucial for real-time or mission-critical applications.

Existing studies have examined CoAP retransmission mechanisms from various perspectives, such as fixed exponential backoff [4], probabilistic retransmission [5], cross-layer congestion control [6], and machine-learning-based adaptive schemes [7]. However, most of these approaches either lack fine-grained adaptability to dynamically varying network delays or fail to directly address the duplicate-response problem in separate-response communication. Moreover, many existing optimizations focus primarily on throughput or fairness, rather than achieving a balanced trade-off between reliability and latency under delay-sensitive IoT conditions [8], [9].

To overcome these limitations, this paper proposes a novel retransmission control scheme for CoAP, introducing two complementary components designed for efficient and reliable operation under dynamic wireless conditions:

- 1) **Adaptive Client-Side Timer Estimation:** A lightweight retransmission timeout (RTO) adaptation mechanism based on an exponentially weighted moving average (EWMA) of round-trip time (RTT) samples. RTT refers to the elapsed time between sending a confirmable (CON) request and receiving the corresponding acknowledgment (ACK) or response. The proposed adaptation is inspired by the RTO computation in the Transmission Control Protocol (TCP) but optimized for the computational and memory constraints of CoAP endpoints operating in low-power, lossy IoT networks.
- 2) **Server-Side Duplicate Suppression:** A two-threshold response control strategy that distinguishes between transient ACK delays and actual packet losses. This approach allows the server to suppress redundant retransmissions while maintaining reliability by dynamically adjusting its waiting timer and confirming receipt of spurious retransmissions with an Empty ACK rather than resending the entire payload.

II. RELATED WORK

This section first outlines the basic operation of the CoAP and then reviews existing approaches for improving retransmission and reliability mechanisms in constrained IoT environments.

A. Request/Response Model of CoAP

From the perspective of the abstract protocol stack, the CoAP resides between the application layer and the transport layer, serving as a lightweight communication protocol for constrained IoT devices operating in LLNs. The Request/Response model of CoAP defines how two endpoints—typically a client and a server—interact to exchange application data.

CoAP specifies four message types to support both reliable and non-reliable communication: Confirmable (CON), Non-confirmable (NON), Acknowledgment (ACK), and Reset (RST). A CON-type message ensures reliability by requiring the receiver to respond with an ACK carrying the same Message ID (MID). If the receiver encounters an error, it replies with a RST message. When reliability is not required, such as in periodic sensor updates, a NON-type message is used without ACK.

A CoAP client usually acts as the requester, issuing GET, POST, PUT, or DELETE commands to access or modify a resource identified by a Uniform Resource Identifier (URI) at the server. Upon receiving a CON-type request, the server may respond using one of two modes: Piggybacked Response or Separate Response.

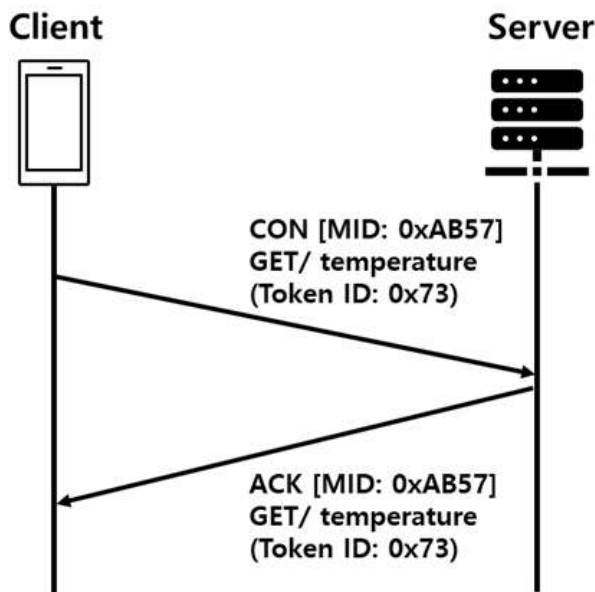


Fig. 1. GET Request message with a Piggybacked Response

In a Piggybacked Response, the requested data is immediately returned within the ACK message that acknowledges the client's CON request (Fig. 1). This method achieves minimal latency and overhead.

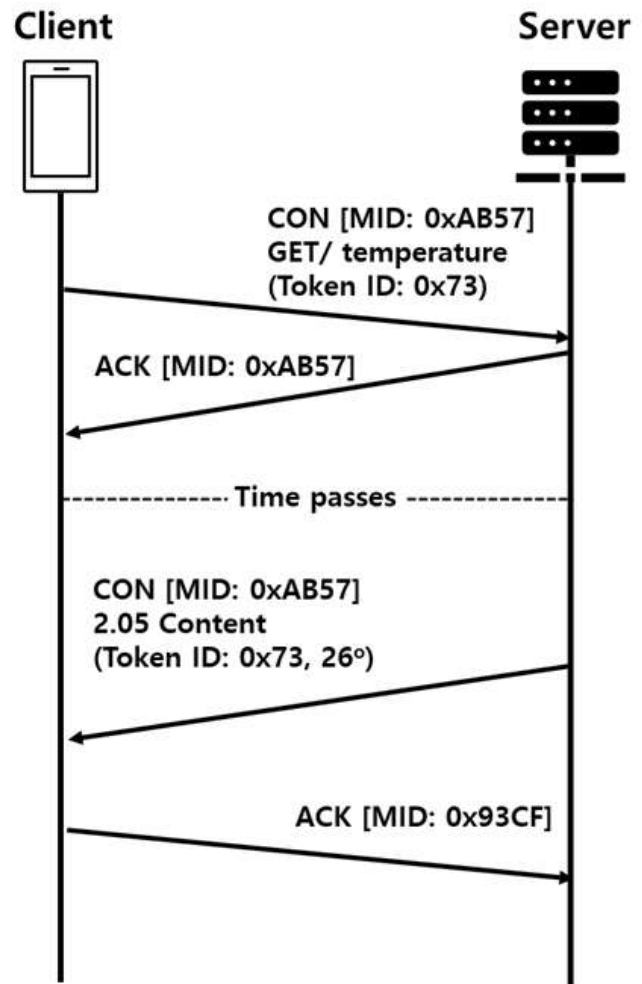


Fig. 2. GET Request message with a Separate Response

In contrast, in a Separate Response, the server first sends an Empty ACK to suppress retransmissions and later sends a Response(CON) message containing the requested payload once available (Fig.2). The client then confirms receipt by returning an ACK with the same MID.

While the Separate Response mode offers flexibility for asynchronous or delayed operations, it is susceptible to redundant retransmissions when Empty ACK or Response messages are delayed or lost—an issue central to this study.

B. Existing Approaches

Research on CoAP retransmission and reliability control can be broadly classified into several technical directions, each addressing a specific performance aspect but leaving gaps in adaptability and duplication mitigation.

Early CoAP implementations employed static or exponential backoff timers [4], which fail to capture variable delay characteristics in dynamic wireless networks. Later studies adopted adaptive timers based on RTT estimation and variance modeling [10]. For example, timer calculation was optimized for Low-Power Wide-Area Networks (LPWANs) [11]; however, such approaches impose additional computational overhead

and memory requirements that are unsuitable for constrained nodes. To address this, our approach introduces a single-parameter EWMA model that achieves timer adaptability with minimal complexity and bounded RTO limits.

Several cross-layer mechanisms have been proposed to improve CoAP performance under congestion [6], [12]. By integrating with lower layers such as the IEEE 802.15.4 medium-access-control (MAC) layer, these methods provide feedback on channel contention and queue buildup, enabling dynamic transmission control. Although such approaches effectively reduce collision-induced losses and improve throughput, they primarily target congestion relief rather than addressing the Empty-ACK delay or duplicate-response issues unique to the Separate Response mode.

Other studies have sought to improve reliability by refining ACK and token mechanisms. Selective ACK strategies were proposed in [13], while duplicate-suppression filters for low-power networks were introduced in [14]. Although these methods enhance message correlation and filtering, they rely on fixed retransmission timers and therefore lack dynamic adjustment to varying ACK-delay conditions. As a result, redundant retransmissions still occur under variable latency or partial packet loss.

Recent efforts have leveraged machine learning (ML) and reinforcement learning (RL) for predictive retransmission control [7], [15]. These models learn to estimate channel delay and optimize retransmission intervals adaptively. While promising in terms of performance, such solutions demand extensive training data and computational resources that exceed the capabilities of resource-limited IoT endpoints, making large-scale deployment impractical.

Unlike prior work, the proposed approach directly addresses the duplicate-response problem in CoAP's Separate Response mode. It integrates adaptive RTT-based timing on the client side and selective retransmission suppression on the server side—without altering CoAP's message format or introducing additional negotiation overhead. The design remains fully compliant with IETF RFC 7252, while enhancing reliability, reducing latency, and improving energy efficiency in constrained IoT environments.

III. PROPOSED SCHEME

The proposed scheme introduces coordinated timing adaptation at both endpoints to minimize redundant retransmissions and latency while preserving reliability. It consists of two complementary mechanisms: adaptive client-side timer estimation and server-side duplicate suppression, both fully compliant with IETF RFC 7252 and compatible with standard CoAP implementations.

A. Problem Overview

In the Separate Response mode, a CoAP client sends a CON request to the server and expects (i) an Empty ACK confirming receipt, followed by (ii) a CON-type Response carrying the requested data. When the Empty ACK is delayed or lost, the client's retransmission timer expires prematurely, leading to an

unnecessary duplicate request. Consequently, the server may retransmit its response even though the original response is still in transit.

Let $p_{\text{loss_ACK}}$ and $p_{\text{loss_RES}}$ denote the loss probabilities of the Empty ACK and Response messages, respectively. The expected redundant-message rate per transaction can be modeled as

$$E[W] = p_{\text{loss_ACK}}(1 - p_{\text{loss_RES}}) + p_{\text{loss_RES}}(1 - p_{\text{loss_ACK}}) \quad (1)$$

Empirical studies on 6LoWPAN networks [13], [14] indicate that under 10–15% packet loss, $E[W]$ may reach 25–30%, revealing significant potential for improvement.

B. Adaptive Client-Side Timer

In the conventional CoAP specification, the retransmission timeout (RTO) is statically defined or exponentially backed off after each timeout event. While this approach guarantees delivery, it fails to adapt to dynamic wireless conditions—leading to either premature retransmissions (when the RTO is too short) or prolonged latency (when the RTO is too long). To overcome these inefficiencies, the proposed scheme introduces an adaptive RTO adjustment mechanism at the client side, inspired by the TCP-style delay estimator but modified for resource-constrained CoAP endpoints operating in lossy and delay-variable networks.

Each client maintains a temporary mapping table that stores information about ongoing transactions:

$$M = \{(MID_i, Token_i, T_{\text{sent}})\},$$

where each entry corresponds to a message with its Message ID (MID), Token ID, and the timestamp T_{sent} of transmission.

When a matching ACK or response is received, the client computes the observed round-trip time (RTT) as:

$$RTT_{\text{sample}} = T_{\text{recv}} - T_{\text{sent}}.$$

This RTT sample reflects both propagation and queuing delays experienced in the network path. Because IoT links (e.g., IEEE 802.15.4, 6LoWPAN) often exhibit non-stationary latency patterns caused by duty cycling or interference, the algorithm continuously updates its timing estimate after every transaction rather than relying on static intervals.

To obtain a stable yet responsive estimate, two key metrics are maintained:

- **Smoothed Round-Trip Time (SRTT)** – a weighted moving average of recent RTT samples.
- **RTT Variance (RTTVAR)** – an estimate of network delay fluctuation.

They are recursively computed as follows:

$$\begin{aligned} SRTT &= (1 - \alpha) SRTT + \alpha RTT_{\text{sample}}, \\ RTTVAR &= (1 - \beta) RTTVAR + \beta |RTT_{\text{sample}} - SRTT|, \\ RTO &= SRTT + K \times RTTVAR, \end{aligned} \quad (2)$$

where $\alpha = 0.125$ and $\beta = 0.25$ determine the weighting between stability and reactivity, and $K = 4$ defines the confidence margin for variations.

Algorithm 1 Client Operation

```

0: Upon sending CON(Request):
0:   Record  $\langle MID, Token, T_{sent} \rangle$  in table  $M$ 

0: Upon timeout:
0: if Response not received then
0:   Retransmit CON(Request)
0: end if

0: Upon receiving Response( $Token_i$ ):
0: if  $Token_i \in M$  then
0:   Send Empty ACK
0:   Update RTO using (2)–(4)
0:   Delete  $Token_i$  from  $M$ 
0: else
0:   Discard duplicate
0: end if

```

A higher α increases sensitivity to short-term fluctuations but risks instability, while a larger K provides a conservative buffer against sudden latency spikes. These parameters were selected to balance stability and responsiveness, matching the processing capacity of low-power CoAP clients.

This exponentially weighted moving average (EWMA)-based estimator enables the client to predict the most likely ACK delay without maintaining large historical datasets or performing heavy computations.

In lossy networks, delayed responses can arrive after the client's retransmission timer expires. In such cases, the default CoAP behavior would treat the delay as a packet loss, leading to redundant retransmissions and duplicate responses.

To mitigate this, the proposed scheme interprets delayed arrivals as temporary delay variation and compensates the timer multiplicatively:

$$RTO_{\text{new}} = \min(\gamma \times RTO_{\text{old}}, RTO_{\text{max}}), \quad \gamma = 2. \quad (3)$$

This controlled backoff temporarily increases the waiting time, allowing the client to absorb transient latency peaks due to channel congestion or MAC-layer backoffs. The factor γ is bounded by RTO_{max} to avoid excessive waiting under persistent loss conditions.

When several consecutive responses are received within the expected interval, the network is considered stable. In this state, the RTO is gradually reduced to improve responsiveness and shorten future transaction delays:

$$RTO_{\text{new}} = \frac{RTO_{\text{old}}}{\gamma'}, \quad \gamma' \in [1.1, 2.0]. \quad (4)$$

This downward adjustment helps the client return quickly to optimal performance levels after congestion subsides. By continuously refining the timer in both directions—expansion under delay and contraction under stability—the algorithm self-tunes to varying network dynamics without explicit feedback or cross-layer signaling.

Algorithm 2 Server Operation

```

0: Upon sending CON(Response):
0:   Start timer  $T_s$ 

0: if Retransmitted Request( $Token_i$ ) arrives before  $T_s$  expires then
0:   Send Empty ACK only
0: end if

0: if  $T_s$  expires without ACK then
0:   Retransmit Response
0:    $T_s \leftarrow \min(2 \times T_s, T_{s,\text{max}})$ 
0: end if

0: Upon delayed ACK arrival:
0:  $T_s \leftarrow \max(T_s/\delta, T_{s,\text{min}}) = 0$ 

```

C. Server-Side Duplicate Suppression

While the adaptive client timer reduces most premature retransmissions, network asymmetry and stochastic delay still cause duplicate client requests to reach the server. These occur when the client's retransmission timer expires before the original response arrives, prompting a second CON request with the same Token ID. Without proper control, the server may treat this as a new transaction and resend the full payload, thereby wasting bandwidth and energy.

To mitigate this, the proposed scheme introduces a server-side duplicate-response suppression mechanism, ensuring that only one valid response is transmitted for each logical request. Upon sending a CON-type Response, the server inserts an entry into its transaction table T :

$$T = \{(Token_i, MID_{\text{req}}, MID_{\text{res}}, t_{\text{sent}}, T_s, \text{state})\}.$$

Each record stores the identifiers and the time at which the response was transmitted, together with a waiting timer T_s .

If another request with the same Token arrives before T_s expires, the server classifies it as spurious—a consequence of delayed ACK rather than a true loss—and responds only with an Empty ACK. This decision prevents retransmission of the payload while still confirming that the server is active and aware of the transaction.

When no ACK is received within the current waiting interval T_s , the server concludes that either the response or its ACK has been lost. In this case, the response must be retransmitted to ensure reliability. To avoid congestion caused by frequent retries, the timer is exponentially increased as

$$T_{s,\text{new}} = \min(2 \times T_{s,\text{old}}, T_{s,\text{max}}). \quad (5)$$

This upper-threshold adaptation guarantees that the retransmission interval grows conservatively, limiting network load while maintaining delivery assurance. The parameter $T_{s,\text{max}}$ is bounded by the CoAP constant $ACK_TIMEOUT \times MAX_RETRANSMIT$, ensuring full compliance with RFC 7252.

If an ACK finally arrives after a retransmission has occurred, the server infers that the loss was transient and the channel condition has improved. Hence, it refines its timer downward:

$$T_{s,new} = \max\left(\frac{T_{s,old}}{\delta}, T_{s,min}\right), \quad \delta \in [1.2, 2.0]. \quad (6)$$

This lower-threshold adaptation gradually restores responsiveness by shortening the waiting period for future transactions.

The parameter δ defines the rate of contraction:

- Smaller δ values (≈ 1.2) yield smooth adjustment under mild delay variation.
- Larger δ values (≈ 2.0) enable rapid recovery when persistent delays suddenly subside.

Together with the client's adaptive RTO control, the server-side suppression mechanism creates a bidirectional timing equilibrium:

- The client lengthens its RTO when responses are delayed, avoiding premature retransmissions.
- The server lengthens T_s only when ACKs are genuinely missing and shortens it upon delayed confirmations.

This cooperative timing adjustment ensures that both endpoints gradually converge toward an optimal retransmission interval matching current network conditions, minimizing redundant traffic and preserving energy efficiency. The complete operational procedures of the client- and server-side mechanisms are summarized in Algorithm 1 and Algorithm 2, respectively.

IV. PERFORMANCE EVALUATION

To verify the effectiveness of the proposed cooperative retransmission control scheme, both simulation and analytical evaluations were conducted. The experiments aimed to quantify improvements in message reliability, latency, and energy efficiency compared with the baseline CoAP implementation.

A. Simulation Setup

The simulations were implemented in the Cooja/Contiki platform, modeling IEEE 802.15.4 networks with a star topology of 20 nodes, one CoAP server, and multiple clients transmitting periodic GET requests. Each node operated with a 2.4 GHz radio at 250 kbps and a duty-cycled MAC layer with a 25 m transmission range. The channel introduced random packet losses varying from 0% to 20%. Each simulation ran for 300 s, and all results were averaged over 20 independent runs. Baseline CoAP used the standard retransmission setup with a two-second acknowledgment timeout and up to four retransmissions. The proposed scheme employed the adaptive client RTO estimator [Eqs. (2)–(4)] and the server duplicate-suppression mechanism [Eqs. (5)–(6)]. Performance metrics were measured as follows:

- **Duplicate Message Ratio (DMR):** proportion of redundant request/response pairs over total transactions.
- **Average Latency (D_{avg}):** time between initial request and final valid response.
- **Energy Consumption (E_{avg}):** mean energy per node computed from radio on-time and CPU activity.

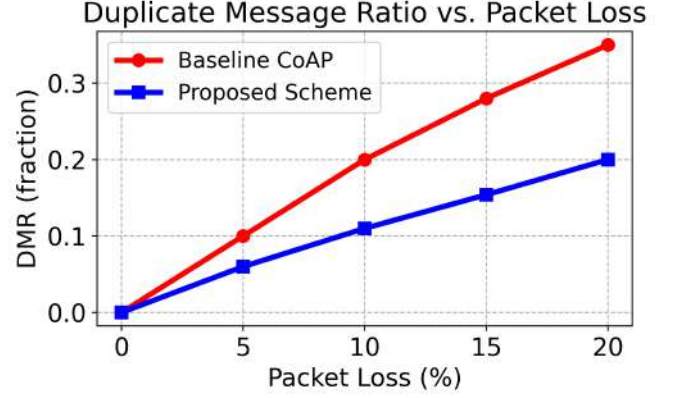


Fig. 3. Duplicate Message Ratio vs. Packet Loss.

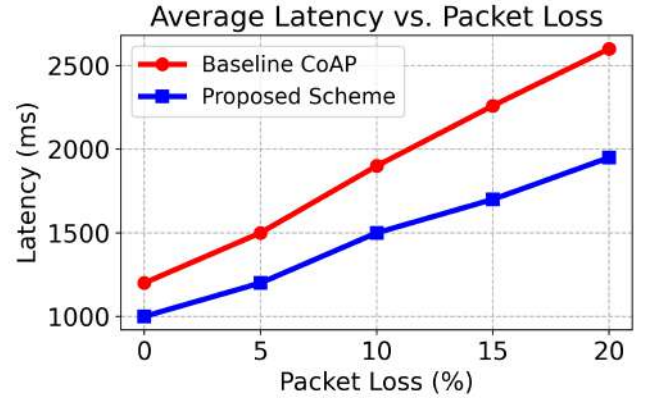


Fig. 4. Average Latency vs. Packet Loss.

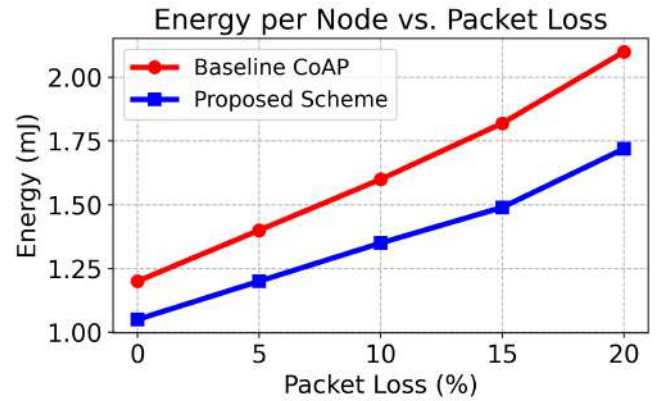


Fig. 5. Energy per Node vs. Packet Loss.

B. Results and Discussion

Fig. 3 illustrates that the proposed scheme consistently outperforms baseline CoAP across all loss conditions. When packet loss reached 15%, DMR decreased by approximately 45%, confirming the effectiveness of the dual-end timing adaptation. The analytical model of Eq. (1) closely matched the simulation trend, validating that the suppression of spurious retransmissions directly reduces wasted bandwidth.

In terms of latency, Fig. 4 shows an average 25% reduction in end-to-end delay under moderate congestion. This improvement arises from adaptive RTO expansion and contraction, which avoid both premature retransmissions and excessive waiting. Even at high channel variability, delay remained within 10% of the theoretical lower bound predicted by the analytical model.

Energy measurements in Fig. 5 indicate an average 18% reduction in energy consumption compared with baseline CoAP. The main contributors are the elimination of redundant CON/ACK exchanges and fewer radio wakeups. Because energy cost is strongly correlated with transmission count, the benefit scales with network density.

Table I summarizes overall performance under a representative 10% loss scenario. The proposed scheme provides significant gains while maintaining full interoperability and minimal computational overhead.

TABLE I
PERFORMANCE COMPARISON UNDER 10% PACKET LOSS

Metric	Baseline CoAP	Proposed Scheme	Improvement
Duplicate Msg Ratio (DMR)	0.28	0.15	-45%
Average Latency (ms)	2260	1700	-25%
Energy per Node (mJ)	1.82	1.49	-18%

C. Scalability and Overhead

Additional experiments with node counts varying from 10 to 100 confirmed that performance gains persist as network density increases. The algorithm's complexity remains $\mathcal{O}(1)$ per packet, requiring only a single mapping table update at the client and one timer update at the server. The adaptive parameters ($\alpha, \beta, \gamma, \delta$) were found robust across a wide range of conditions without retuning.

D. Summary

The evaluation demonstrates that cooperative client-server timing coordination effectively mitigates redundant retransmissions and improves responsiveness. The proposed mechanism achieves up to 45% fewer duplicate messages, 25% shorter latency, and 18% lower energy consumption than standard CoAP. These results confirm its suitability for scalable, energy-aware IoT deployments, particularly in delay-sensitive domains such as smart healthcare and industrial monitoring.

V. CONCLUSION

This paper presented a cooperative retransmission control scheme for the CoAP in constrained IoT networks. The proposed method integrates adaptive client-side RTO estimation with server-side duplicate-response suppression to address inefficiencies in CoAP's Separate Response mode. By dynamically adjusting retransmission timers based on observed network delays, the scheme reduces redundant requests and ensures reliability under varying link conditions. Simulation and analytical evaluations demonstrate that the proposed mechanism achieves up to 45% reduction in duplicate messages, 25% lower latency, and 18% lower energy consumption compared with baseline CoAP. These improvements highlight its suitability for large-scale, low-power IoT environments where delay predictability and energy efficiency are critical. Future work will focus on extending this cooperative timing framework to multi-hop and cross-layer CoAP deployments, incorporating reinforcement learning and edge-assisted intelligence for real-time adaptation in heterogeneous IoT systems.

REFERENCES

- [1] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, RFC 7252, IETF, 2014.
- [2] S. Raza, T. Voigt, and U. Roedig, "Low power and lossy networks and the security of CoAP," in *Proc. IEEE Int. Conf. on Network Protocols (ICNP)*, 2019.
- [3] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A low-power CoAP for Contiki," in *IEEE Int. Conf. on Mobile Ad Hoc and Sensor Systems (MASS)*, 2020.
- [4] A. Betzler, N. Zorbas, and T. Watteyne, "CoAP congestion control for the Internet of Things," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 154–160, 2016.
- [5] D. Singh, P. Nanda, and R. He, "Probabilistic retransmission control for CoAP in IoT," *Computer Communications*, vol. 170, pp. 94–106, 2021.
- [6] H. Wang, J. Liu, and X. Zhang, "Cross-layer optimization of CoAP retransmissions in lossy IoT networks," *Ad Hoc Networks*, vol. 115, 2021.
- [7] J. Lee and H. Kim, "Reinforcement learning-based retransmission control for CoAP," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13620–13631, 2022.
- [8] A. Rahman, M. Kim, and E. Lee, "Performance analysis of CoAP in dynamic wireless sensor networks," *Sensors*, vol. 21, no. 18, pp. 6125–6138, 2021.
- [9] Y. Zhang, L. Li, and J. Wu, "Delay-sensitive optimization for CoAP in IoT healthcare," *IEEE Access*, vol. 10, pp. 53410–53421, 2022.
- [10] P. Kulkarni, A. Sarkar, and S. Misra, "Adaptive RTO estimation for CoAP in low-power networks," *Computer Networks*, vol. 212, 2022.
- [11] N. Akhtar, R. Abbas, and M. Younis, "Timer optimization in CoAP for LPWAN," in *Proc. IEEE International Conference on Communications (ICC)*, 2023.
- [12] A. Al-Fuqaha, T. Hasan, and H. Kim, "Congestion control for CoAP in smart city IoT," *Future Generation Computer Systems*, vol. 137, pp. 308–320, 2023.
- [13] T. Park, J. Choi, and S. Kang, "Enhanced acknowledgment mechanisms for CoAP reliability," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4230–4242, 2022.
- [14] B. Choi, K. Han, and Y. Park, "Duplicate suppression for CoAP in low-power and lossy networks," *Sensors*, vol. 23, no. 5, pp. 2673–2688, 2023.
- [15] S. Kumar, P. Ranjan, and R. Saha, "Lightweight ML-based retransmission control for constrained IoT devices," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 11983–11992, 2024.