JATO: Deep Reinforcement Learning-based Joint Optimization for Task Offloading and Adaptive Transmission in Multimedia IoT Systems

Gina Purnama[1], Irma Amelia Dewi[1], Armein Z. R. Langi[1,2] & Yoanes Bandung[1,2,*]

[1]School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jalan Ganesa No. 10, Bandung, 40132, Indonesia
[2]ITB Research Center for Information and Communication Technology, Institut Teknologi Bandung, Jalan Ganesa No. 10, Bandung, 40132, Indonesia
*E-mail: yoanes.bandung@itb.ac.id

Abstract. As the Multimedia Internet of Things (M-IoT) evolves, the orchestration of numerous resources that offer support for high-bandwidth, low-latency applications arises as a key challenge. Architecturally, the edge-cloud framework alleviates structural concerns, but the linked nature of compute and data transfer poses problems of resource management. Approaches that tackle task offloading and adaptive transmission that think independently of each other tend to have problems such as user-server cross-region overloads or network congestion. This paper presents JATO, a framework to jointly tackle the problems of adaptive task offloading and transmission optimization using Deep Reinforcement Learning. JATO offers a mono-faceted solution, learning a policy to simultaneously determine the best offloading target and the transmission quality. The framework was implemented for evaluation with a combination of different edge devices in a testbed alongside a simulation environment. JATO recorded a result of 0.9321 as the holistic score of the overall framework endpoint, a score significantly better than that of all the other frameworks that were used as functional baselines. JATO was able to resource optimally with a network lag of 131.65 milliseconds and a network freeze of 0.09% with the resources utilized. This is evidence that offloading and rate control in combination provides better resource elasticity for M-IoT systems.

Keywords: *adaptive transmission; deep reinforcement learning; edge-cloud computing; joint optimization; quality of service; task offloading; multimedia IoT.*

1 Introduction

The rapid evolution of the Internet of Things (IoT) into the Multimedia IoT (M-IoT) paradigm is driven by the explosion of visual and audio data volumes. Unlike traditional scalar sensor data, M-IoT applications such as intelligent surveillance, telemedicine, and virtual reality demand continuous, high-volume multimedia streaming [1, 2]. This poses severe challenges for quality of service (QoS) assurance, specifically in balancing low latency, high throughput, and limited computational resources [3]. Addressing these challenges requires

Received on December 9th, 2025, Revised on May 20th, 2026, Accepted for publication on July 15th, 2026.

accurate network throughput prediction and robust resource management frameworks [4], and to address device-level computational limitations, hybrid edge-cloud computing architectures have been widely adopted, bringing processing closer to data sources to reduce propagation delays and mitigate backhaul congestion [5, 6].

However, agile resource orchestration in this environment remains a complex, multi-dimensional challenge [7]. A critical review of the existing literature revealed a fundamental research gap, namely that current optimization strategies predominantly treat computational task offloading and network-aware adaptive transmission as two isolated problem domains [8, 9]. This partial approach ignores the inherent cross-domain dependencies within M-IoT systems. For instance, an optimal offloading decision becomes ineffective if the underlying network is congested. Conversely, transmitting high-fidelity data is futile if the destination edge server is overloaded and experiencing queueing delays. Such siloed approaches often yield reactive, fragile, and sub-optimal solutions that fail to maintain stability in stochastic real-world environments [10, 11].

To overcome these limitations, dynamic environments require intelligent, autonomous orchestration [12, 13]. While deep reinforcement learning (DRL) has been applied to IoT to optimize dynamic systems, most DRL-based frameworks still operate within siloed domains to optimize either the offloading decision or the transmission parameters independently [14]. To bridge this gap, we propose JATO (Joint Adaptive Task Offloading and Transmission Optimization), a novel DRL-based framework [15]. The primary novelty of this research lies in the JATO agent's capability to learn a unified policy that executes joint actions simultaneously [16]. Instead of making sequential or isolated decisions, JATO decides both where a task should be processed (computation location) and how the data should be transmitted (delivery resolution) in a single inference step, based on a holistic state observation encompassing network latency, bandwidth, and edge CPU utilization [17]. To guide this study, we formulated the following research questions:

1. How can task offloading and adaptive transmission decisions be integrated into a unified Markov decision process (MDP) to prevent cross-domain bottlenecks?
2. How can task offloading and adaptive transmission decisions be integrated into a unified Markov decision process (MDP) to prevent cross-domain bottlenecks?
3. How can a Deep Q-Network (DQN) architecture be designed to learn a unified policy that adapts to fluctuating network latency and CPU resource availability?

4. How does the proposed joint optimization framework perform compared to isolated or static baseline heuristics in terms of latency, service quality, and resource utilization?

Consequently, the objectives of this research were to design, develop, and empirically validate the JATO framework to demonstrate that a joint optimization approach provides a more robust and balanced resource management solution for M-IoT ecosystems [18-20]. Thus, this study's principal contributions can be outlined as follows:

1. Propose a comprehensive joint optimization framework that formulates the intertwined issues of task offloading and adaptive transmission into a single, unified MDP, eliminating cross-domain barriers.
2. Construct an intelligent orchestration agent based on a DQN architecture capable of learning a unified policy to navigate the complex trade-offs between competing QoS metrics (e.g., visual quality vs system stability).
3. Rigorously evaluate the proposed framework against baseline approaches in both a simulated environment and a physical proof-of-concept prototype (using Raspberry Pi gateways), providing measurable performance gains.

2 Related Works

The existing literature on M-IoT resource optimization predominantly branches into two isolated trajectories, i.e., computation-centric task offloading and network-centric adaptive transmission. This section briefly reviews recent advancements in these domains and highlights the fundamental research gap that motivated our proposed joint optimization framework.

2.1 Task Offloading and Computational Resource Management

Recent studies on edge computing focused on intelligent task offloading to minimize processing latency and energy consumption. Ghosh *et al.* [13] introduced the PArtNNer framework, utilizing dynamically partitioned deep neural networks (DNNs) to balance inferencing demands between end devices and edge servers. To handle stochastic task arrivals, Xu *et al*. [21] proposed predictive traffic flow models to prevent edge server overbooking, while Li *et al.* [22] developed the Queec framework for QoS-aware offloading scheduling. Despite their robustness, these computation-centric approaches operate under the implicit assumption of a stable and highly capable communication channel. By neglecting network volatility, these strategies risk severe performance degradation when the underlying network experiences high jitter or congestion [23].

2.2 Adaptive Transmission and Network Aware Mechanisms

Conversely, network-centric approaches focus on modulating data transmission parameters such as video resolution and bitrate to accommodate fluctuating bandwidth. Bandung *et al.* [15] implemented a machine learning-based frame resolution adjustment system on IoT gateways to maintain smooth frame rates during bandwidth drops. Extending this concept, Lei *et al*. [24] proposed ACORN+, an adaptive compression-reconstruction framework where devices send low-resolution data that edge servers refine into high-resolution. Additionally, Tan *et al.* [20] introduced DACOD360, employing Deep Reinforcement Learning (DRL) to enhance the delivery of 360-degree videos. However, a critical limitation in these network-oriented studies is the assumption that the destination server possesses unlimited computational capacity. Optimizing only the data delivery path risks shifting the bottleneck from the network to the server, ultimately increasing end-to-end latency due to severe queueing delays [25-27].

2.3 Deep Reinforcement Learning in Dynamic Orchestration

To navigate the stochastic characteristics of IoT ecosystems, DRL has emerged as a powerful orchestration tool. Recent studies have further demonstrated that DRL-based intelligent approaches significantly outperform traditional heuristic methods in dynamic resource and channel allocation [28]. Wang *et al.* [29] utilized DRL in the DeepCast framework to personalize quality of experience (QoE) by learning user preferences and edge server states. Furthermore, Lekharu *et al.* [18] applied DRL for collaborative video caching at the edge, successfully improving cache hit ratios and reducing backhaul traffic. These applications demonstrate DRL's capability to autonomously optimize complex policies without relying on brittle, static heuristics.

2.4 Synthesis and the Research Gap

Despite the proven efficacy of DRL, existing approaches remain siloed, addressing either computation or transmission optimization independently. This dichotomy creates a blind spot. For example, aggressively transmitting high-fidelity video over a perfect network is futile if the destination edge server is computationally overwhelmed. Conversely, conservatively throttling video quality due to perceived poor network conditions ignores the possibility of routing the task to a highly capable cloud server.

JATO differs fundamentally from existing research by addressing this cross-domain interdependency. By integrating both computational constraints (edge CPU utilization) and communication constraints (latency and bandwidth) into a single, unified Markov decision process (MDP), JATO ensures that decisions

regarding where to compute and how to transmit are made simultaneously. This joint optimization is essential for achieving true stability and superior QoS in volatile M-IoT environments.

3 Methodology and System Model

This study employed the design science research (DSR) paradigm to design, develop, and evaluate JATO. In this section, we define the conceptual architecture, the formal MDP modeling, the deep neural network design, and the baseline algorithms used for comparative evaluation.

3.1 Hierarchical M-IoT System Architecture

The JATO framework operates within a three-tier hierarchical M-IoT architecture consisting of a device/gateway layer, an edge layer, and a cloud layer. The device layer comprises multimedia sensors (e.g., IP cameras) connected to an intelligent gateway (e.g., Raspberry Pi). To achieve minimal inference delay, the JATO DRL agent is deployed directly on the gateway. The gateway acts as a smart orchestrator that continuously monitors the environment and autonomously decides whether to offload computation to the low-latency but resource-constrained edge server, or to the highly scalable but distant cloud server, while simultaneously adjusting the video transmission resolution.

3.2 Markov Decision Process (MDP) Formulation

The joint optimization problem is formulated as a model-free MDP defined by a tuple (S, A, R, γ), representing the state space, action space, reward function, and discount factor, respectively.

3.2.1 State Space ($\boldsymbol{S}$)

The state vector s_t provides a parsimonious yet holistic observation of both the network and computational domains at time step t. It is defined as Eq. (1):

$$s_t = [l_t, b_t, u_t] \tag{1}$$

where l_t is the dynamic network latency to the edge server (in milliseconds), b_t is the available wireless bandwidth (in Mbps), and u_t is the current CPU utilization of the edge server (in percentage). This specific combination allows the agent to anticipate both network congestion and computational bottlenecks.

3.2.2 Joint Action Space ($\boldsymbol{A}$)

Unlike conventional approaches, JATO uses a unified action space that simultaneously determines the offloading destination and transmission

resolution. The entire collection of discrete joint actions that is available to the agent is captured mathematically in Eq. (2):

$$A = \{a_0, a_1, a_2\} \tag{2}$$

1. a_0 (Edge-LowRes) – A resource-conservative action that compresses video to a lower resolution before offloading to the edge server, minimizing both bandwidth and edge CPU load.
2. a_1 (Edge-HighRes) – An aggressive action that offloads high-fidelity, uncompressed data to the edge server, maximizing visual quality at the cost of high resource demand.
3. a_2 (Cloud-HighRes) – An escalation strategy that transmits high-resolution data to the cloud server. This action bypasses the edge server entirely to avoid local computational bottlenecks, albeit incurring higher propagation latency.

Other combinations, such as Cloud-LowRes, were deliberately excluded to streamline the learning process by eliminating sub-optimal pareto frontiers. The rationale is that if the system commits to enduring the high propagation latency of the cloud, it should fully exploit the cloud's virtually unlimited computational capacity by processing the highest-quality data. Compressing data sent to the cloud would artificially degrade visual quality without providing any meaningful computational benefit at the destination, rendering it a strictly dominated action.

3.2.3 Multi-Objective Reward Function (R)

The reward function mathematically translates the conflicting QoS objectives into a single scalar learning signal using weighted utility scalarization. The reward r_t received after executing action a_t in state s_t is calculated as follows:

$$r_t = w_q . Q(a_t) - w_l . L_{norm}(s_t, a_t) - w_u . U(a_t) \tag{3}$$

In Eq. (3), $Q(a_t)$ represents the visual quality score, which assigns a value of 8.0 for high-resolution actions (a_1, a_2) and 3.0 for the low-resolution action (a_0). The term $L_{norm}(s_t, a_t)$ denotes the total estimated end-to-end latency, encompassing transmission, queueing, and processing delays, normalized to a [0, 1] scale using $min\ (\frac{latency}{1000}, 1.0)$. Finally, $U(a_t)$ indicates the resultant edge CPU utilization penalty, ranging from 0 to 1.0, where this penalty is strictly set to 0 if the cloud is selected as the offloading destination.

To systematically determine the priority weights (w_q, w_l, w_u), an analytic hierarchy process (AHP) was employed, a method proven to provide robust feature weighting and multi-criteria decision-making in machine learning frameworks [30], as detailed in Table 1. Visual quality was judged slightly more

important than edge CPU utilization (a score of 3), and strongly more important than latency for surveillance tasks (a score of 5). Meanwhile, utilization was judged slightly more important than latency (a score of 3). This pairwise comparison matrix yields an eigenvector of approximately $w_A \approx 0.63$, $w_B \approx 0.26$, and $w_C \approx 0.11$. Rounding these values establishes the optimal configuration for M-IoT surveillance, i.e., $w_q = 0.6$, $w_l = 0.1$, and $w_u = 0.3$, ensuring that visual fidelity is prioritized while strictly penalizing edge saturation.

Table 1 AHP pairwise comparison matrix for priority weights.

Criteria	Visual Quality (A)	CPU Utilization (B)	Latency (C)
Visual quality (A)	1	3	5
CPU utilization (B)	1/3	1	3
Latency (C)	1/5	1/3	1

3.3 Deep Q-Network (DQN) Architecture and Hyperparameters

To map the continuous state space to discrete joint actions, we implemented a DQN architecture. The Q-network is a fully connected deep neural network consisting of:

1. Input layer – Three neurons (corresponding to the state vector size).
2. Hidden layers – Two fully connected hidden layers, each containing 128 neurons, utilizing the Rectified Linear Unit (ReLU) activation function to capture non-linear cross-domain dependencies.
3. Output layer – Three neurons with a linear activation function, outputting the estimated Q-values for actions a_0, a_1, a_2.

To stabilize training, the agent utilizes Experience Replay and a fixed target network. The complete and explicit hyperparameter configuration used for training the JATO agent is detailed in Table 2.

Table 2 Deep Q-network architecture and learning hyperparameters.

Category	Parameters	Value
Exploration	*Initial epsilon (ε_{start})*	1.0
	Minimum epsilon (ε_{end})	0.01
	Epsilon decay rate	4.95×10^{-5} per step
Learning	*Learning rate (α)*	0.0005
	Discount factor (γ)	0.99
	Mini-batch size (B)	64
Architecture	*Hidden layers & neurons*	$2\ layers\ \times\ 128\ neurons$
	Activation functions	ReLU (hidden), linear (output)
	Replay memory capacity (N)	10,000 transitions
Training	*Target network update frequency (C)*	Every 500 steps
	Total training steps (T_{train})	20,000 steps
	Total evaluation steps (T_{eval})	5,000 steps

3.4 Baseline Algorithms for Comparison

To rigorously evaluate the efficacy of the proposed joint optimization, JATO was compared against three standard heuristic baseline policies:

1. Edge-only – A greedy, static policy that always offloads tasks to the edge server at maximum resolution, completely ignoring edge CPU saturation risks.
2. Cloud-only – A conservative, static policy that offloads all high-resolution tasks to the cloud server, bypassing the edge to avoid local bottlenecks but enduring maximum network latency.
3. Reactive (rule-based) – A semi-adaptive policy that mimics traditional systems. It defaults to offloading to the edge server but reactively switches to the cloud server only when the edge CPU utilization exceeds a hardcoded threshold of 70%.

4 Results and Discussion

This section presents the comprehensive empirical evaluation of the JATO framework. The evaluation transitions from establishing learning convergence to a comparative performance analysis against baseline policies.

4.1 Experimental Setup and Workload Characteristics

To bridge the gap between simulated theory and physical application, the evaluation was conducted using a hybrid approach. The physical prototype consisted of a Raspberry Pi 4 acting as the IoT gateway, a local workstation serving as the edge server, and a DigitalOcean virtual instance acting as the cloud server.

The workload specifically utilized a continuous multimedia stream processed by a YOLOv8 object detection model. To simulate real-world variable bitrate (VBR) characteristics, the baseline frame size was set to $\sim 400\ KB$ (for high resolution) with a uniform random variation of $\pm 50\%$, operating at an average frame rate of 15 FPS. End-to-end latency (L_t) was rigorously measured as the total time elapsed from frame acquisition at the gateway, including network transmission, server queueing, and processing time, until the return of the JSON-formatted inference result. Network congestion conditions were modeled by dynamically injecting latency spikes ($> 300\ ms$) and bandwidth throttling ($< 2\ Mbps$).

4.2 Learning Convergence

The first phase of the evaluation analyzed the learning stability of the DQN agent across 20,000 training episodes. The agent's performance was highly sensitive to

the reward function weights. Extensive sensitivity analysis revealed that the Quality-Efficient configuration ($w_q = 0.6,\ w_l = 0.1,\ w_u = 0.3$) yielded the most optimal and stable convergence. As illustrated in the optimal configuration learning curve in Figure 1, the agent transitions from an initial exploration phase characterized by high volatility to a stable exploitation phase, eventually plateauing at a high cumulative reward of approximately 4.70. This convergence quantitatively validates that the DQN agent successfully mapped the state observations to the optimal joint actions.

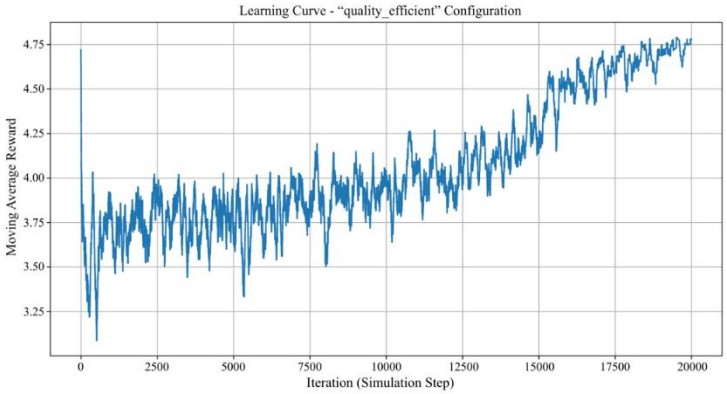

Figure 1 The learning curve of the JATO agent showing the transition from exploration to convergence.

4.3 Comparative Performance Analysis

The operational efficacy of JATO was benchmarked against three static heuristic baselines over 5,000 evaluation steps: Edge-Only (aggressively offloads to edge at max resolution), Cloud-Only (conservatively offloads to cloud at max resolution), and Reactive (a rule-based policy that shifts to cloud only when edge CPU exceeds 70%).

The aggregate key performance indicators (KPIs), visualized in Figure 2, demonstrated a substantial performance difference between the proposed framework and the isolated strategies. Under normal operational scenarios, the JATO framework achieved a highly competitive average latency of 100.92 milliseconds, alongside a near-zero average freezing delay (AFD) of 0.04% and a maximum average service quality (ASQ) score of 8.00.

In stark contrast, the Edge-Only policy suffered from severe performance degradation due to resource contention, recording an average latency of 872.26 milliseconds and an unviable edge CPU utilization rate of 136.48% resulting in

an AFD of 100%. The Reactive policy, hindered by its rigid threshold, achieved a stable latency but severely compromised visual fidelity, yielding an ASQ score of only 3.00.

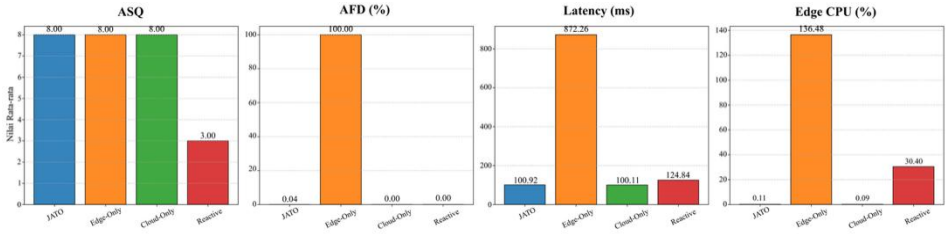

Figure 2 Comparative analysis of key QoS metrics under normal operational scenarios.

4.4 In-Depth Interpretation and Statistical Significance

To validate the observed performance gains, a Welch's t-test was conducted on the latency distributions across the evaluation steps. The analysis confirmed that the latency reduction achieved by JATO compared to the Edge-Only baseline was statistically significant ($p < 0.01$). Figure 3 presents the box plot distribution of the end-to-end latency, further highlighting JATO's superior stability and minimal jitter compared to the heavy tail variance observed in the Edge-Only policy.

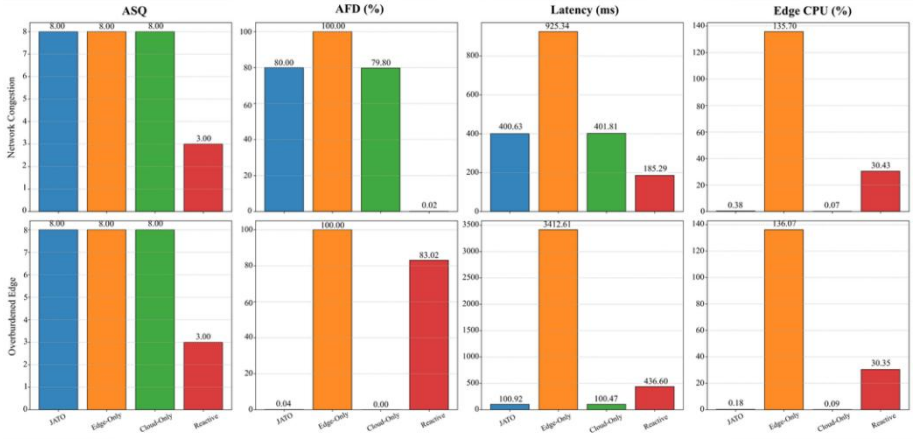

Figure 3 Performance degradation analysis under computational stress conditions.

The underlying mechanism driving this significant improvement lies in JATO's proactive, cross-domain awareness. Based on queueing theory, when a shared resource (edge CPU) approaches 100% utilization, queueing delays increase exponentially. The Edge-Only policy blindly forces tasks into this saturated environment, creating a severe computational bottleneck. JATO, however, continuously observes the edge CPU state and network conditions. By intelligently redirecting tasks to the cloud or dynamically compressing the resolution before the edge server reaches critical saturation, JATO effectively circumvents the $M/M/1$ queueing explosion.

To consolidate these conflicting metrics into a single objective evaluation, the Simple Additive Weighting (SAW) method was applied. As detailed in Table 3, JATO achieved a dominant holistic performance score of 0.9321, significantly outperforming the baseline methods.

Table 3 Final holistic performance scores and ranking using SAW method.

Framework	Rank	Holistic Score
JATO	**1**	**0.9321**
Edge-Only	2	0.8994
Cloud-Only	3	0.6000
Reactive	4	0.1235

4.5 System Limitations

While JATO provides robust optimization, the evaluation identified specific boundaries to its operational effectiveness. A primary limitation is the agent's susceptibility to extrapolation bias when encountering out-of-distribution (OOD) network anomalies. If network conditions fluctuate drastically within a window smaller than the agent's inference cycle (e.g., high-frequency jitter), the state information becomes stale, potentially leading to sub-optimal actions.

5 Conclusion

This study addresses the critical challenge of agile resource orchestration in Multimedia Internet of Things (M-IoT) systems by proposing JATO, a deep reinforcement learning-based framework. Unlike traditional approaches that isolate computational task offloading from adaptive network transmission, JATO integrates these domains into a unified Markov decision process. Through extensive simulations and a physical Raspberry Pi-based prototype, the DQN agent successfully demonstrated its capability to learn a unified policy that executes joint actions simultaneously.

The empirical results, supported by statistical significance analysis ($p < 0.01$), indicated that JATO effectively circumvents exponential queueing delays at the

edge while adapting to fluctuating bandwidth. By proactively balancing these constraints, JATO achieved a superior holistic performance score of 0.9321, significantly outperforming standard heuristic baselines. These findings confirm that joint optimization provides a robust and reliable architecture suitable for latency-sensitive M-IoT applications, such as intelligent surveillance and telemedicine.

Despite these advancements, the current framework is constrained by its single-agent architecture, which is evaluated on a single gateway. In real-world dense IoT deployments, multiple gateways often compete for the same limited edge resources. Therefore, future research should explore the scalability of this joint optimization paradigm by implementing multi-agent reinforcement learning (MARL) architectures, utilizing advanced algorithms such as Proximal Policy Optimization (PPO) or Actor-Critic, to manage distributed resource competition effectively.

Acknowledgement

This work was supported in part by the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia and the School of Electrical Engineering and Informatics, Institut Teknologi Bandung.

References

[1] Nauman, A., Qadri, Y.A., Amjad, M., Bin Zikria, Y., Afzal, M.K. & Kim, S.W., *Multimedia Internet of Things: A Comprehensive Survey*, IEEE Access, **8**, pp. 8202-8250, 2020, doi: 10.1109/ACCESS.2020.2964280.

[2] de Moraes, A.L.S., de Macedo, D.D.J. & Pioli, L., *Video Streaming on Fog and Edge Computing Layers: A Systematic Mapping Study*, Dec. 01, 2024, Elsevier B.V. doi: 10.1016/j.iot.2024.101359.

[3] Dai, P., Song, F., Liu, K., Dai, Y., Zhou, P. & Guo, S., *Edge Intelligence for Adaptive Multimedia Streaming in Heterogeneous Internet of Vehicles*, IEEE Trans. Mob. Comput., **22**(3), pp. 1464-1478, Mar. 2023, doi: 10.1109/TMC.2021.3106147.

[4] Eliviani, R. & Bandung, Y., WSN-IoT Forecast: Wireless Sensor Network Throughput Prediction Framework in Multimedia Internet of Things," Journal of ICT Research and Applications, **17**(3), pp. 336-355, Dec. 2023, doi: 10.5614/itbj.ict.res.appl.2023.17.3.4.

[5] Xiao, H., Zhuang, Y., Xu, C., Wang, W., Zhang, H., Ding, R., et al., *Transcoding-Enabled Cloud-Edge-Terminal Collaborative Video Caching in Heterogeneous IoT Networks: An Online Learning Approach With Time-Varying Information*, IEEE Internet Things J., **11**(1), pp. 296-310, Jan. 2024, doi: 10.1109/JIOT.2023.3312916.

[6] Benzerogue, S., Abdelatif, S., Merniz, S., Harous, S. & Khamer, L., *Multi-Path Transmission Protocol for Video Streaming Over Vehicular Fog Computing Environments*, IEEE Access, **12**, pp. 87199-87216, 2024, doi: 10.1109/ACCESS.2024.3417288.

[7] Sun, Y., Wei, T., Li, H., Zhang, Y. & Wu, W., *Energy-Efficient Multimedia Task Assignment and Computing Offloading for Mobile Edge Computing Networks*, IEEE Access, **8**, pp. 36702-36713, 2020, doi: 10.1109/ACCESS.2020.2973359.

[8] Nguyen, T., Katila, R. & Gia, T.N., *An Advanced Internet-of-Drones System with Blockchain for improving Quality of Service of Search and Rescue: A Feasibility Study*, Future Generation Computer Systems, **140**, pp. 36-52, Mar. 2023, doi: 10.1016/j.future.2022.10.002.

[9] Mehrabi, A., Siekkinen, M., Kämäräinen, T. & Ylä-Jääski, A., *Multi-Tier CloudVR: Leveraging Edge Computing in Remote Rendered Virtual Reality*, ACM Transactions on Multimedia Computing, Communications and Applications,. **17**(2), Jun. 2021, doi: 10.1145/3429441.

[10] Alsmirat M. & Sarhan, N.J., *Intelligent Optimization for Automated Video Surveillance at the Edge: A Cross-Layer Approach*, Simul. Model. Pract. Theory, **105**, Dec. 2020, doi: 10.1016/j.simpat.2020.102171.

[11] Romero, J.S., Ponce, A.d.R., Nakimuli, W., Bermejo, D.J., Garcia-Reinoso, J, Contreras, L.M. & Alvarez, F., *Design, Implementation, and Validation of a Multi-Site Gaming Streaming Service Over a 5G-Enabled Platform," IEEE Transactions on Broadcasting*, **68**(2), pp. 464-474, Jun. 2022, doi: 10.1109/TBC.2022.3141615.

[12] Kang, J. & Chung, K., *RL-based HTTP adaptive streaming with edge collaboration in multi-client environment*, Mar. 01, 2024, Academic Press. doi: 10.1016/j.jnca.2024.103833.

[13] Ghosh, S.K., Raha, A., Raghunathan, V. & Raghunathan A., *PArtNNer: Platform-Agnostic Adaptive Edge-Cloud DNN Partitioning for Minimizing End-to-End Latency*, ACM Transactions on Embedded Computing Systems, **23**(1), Jan. 2024, doi: 10.1145/3630266.

[14] Feng, H., Qiao, L. & Lv, Z., *Innovative soft computing-enabled cloud optimization for next-generation IoT in digital twins*, Appl. Soft Comput., **136**, Mar. 2023, doi: 10.1016/j.asoc.2023.110082.

[15] Bandung, Y., Wicaksono, M.A., Pribadi, S., Langi, A.Z.R. & Tanjung, D., *IoT Video Delivery Optimization through Machine Learning-Based Frame Resolution Adjustment*, ACM Transactions on Multimedia Computing, Communications and Applications, **20**(9), Sep. 2024, doi: 10.1145/3665929.

[16] Badshah, A., Iwendi, C., Jalal, A., Ul Hasan, S.S., Said, G., Band, S.S. & Chang, A., *Use of Regional Computing to Minimize the Social Big Data Effects*, Comput. Ind. Eng., **171**, 108433, Sep. 2022, doi: 10.1016/j.cie.2022.108433.

[17] F. Santos, R. Immich, and E. R. M. Madeira, "*Multimedia Services Placement Algorithm for Cloud–Fog Hierarchical Environments*, Comput. Commun., **191**, pp. 78-91, Jul. 2022. doi: 10.1016/j.comcom.2022.04.009.

[18] Lekharu, A., Gupta, P., Sur, A. & Patra, M., *Collaborative Video Caching in the Edge Network using Deep Reinforcement Learning*, ACM Transactions on Internet of Things, **5**(3), Jun. 2024, doi: 10.1145/3664613.

[19] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, **518**(7540), pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[20] Tan, X., Wang, S., Xu, X., Zheng, Q., Yang, J. & Chen, S., *DACOD360: Deadline-Aware Content Delivery for 360-Degree Video Streaming Over MEC Networks*, IEEE Trans. Multimedia, **26**, pp. 4168-4182, 2024, doi: 10.1109/TMM.2023.3321439.

[21] Xu, X., Fang, Z., Qi, L. Zhang, X., He, Q. & Zhou, X., *TripRes: Traffic Flow Prediction Driven Resource Reservation for Multimedia IoV with Edge Computing*, ACM Transactions on Multimedia Computing, Communications and Applications, **17**(2), Jun. 2021, doi: 10.1145/3401979.

[22] Li, B., Dong, W., Guan, G., Zhang, J., Gu, T., Bu, J. & Gao, Y., *Queec: QoE-aware edge computing for iot devices under dynamic workloads*, ACM Trans. Sens. Netw., **17**(3), pp. 1-23, Aug. 2021, doi: 10.1145/3442363.

[23] Baccour, E., Erbad, A., Bilal, K., Mohamed, A. & Guizani, M., *PCCP: Proactive Video Chunks Caching and Processing in edge networks*, Future Generation Computer Systems, **105**, pp. 44–60, Apr. 2020, doi: 10.1016/j.future.2019.11.006.

[24] Lei, J., Yang, P., Kong, L., Ma, Y., Lin, D., Chen, G. &Zhao, E., *ACORN+: Adaptive Compression-Reconstruction for Device-Cloud Collaboration Video Services*, ACM Transactions on Autonomous and Adaptive Systems, Nov. 2024, doi: 10.1145/3703000.

[25] Chen, L., Tang, Y., Xia, J., Chen, S., Zheng, C., Lin, H., Wang, W., *Multi-MEC Collaboration for VR Video Transmission: Architecture and Cache Algorithm Design*, Computer Networks, **234**, 109864, Oct. 2023, doi: 10.1016/j.comnet.2023.109864.

[26] Hu, Z., Fang, C., Wang, Z., Tseng, S.M. & Dong, M., *Many-Objective Optimization-Based Content Popularity Prediction for Cache-Assisted Cloud-Edge-End Collaborative IoT Networks*," IEEE Internet Things J., **11**(1), pp. 1190-1200, Jan. 2024, doi: 10.1109/JIOT.2023.3290793.

[27] Kumar, S., Bhagat, L., A, A.F. & Jin, J. *Multi-neural network based tiled 360°video caching with Mobile Edge Computing*, Journal of Network and Computer Applications, **201**, 103342, May 2022, doi: 10.1016/j.jnca.2022.103342.

[28] Davronbekov, D., Pisetskiy, Y., Khayrullaev, A., Khamidov, K., Pulatov, S., Isroilov, J. & Votinov, K., , *Intelligent Approaches to Managing Communication Channels in Hybrid Terrestrial-Satellite Networks with LEO Segment*, Journal of ICT Research and Applications, **20**(1), pp. 1-20, Jun. 2026, doi: 10.5614/itbj.ict.res.appl.2026.20.1.1.

[29] Wang, F., Zhang, C., Wang, F., Liu, J., Zhu, H. & Pang, H., , *DeepCast: Towards Personalized QoE for Edge-assisted Crowdcast with Deep Reinforcement Learning*, IEEE/ACM Transactions on Networking, **28**(3), pp. 1255-1268, Jun. 2020, doi: 10.1109/TNET.2020.2979966.

[30] Guozhang, L., Alfred, R., Pailus, R., Fengchang, X. & Haviluddin, *Scalable and Efficient Student Behavior Prediction using Parallelized Clustering and AHP-weighted KNN*," Journal of ICT Research and Applications, **19**(2), pp. 142-165, Jan. 2025, doi: 10.5614/itbj.ict.res.appl.2025.19.2.3.