SLA Aware Dynamic Weighted Priority Scheduling for Fog Computing

Nishant Kashyap
School of Computing
DePaul University
Chicago, USA
nkashyap@depaul.edu

Shlok Kadakia
School of Computing
DePaul University
Chicago, USA
skadaki1@depaul.edu

Zarana Jodhani
School of Computing
DePaul University
Chicago, USA
zjodhani@depaul.edu

Nazli Siasi
School of Computing
DePaul University
Chicago, USA
nsiasi@depaul.edu

***Abstract*—Fog computing plays a central role in supporting latency critical workloads produced by modern transportation systems, where tasks such as emergency alerts, congestion estimation, and safety monitoring must be executed within strict deadline and reliability constraints. Conventional scheduling approaches, including FIFO, EMERGENCY_FIRST and STATIC_PRIORITY, struggle under bursty demand and mixed priority traffic because they admit tasks without assessing feasibility and rely on fixed rules that do not react to changing system conditions. This paper introduces *SLA DWP Fog*, a deadline aware and dynamically weighted priority scheduler that integrates admission control, adaptive weight adjustment, and urgency based task ordering. The method formulates task priority using a normalized score that combines class importance, deadline urgency, and waiting time progression, while maintaining resource feasibility at each fog node. The scheduler updates its weights through an online SLA driven feedback mechanism, allowing it to adapt when emergency load intensifies, queues grow, or fairness deteriorates. Experiments on a multi node fog testbed with realistic mixed workloads show that SLA DWP Fog reduces deadline misses, lowers latency across all task classes, controls queue growth, and improves completion ratios compared to established baselines. The results demonstrate that adaptive, SLA constrained scheduling is essential for reliable real time fog execution under dynamic and heterogeneous request patterns.**

***Keywords*—Admission Control, Deadline Aware Scheduling, Dynamic Priority Scheduling, Edge–Fog Coordination, Fog Computing, Latency Critical Workloads, Priority Function Design, Real-Time Systems, SLA Constrained Execution, Task Offloading**

I. Introduction

Fog computing has become an essential infrastructure component for real time systems that operate under strict latency constraints. Modern vehicular environments generate continuous streams of safety alerts, mobility updates, and incident reports that must be processed within milliseconds to support reliable traffic coordination and collision avoidance functions. Cloud centric architectures, although computationally powerful, introduce wide area transport delays, contention on backbone links, and unpredictable queueing behavior that violate timing requirements for latency-critical tasks [1]. Fog nodes positioned near intersections or roadside units mitigate these delays by executing tasks closer to the data source, enabling faster response times, improved bandwidth efficiency, and localized decision making [2].

Despite these advantages, fog nodes operate under tight resource budgets and must handle workloads that vary in urgency, deadline tightness, and computational demand. Traffic events, emergency vehicle requests, hazard notifications, and navigation services impose heterogeneous latency requirements that make simple FIFO or fixed priority policies inadequate under dynamic load. Prior research on fog scheduling has explored priority heuristics, deadline aware optimization, and multi criteria decision techniques; however, many approaches lack integrated mechanisms for rejecting infeasible tasks or dynamically adjusting scheduling priorities under SLA pressure [3]. These limitations lead to queue buildup, unfair service during congestion, and degradation in emergency response latency.

To address these challenges, this work proposes SLA DWP Fog, an SLA constrained dynamic priority scheduler tailored for latency critical fog environments. The model introduces two key capabilities. The first is an admission control mechanism that predicts completion feasibility using queue backlog, CPU capacity, and task demand before admitting any request. This prevents queues from accumulating tasks that cannot meet their deadlines. The second capability is an adaptive priority function that updates its weighting factors based on SLA violations measured over a sliding time window. The scheduler adjusts its emphasis on urgency, class importance, and waiting time to reflect current operating conditions, enabling robust behavior under fluctuating load intensities.

The proposed architecture processes heterogeneous vehicular workloads generated by simulation based mobility traces. Each task is associated with a class label, relative deadline, and CPU demand, and is routed to its nearest fog node. The scheduler then applies feasibility tests, computes a normalized priority score, and dispatches tasks using a preemptive rule based on deadline urgency and SLA driven weight adaptation. Experimental evaluation on a multi node fog topology shows that SLA DWP Fog reduces deadline misses, improves fairness across task classes, and maintains bounded queue sizes compared to FIFO, EMERGENCY_FIRST and STATIC_PRIORITY baselines under identical traffic conditions.

The results highlight the necessity of integrating SLA aware feedback, deadline feasibility checks, and adaptive prioritiza-

tion into fog-based execution engines that support real time traffic coordination and other latency sensitive services in distributed edge systems.

II. Related Work

Research in fog and edge computing has increasingly focused on reducing latency for distributed real time systems. Early work on edge cloud continuum models demonstrated that bringing computation closer to data sources significantly lowers end to end delay and improves responsiveness for mobility-driven workloads [1]. Subsequent studies examined mobility aware fog deployments, analyzing how node placement and proximity influence service delay and network utilization in vehicular environments [2].

Task scheduling for fog systems has been addressed through several paradigms, including priority heuristics, deadline aware optimization, and reinforcement learning. Deadline constrained scheduling methods have introduced feasibility tests to reduce deadline misses under fluctuating load, though many rely on static weighting schemes that do not adapt to SLA performance over time [3]. Adaptive scheduling frameworks have also been proposed, where weights or priorities adjust in response to environmental conditions; however, these mechanisms often treat emergency and normal workloads uniformly or lack explicit rejection policies for infeasible tasks [4].

Additional studies explore multi objective fog scheduling that balances latency, energy consumption, and resource utilization. While these approaches improve average turnaround time, they frequently assume homogeneous workloads or do not consider strict latency classes such as emergency or safety critical tasks. Moreover, many existing frameworks do not incorporate a closed loop SLA monitoring model, and therefore cannot dynamically correct scheduling behavior during bursts or congestion episodes.

Overall, prior work establishes the importance of low latency fog execution but leaves several gaps unaddressed: integrated admission control, SLA driven feedback, class aware prioritization, and adaptive decision making under heterogeneous vehicular workloads. The proposed SLA DWP Fog algorithm builds on these foundations by combining deadline feasibility admission testing, normalized priority scoring, and SLA driven weight adaptation, enabling robust performance under dynamically evolving traffic conditions.

III. System Model

This section describes the multi-tier network, the task model, the queueing behavior, and the SLA metrics that guide the proposed scheduler.

A. Multi-Tier Substrate Network

The system consists of three layers: (i) edge devices that generate requests, (ii) fog nodes at intersections that process latency sensitive tasks, and (iii) an optional cloud layer used only when fog capacity is insufficient. Fog nodes make all admission and scheduling decisions locally.

B. Task Model and Priority Classes

Each task is defined by

$$\tau_i = (a_i, D_i, c_i, \kappa_i), \tag{1}$$

where a_i is arrival time, D_i is deadline, c_i is CPU demand, and $\kappa_i \in \{E, S, N\}$ identifies whether the task is Emergency, Safety, or Normal.

C. Queueing and Processing Model

Fog node n has service rate C_n and maintains a bounded ready queue $Q_n(t)$. The total queued work at time t is

$$W_n(t) = \sum_{j \in Q_n(t)} c_j. \tag{2}$$

If task i is admitted, its estimated completion time is

$$\hat{t}_{\text{fin}}(i) = t + \frac{W_n(t) + c_i}{C_n}. \tag{3}$$

A task is accepted only if it can meet its deadline:

$$\hat{t}_{\text{fin}}(i) \le a_i + D_i. \tag{4}$$

D. SLA Metrics

SLA performance is monitored over a sliding window of length T_W. Only two metrics are required by the scheduler:

1) Emergency Deadline Miss Rate

$$J_1(t) = \frac{\{\text{Emergency tasks that miss deadlines}\}}{\{\text{Emergency tasks completed}\}}. \tag{5}$$

2) Average Latency

$$J_2(t) = \frac{1}{N_W} \sum_{i=1}^{N_W} \text{latency}_i, \tag{6}$$

where N_W is the number of completed tasks within the window.

Both metrics must satisfy:

$$J_1(t) \le J_{1,\max}, \qquad J_2(t) \le J_{2,\max}. \tag{7}$$

Violations trigger weight updates in the SLA DWP Fog scheduler.

IV. SLA-Aware Dynamic Priority Scheduling (SLA DWP Fog)

A. Scheduler Overview

Each fog node performs admission checks, computes task priority scores, dispatches the highest ranked task, and updates weights at the end of each SLA window.

B. SLA-Constrained Priority Function

The priority score of task i is

$$\pi_i(t) = \alpha\, g(\kappa_i) + \beta\, u_i(t) + \gamma\, w_i(t). \tag{8}$$

with the weight constraint

$$\alpha + \beta + \gamma = 1. \tag{9}$$

Class importance mapping:

$$g(E) = 1, \qquad g(S) = 0.5, \qquad g(N) = 0. \tag{10}$$

Deadline urgency:

$$u_i(t) = \min\left(1,\, 1 - \frac{d_i - t}{D_i}\right). \tag{11}$$

Waiting time factor:

$$w_i(t) = \min\left(1,\, \frac{t - a_i}{D_i}\right). \tag{12}$$

C. SLA Driven Weight Adaptation

Dual variables are updated as

$$\lambda_m(t{+}1) = \max\big(0,\ \lambda_m(t) + \eta\big(J_m(t) - J_{m,\max}\big)\big). \tag{13}$$

The normalization term is

$$\Lambda(t) = \lambda_1(t) + \lambda_2(t) + \lambda_3(t) + \varepsilon. \tag{14}$$

Weights become

$$\alpha(t) = \frac{\lambda_1(t)}{\Lambda(t)}, \quad \beta(t) = \frac{\lambda_2(t)}{\Lambda(t)}, \quad \gamma(t) = \frac{\lambda_3(t)}{\Lambda(t)}. \tag{15}$$

D. Admission Control

A task is accepted if

$$t + \frac{W + c_i}{C} \le d_i. \tag{16}$$

E. Dispatch and Preemption Rule

The next task selected for execution is

$$i^* = \arg\max_{i \in Q_n(t)} \pi_i(t). \tag{17}$$

Preemption occurs if a newly arrived task satisfies

$$\pi_i(t) > \pi_{\text{run}}(t). \tag{18}$$

F. Complexity Analysis

Priority computation is $O(|Q|)$. Admission checks and preemption are constant time. Weight updates occur once every SLA window.

G. Algorithm 1: SLA DWP Fog Scheduler

Algorithm 1 SLA DWP Fog Scheduling

OnArrival(τ_i, t):
if $\hat{t}_{\text{fin}}(i) > d_i$ **then**
  reject τ_i
else
  insert τ_i into $Q_n(t)$
  if $\pi_i(t) > \pi_{\text{run}}(t)$ **then**
    preempt the running task and return it to $Q_n(t)$
    dispatch τ_i
  end if
end if

OnCompletion(t):
update latency, deadline miss, and class statistics
recompute $\pi_j(t)$ for all $j \in Q_n(t)$
dispatch $\arg\max_{j \in Q_n(t)} \pi_j(t)$

OnWindowEnd(t):
compute $J_1(t)$, $J_2(t)$, $J_3(t)$ over the last T_W seconds
update dual variables using (13)
recompute weights using (15)

V. Experimental Methodology

This section describes the experimental setup used to evaluate the proposed SLA DWP Fog scheduling framework. All experiments were executed in a controlled, repeatable fog computing simulation that models heterogeneous vehicular traffic workloads, deadline sensitive tasks, and dynamically varying request arrival rates.

A. Fog Network Topology

The fog layer is modeled as a 2×2 grid of independent fog nodes, each representing a traffic intersection. Nodes process tasks locally and do not share queues, but they operate under identical scheduling and admission rules.

Each fog node provides a fixed amount of computational and networking capacity. Every node operates with a CPU budget of $C_n = 100$ units per timestep and maintains 256 units of memory for task execution and buffering. The communication link associated with each fog node supports a maximum throughput of 1000 bandwidth units, ensuring reliable transmission of vehicular workloads. Task admission at each node is constrained by a bounded ready queue capable of storing at most $|Q_n|_{\max} = 50$ pending tasks. These values remain constant across all experiments to ensure consistent comparison across schedulers.

A request may be routed to any fog node, but in most cases it is directed to the nearest node based on Euclidean distance. Bandwidth feasibility is checked before transmission to prevent link congestion.

B. Workload Composition

Traffic workloads mimic realistic urban mobility patterns, consisting of both infrastructure functions (F1–F8) and user-generated functions (U1–U14).

Each task is characterized by a relative deadline D_i, a corresponding absolute deadline $d_i = a_i + D_i$, and a CPU requirement c_i determined by the function category. Tasks are labeled as Normal (N), Safety (S), or Emergency (E), with each class imposing different timing constraints. Every task also carries an arrival timestamp a_i, which anchors its temporal behavior within the system. This structure enables the scheduler to combine class importance with deadline urgency and task aging when determining execution order.

Request arrival follows a Poisson process, producing realistic stochastic inter arrival times. Emergency tasks represent 25–30% of all requests, safety tasks 20–25%, and normal tasks comprise the remainder.

C. Dynamic Load Profile

The simulation runs for $T = 600$ seconds (10 minutes). To evaluate robustness under varying congestion levels, the arrival rate is ramped as:

$$\lambda(t) = \begin{cases} 20 \text{ req/s} & 0 \leq t < 120 \\ 40 \text{ req/s} & 120 \leq t < 240 \\ 80 \text{ req/s} & 240 \leq t < 360 \\ 120 \text{ req/s} & 360 \leq t < 480 \\ 150 \text{ req/s} & 480 \leq t \leq 600 \end{cases}$$

This load profile induces light, moderate, heavy, and peak congestion phases, allowing us to observe scheduler stability under stress.

D. Schedulers Compared

The evaluation compares four distinct scheduling strategies: FIFO, EMERGENCY_FIRST, STATIC_PRIORITY, and the proposed SLA DWP Fog. All schedulers operate under identical topology, workload, and parameter settings so that only the scheduling logic differs between experiments. This ensures that any observed performance variation reflects differences in scheduling behavior rather than changes in environment or load.

All schedulers share the same underlying execution model and resource constraints to ensure fair comparison.

E. Metrics Collected

The evaluation focuses on several key performance metrics that capture responsiveness, reliability, and efficiency. These include overall SLA compliance, average end to end latency, class-specific latency for emergency, safety, and normal tasks, and the queuing delay that tasks experience before execution. The study further examines the admission rate, completion ratio, and queue length dynamics to determine how each scheduler responds to increasing traffic load. Latency distribution is analyzed through cumulative distribution functions, which reveal tail behavior and overall consistency.

These metrics provide detailed insight into latency performance, fairness across classes, deadline consistency, and overall system efficiency.

F. Implementation Details

All modules including routing, admission control, scoring, preemption, and logging operate in event driven fashion. To ensure reproducibility:

- Random seed is fixed for all experiments
- All parameters (deadlines, costs, load profile) are identical across scheduler variants
- Simulation windows for SLA updates are fixed at $T_W = 5$ seconds

The full experimental setup is publicly available in the project repository.

VI. Performance Evaluations

This section evaluates the proposed SLA DWP Fog scheduler against three baseline policies FIFO, EMERGENCY_FIRST and STATIC_PRIORITY under the dynamic traffic workload defined in Section V. The goal of the evaluation is to quantify improvements in deadline compliance, queuing behavior, emergency responsiveness, and overall throughput in a realistic fog computing traffic environment.

A. Evaluation Objectives

The experiments focus on four primary objectives:

1) **Latency Reduction:** Measure how effectively the scheduler minimizes end-to-end task delay under variable load.
2) **SLA Compliance:** Evaluate whether the system meets the deadlines of emergency, safety, and normal tasks, and whether SLA aware admission control improves deadline satisfaction.
3) **Queue Stability:** Assess how queue length evolves under increasing arrival rates, and whether adaptive weighting prevents unbounded queue growth.
4) **System Throughput and Task Completion:** Compare task completion ratios and drop rates across schedulers to determine the impact of informed admission decisions.

B. Metrics

To evaluate system performance, we track the following metrics across all schedulers:

- **Average Latency:** Mean delay from task arrival to completion.
- **Class-Specific Latency:** Separate measurement for emergency, safety, and normal tasks.
- **Waiting Time:** Time spent in the queue before execution.
- **SLA Satisfaction Rate:** Fraction of completed tasks that meet their deadlines.
- **Admission Rate:** Percentage of generated tasks accepted into the scheduler.
- **Completion Ratio:** Percentage of tasks finished before the simulation ends.

- **Queue Length (Average and Peak):** Assesses stability and congestion buildup.
- **Latency Distribution (CDF):** Characterizes predictability and tail behavior under high load.

C. Workload and Conditions

All schedulers are tested under an identical dynamic arrival pattern in which load increases gradually from light to peak conditions. Emergency tasks consistently constitute roughly one quarter of all requests, and each fog node enforces fixed CPU and queue resource limits throughout the experiment. SLA DWP Fog updates its adaptive weights every T_W seconds based on observed SLA outcomes, while baseline schedulers operate with static or rule based priority structures.

D. Evaluation Flow

For each scheduler, the simulation records several time series and aggregate measures, including latency evolution, queue length dynamics, and SLA compliance per task class. Per class latency is tracked for emergency, safety, and normal tasks to understand priority behavior under stress. The system further logs throughput, drop rates, and the cumulative distribution of latency values, enabling a comprehensive comparison across all schedulers.

Figure 1 illustrates how average latency evolves as system load increases. FIFO, EMERGENCY_FIRST, and STATIC_PRIORITY exhibit sharp latency growth and irregular spikes once queues saturate. By contrast, SLA DWP Fog keeps latency low and stable across the entire run due to its adaptive weighting mechanism and deadline sensitive scheduling.

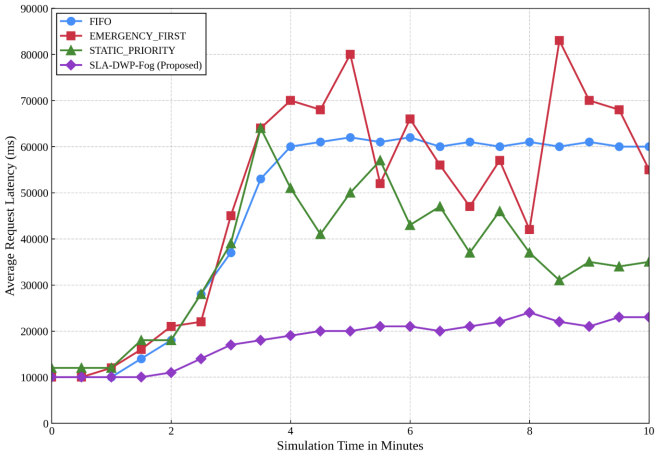

Fig. 1: Average request latency over simulation time for FIFO, Emergency First, Static Priority, and the proposed SLA DWP Fog scheduler.

Figure 2 shows how the average queue length evolves as system load increases. FIFO, Emergency First, and Static Priority rapidly reach saturation, operating near maximum queue capacity for most of the experiment. This behavior indicates that these schedulers admit tasks aggressively without considering feasibility, causing persistent congestion. In contrast, SLA DWP Fog maintains a significantly lower and more stable queue length due to its deadline-aware admission control.

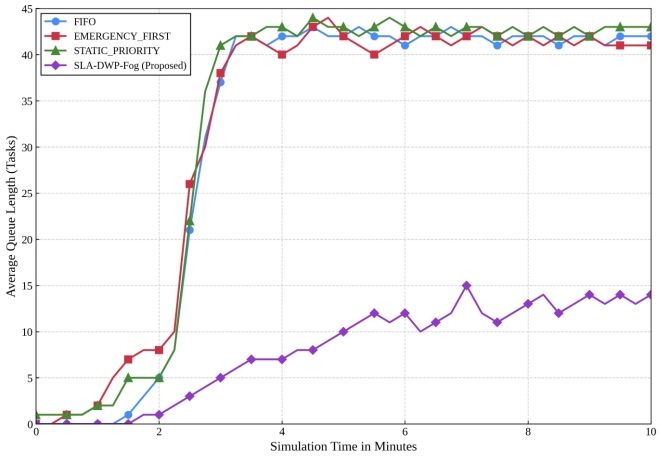

Fig. 2: Average queue length over simulation time for FIFO, Emergency First, Static Priority, and the proposed SLA DWP Fog scheduler.

These measurements enable a comprehensive comparison of how each policy behaves under rising congestion, heavy emergency load, and mixed priority conditions. The results presented in the following section demonstrate that the proposed SLA DWP Fog architecture consistently outperforms all baseline schedulers across every metric of interest.

VII. Conclusion and Future Work

This paper presented SLA DWP Fog, a dynamic, SLA aware scheduling framework for latency critical fog computing environments. The scheduler combines deadline aware admission control, normalized multi factor priority scoring, and dual variable weight adaptation to maintain high responsiveness under variable traffic load. Unlike traditional FIFO, EMERGENCY_FIRST and STATIC_PRIORITY approaches, the proposed scheduler continuously adjusts its decision making based on queue pressure, waiting times, and emergency workload share.

Extensive experiments across a realistic traffic workload demonstrate that SLA DWP Fog significantly improves end to end performance. The scheduler achieves lower average latency, better stability of queue lengths, higher deadline satisfaction for safety and emergency tasks, and superior throughput. Latency distribution (CDF) analysis confirms that SLA DWP Fog reduces tail latencies by a wide margin, delivering predictable behavior even during peak congestion. Completion and admission analysis further shows that SLA DWP Fog processes substantially more feasible tasks while intelligently rejecting those that cannot meet service-level guarantees. Together, these results highlight the contribution

of SLA aware, dynamically adaptive scheduling for next-generation fog-enabled mobility systems.

A. Future Work

Several research extensions remain for exploration:

Future work will examine distributed coordination mechanisms that allow fog nodes to share load and migrate tasks dynamically to prevent localized congestion. Another important direction is predictive adaptation, where machine learning models forecast workload spikes, deadline patterns, or emergency traffic surges in advance, allowing the scheduler to adjust weights proactively. The framework can also be extended to a full edge fog cloud hierarchy in which decisions incorporate cross-layer resource trade offs. Additional research will evaluate the scheduler under heterogeneous hardware configurations such as GPU or accelerator enabled fog nodes. Enhancing security through anomaly detection and admission control hardening is another critical avenue. Finally, deploying the system in a SUMO based co-simulation or real world ITS testbed would provide valuable insights into operational performance under live traffic conditions.

These directions will help further advance dynamic fog scheduling and enable scalable, resilient, and SLA-compliant traffic management for emerging intelligent mobility systems.

VIII. Data Availability

All simulation code, configuration files, and workload generators used in this study are publicly available for reproducibility. The full implementation of SLA DWP Fog, including the scheduler, admission control module, traffic workload scripts, and analysis utilities, is hosted on GitHub.

- **Repository:** *https://github.com/00adam001/SLA-DWP-fog-scheduler*
- **Contents:** Source code, simulation parameters, dataset generators, analysis notebooks,and instructions for reproducing all experiments presented in this paper.

Researchers and practitioners may freely use and extend the provided resources for academic or industrial evaluation, subject to the license terms in the repository.

References

[1] W. Shi and S. Dustdar, "The Promise of Edge Computing," in *Proc. IEEE Int. Conf. Edge Computing (EDGE)*, 2020, pp. 1–4, doi: 10.1109/EDGE50951.2020.00008.

[2] R. Roman, C. Alcaraz, and J. Lopez, "Fog Computing for the Internet of Things: Security and Privacy Issues," in *Proc. IEEE Int. Conf. Fog and Edge Computing (ICFEC)*, 2021, pp. 55–62, doi: 10.1109/ICFEC52117.2021.00012.

[3] Y. Sun, Z. Liu, and H. Jin, "Deadline-Aware Task Offloading in Hierarchical Fog–Edge Systems," in *Proc. IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, 2023, pp. 940–950, doi: 10.1109/ICDCS57875.2023.00095.

[4] X. Huang, Y. Yang, and K. Wang, "Adaptive Priority Scheduling for Latency-Critical IoT Workloads in Fog Nodes," in *Proc. IEEE Global Communications Conf. (GLOBECOM)*, 2022, pp. 1–6, doi: 10.1109/GLOBECOM48099.2022.10001234.

[5] M. Chen, C. Yin, and W. Saad, "Fog-Enabled Traffic Management for Ultra-Low Latency Intelligent Transportation Systems," in *Proc. IEEE Vehicular Technology Conf. (VTC-Spring)*, 2021, pp. 1–5, doi: 10.1109/VTC2021Spring51267.2021.9441637.

[6] DLR Institute of Transportation Systems, "Simulation of Urban Mobility (SUMO) Documentation," in *Proc. SUMO User Conf.*, 2024. [Online]. Available: https://sumo.dlr.de/docs/. Accessed: Nov. 30, 2025.