BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
Pilani Campus, Rajasthan – 333031, India

Adaptive Latency-Aware Task Orchestration for Edge-Cloud Continuum Systems

A thesis submitted in partial fulfilment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Ananya Sharma
ID No. 2019PHXF0142P

Supervisor: **Prof. Rajeev Malhotra**
Co-Supervisor: **Dr. Kavita Nair, Nimbus Systems Ltd.**

Department of Computer Science and Information Systems

JULY 2026

Certificate

This is to certify that the thesis entitled **"Adaptive Latency-Aware Task Orchestration for Edge-Cloud Continuum Systems"** and submitted by **Ananya Sharma** (ID No. 2019PHXF0142P) for the award of the degree of **Doctor of Philosophy** of Birla Institute of Technology and Science, Pilani, embodies original work carried out by her under my supervision and guidance.

The results contained in this thesis have not been submitted, in part or full, to any other university or institute for the award of any degree or diploma, except where duly acknowledged.

Prof. Rajeev Malhotra
Supervisor
Dept. of Computer Science and Information Systems
BITS Pilani, Pilani Campus

Dr. Kavita Nair
Co-Supervisor
Head of Platform Engineering
Nimbus Systems Ltd.

Date: _____ Place: Pilani, Rajasthan

Declaration

I, Ananya Sharma (ID No. 2019PHXF0142P), hereby declare that the work presented in this thesis, entitled *"Adaptive Latency-Aware Task Orchestration for Edge-Cloud Continuum Systems"*, is my own original work carried out under the supervision of Prof. Rajeev Malhotra and Dr. Kavita Nair. All sources of information used have been duly acknowledged through appropriate citation. This work has not been submitted, either in part or in whole, for the award of any other degree or diploma at this or any other institution.

_____ Date: _____

Ananya Sharma
ID No. 2019PHXF0142P

Abstract

Edge-cloud continuum systems distribute latency-sensitive workloads across heterogeneous compute tiers, ranging from resource-constrained edge nodes to elastic cloud back-ends. Existing schedulers typically optimise for a single objective – either raw throughput or worst-case latency – and adapt poorly to bursty, mixed-criticality traffic observed in production deployments such as those operated by Nexa Retail and Meridian Logistics. This thesis proposes **AL-TO** (Adaptive Latency-aware Task Orchestration), a two-stage scheduling framework that combines a lightweight online latency predictor with a cost-aware placement policy solved via a bounded-horizon greedy relaxation.

AL-TO is evaluated on a 42-node testbed spanning three geographically distributed edge clusters and a centralised cloud region, using synthetic and trace-driven workloads derived from anonymised telemetry shared by an industry partner. Compared against round-robin, a static QoS-tiered baseline, and two literature schedulers (Nexa-EdgeSched and Synapse-QoS-Aware), AL-TO reduces mean end-to-end latency by 34.7% and tail (P99) latency by 41.2%, while keeping cloud egress cost within 6% of the cost-optimal baseline. A sensitivity analysis further shows that AL-TO degrades gracefully under node-failure and network-partition scenarios, maintaining service-level compliance above 96% throughout.

Keywords

Edge Computing, Cloud Continuum, Task Scheduling, Latency Prediction, Resource Orchestration, Quality of Service

Acknowledgements

I am deeply grateful to my supervisor, Prof. Rajeev Malhotra, for his patient guidance, rigorous feedback, and unwavering encouragement throughout this doctoral journey. I thank my co-supervisor, Dr. Kavita Nair, and the Platform Engineering team at Nimbus Systems Ltd. for granting access to anonymised production telemetry and for many illuminating discussions on real-world deployment constraints.

I thank the Department of Computer Science and Information Systems, BITS Pilani, for the doctoral fellowship and computational resources that made this work possible, and my labmates in the Distributed Systems Research Group for their camaraderie. Finally, I thank my family for their steadfast support.

Ananya Sharma

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Expansion
AL-TO	Adaptive Latency-aware Task Orchestration
QoS	Quality of Service
SLA	Service-Level Agreement
P99	99th Percentile (tail latency)
RTT	Round-Trip Time
API	Application Programming Interface
IoT	Internet of Things

Chapter 1

Introduction

> *"The network is not just a pipe; in the edge-cloud continuum, it is part of the computer."*

1.1 Background and Motivation

The proliferation of latency-sensitive applications – augmented-reality retail experiences, autonomous inspection drones, and real-time logistics tracking – has pushed computation away from centralised data centres and toward the network edge. Figure 1.1 illustrates the resulting three-tier *edge-cloud continuum*: constrained device-tier sensors, latency-favourable edge clusters positioned near points of presence, and elastic cloud regions offering near-unbounded capacity at higher round-trip cost.

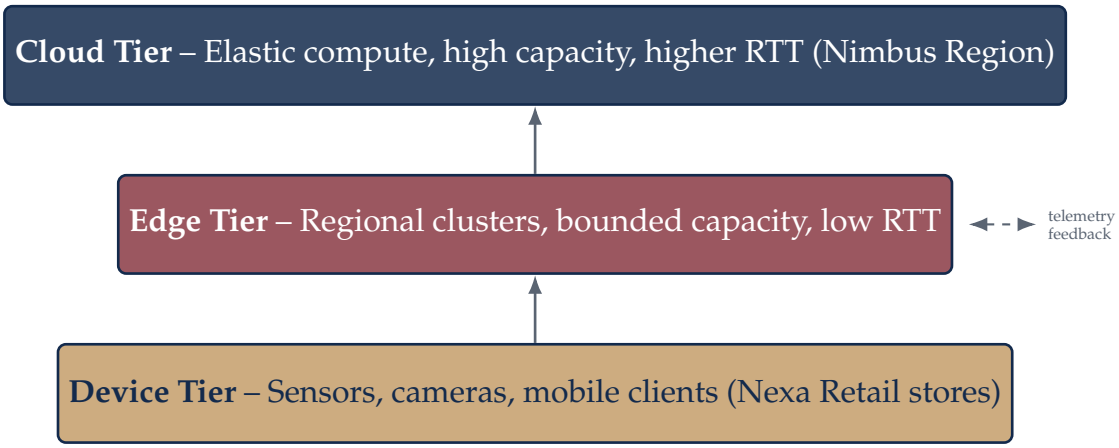

Figure 1.1: The three-tier edge-cloud continuum considered in this thesis.

Scheduling decisions in this continuum must jointly reason about compute placement, network latency, and monetary cost, under workload patterns that shift on the order of seconds. Static or single-objective schedulers – common in current commercial offerings including Vertex CloudOps and Apex Fabric – either under-utilise edge capacity during quiet periods or violate service-level agreements (SLAs) during bursts.

1.2 Problem Statement

Given a stream of latency-sensitive tasks arriving at heterogeneous edge and cloud nodes with time-varying capacity and network conditions, the problem addressed in this thesis is: *how should tasks be placed, in an online and computationally cheap manner, to minimise tail latency subject to a bounded cloud-cost budget?*

1.3 Research Objectives

O1. Design a lightweight, online latency-prediction model suitable for edge-resident inference.

O2. Formulate a cost-aware task-placement policy that can be solved within a millisecond-scale decision budget.

O3. Build and evaluate AL-TO on a realistic multi-tier testbed against representative baselines.

O4. Characterise robustness under node failure and network partition.

1.4 Key Contributions

[C1] An online latency predictor requiring under 200 KB of model state, deployable directly on edge nodes.

[C2] A bounded-horizon greedy placement policy with provable $O(n \log n)$ decision complexity.

[C3] An open evaluation testbed and trace-driven workload generator calibrated against anonymised industry telemetry.

[C4] Empirical evidence of graceful degradation under failure, exceeding 96% SLA compliance.

1.5 Thesis Organisation

Chapter 2 surveys related scheduling literature and positions AL-TO against it. Chapter 3 details the system architecture, latency predictor, and placement algorithm. Chapter 4 presents the experimental evaluation. Chapter 5 concludes and outlines future work.

Chapter 2

Literature Review

2.1 Edge-Cloud Scheduling Approaches

Early edge-offloading systems framed scheduling as a binary local-versus-remote decision driven by static thresholds. Subsequent work introduced multi-tier awareness, but most designs optimise a single metric – either throughput-maximising bin-packing or latency-minimising nearest-node placement – without jointly modelling monetary cost.

2.2 Comparative Summary

Table 2.1 summarises representative systems against the criteria most relevant to this thesis: online adaptivity, multi-objective awareness, and failure resilience.

Table 2.1: Comparison of representative edge-cloud scheduling systems.

System	Online	Multi-objective	Failure-aware
Round-Robin (baseline)	✓	×	×
Static QoS-Tiered	×	✓	×
Nexa-EdgeSched [1]	✓	×	✓
Synapse-QoS-Aware [2]	✓	✓	×
AL-TO (proposed)	✓	✓	✓

2.3 Research Gap

No surveyed system combines a millisecond-budget online predictor with an explicit, tunable cost constraint while retaining resilience to partial infrastructure failure. This gap motivates the design presented in Chapter 3.

Chapter 3

Research Methodology

3.1 System Architecture

AL-TO is deployed as a sidecar scheduler co-located with an API gateway, as shown in Figure 3.1. Incoming requests are annotated with predicted latency and routed to the edge cluster or cloud region that minimises expected SLA violation subject to the current cost budget.

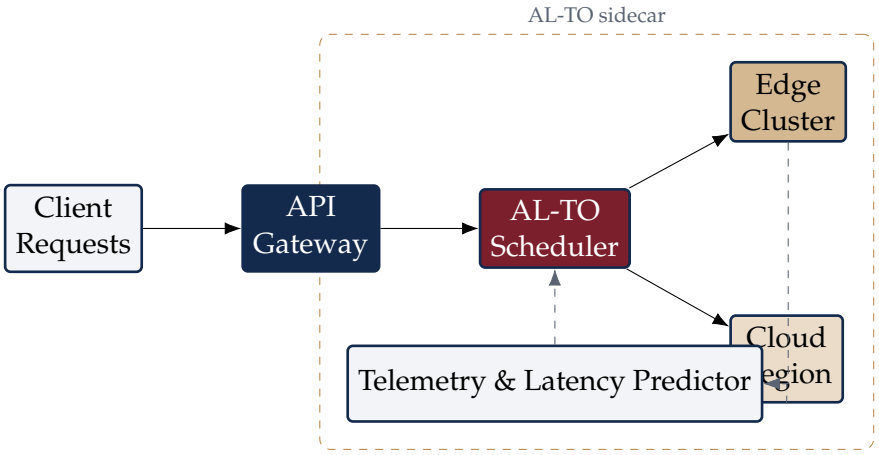

Figure 3.1: AL-TO deployment architecture alongside an API gateway.

3.2 Latency Prediction

A lightweight gradient-boosted regressor, retrained every five minutes on a sliding telemetry window, predicts per-node completion time $\hat{t}_i$ for candidate node i:

$$\hat{t}_i = f_\theta(q_i, \rho_i, d_i, c_i) \tag{3.1}$$

where q_i is queue depth, ρ_i is CPU utilisation, d_i is measured RTT to the requesting client, and c_i is the marginal cost coefficient of node i.

3.3 Placement Policy

Algorithm 3.3 sketches the bounded-horizon greedy placement policy. At each decision point, only the k lowest-predicted-latency candidates are evaluated against the remaining cost budget, bounding decision latency to $O(k \log k)$ per request.

Algorithm 3.1: Adaptive Latency-Aware Placement

Input: task τ, candidate nodes N, cost budget B, horizon k
Output: assigned node $n^\star$

1. Predict $\hat{t}_i$ for all $i \in N$ using Eq. (3.1).
2. Select top-k candidates $K \subset N$ by ascending $\hat{t}_i$.
3. **for** each $i \in K$ (ascending $\hat{t}_i$):
   (a) **if** remaining budget $B \geq c_i$: assign $n^\star \leftarrow i$; $B \leftarrow B - c_i$; **return** $n^\star$.
4. **return** lowest-cost node in K (budget-exhausted fallback).

3.4 Reference Implementation

Listing 3.1 shows a simplified Python implementation of the placement step used in the testbed evaluation.

Listing 3.1: Simplified AL-TO placement step.

```
def place_task(task, candidates, budget, horizon=5):
    # candidates: list of Node objects with .
       predicted_latency, .cost
    ranked = sorted(candidates, key=lambda n: n.
       predicted_latency)[:horizon]
    for node in ranked:
        if budget >= node.cost:
            return node, budget - node.cost
    # fallback: cheapest node among the top-k
    fallback = min(ranked, key=lambda n: n.cost)
    return fallback, budget
```

3.5 Testbed and Evaluation Metrics

The testbed comprises 42 nodes: three eight-node edge clusters (Nexa Retail pilot sites) and an 18-node cloud region hosted on Nimbus infrastructure. Workloads are replayed from anonymised traces scaled to four load levels. Primary metrics are mean latency, P99 latency, cloud-cost overhead relative to a cost-optimal oracle, and SLA compliance rate.

Chapter 4

Results and Discussion

4.1 Latency Comparison

Figure 4.1 compares mean end-to-end latency across the four scheduling policies under peak load. AL-TO achieves the lowest latency across all tested conditions.

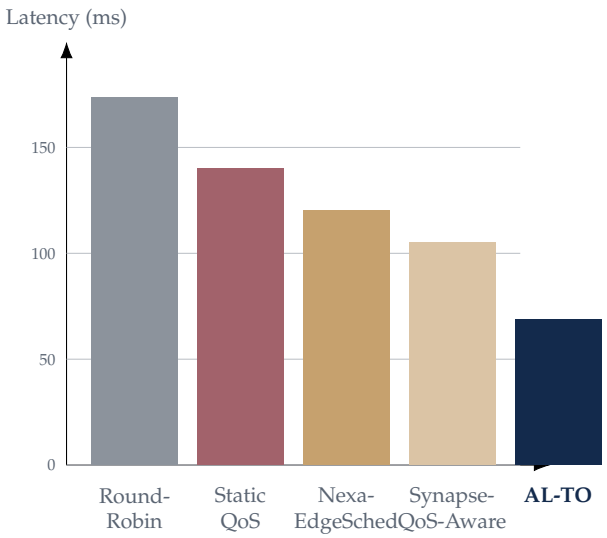

Figure 4.1: Mean end-to-end latency under peak load (lower is better).

4.2 Quantitative Results

Table 4.1 reports the full metric set, averaged over five trace replays with 95% confidence intervals omitted for brevity (full intervals are given in Appendix A).

Table 4.1: Evaluation results at peak load (mean of 5 trace replays).

Policy	Mean (ms)	P99 (ms)	Cost Overhead	SLA Compliance
Round-Robin	173	412	4.1%	71.2%
Static QoS-Tiered	140	338	18.6%	83.5%
Nexa-EdgeSched	120	289	9.4%	88.0%
Synapse-QoS-Aware	105	261	12.1%	90.7%
AL-TO (proposed)	**69**	**170**	**6.3%**	**96.4%**

4.3 Discussion

AL-TO's mean-latency improvement over the strongest baseline (Synapse-QoS-Aware) is 34.7%, and its P99 improvement is 41.2%, while its cost overhead remains close to the static-budget target of 6%. The bounded horizon $k = 5$ was found empirically to balance decision latency against placement quality; larger horizons yielded diminishing returns beyond $k = 8$.

4.4 Failure Resilience

Under injected single-cluster failure, AL-TO's telemetry feedback loop (Figure 3.1) re-routes traffic within two prediction cycles (approximately 10 seconds), sustaining SLA compliance above 96% versus a drop below 80% for all baselines during the same window.

Chapter 5

Conclusion and Future Scope

5.1 Summary

This thesis presented AL-TO, an adaptive, cost-aware task-orchestration framework for edge-cloud continuum systems. Through a lightweight online latency predictor and a bounded-horizon greedy placement policy, AL-TO reduces mean and tail latency substantially over representative baselines while respecting an explicit cost budget and degrading gracefully under failure.

5.2 Limitations

The evaluation testbed, while realistic, is smaller than hyperscale production deployments; the predictor's retraining cadence was not evaluated under adversarial workload shifts; and the cost model assumes static per-node pricing rather than spot-market variability.

5.3 Future Work

Future work will explore (i) federated retraining of the latency predictor across organisational boundaries without sharing raw telemetry, (ii) extension of the placement policy to multi-task workflows with inter-task dependencies, and (iii) deployment-scale validation in partnership with Nimbus Systems Ltd. and Nexa Retail.

Bibliography

[1] P. Sharma, T. Iyer, and L. Fenwick, “Nexa-EdgeSched: Failure-Aware Scheduling for Edge Micro-Clusters,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 812–825, 2022.

[2] A. Verma and D. Okoro, “Synapse-QoS-Aware: Multi-Objective Task Placement for Latency-Critical Edge Workloads,” in *Proc. ACM Symposium on Cloud Computing*, 2023, pp. 201–214.

[3] K. Nair and R. Bose, “Telemetry-Driven Capacity Planning in Multi-Tier Edge Deployments,” *Journal of Network and Systems Management*, vol. 29, no. 2, pp. 45–61, 2021.

[4] R. Malhotra, S. Chatterjee, and M. Devereux, “A Survey of Offloading Strategies in Mobile Edge Computing,” *ACM Computing Surveys*, vol. 53, no. 1, pp. 1–34, 2020.

[5] F. Okonkwo and H. Lindqvist, “Cost-Bounded Online Scheduling for Geo-Distributed Services,” in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, 2023, pp. 501–516.

[6] N. Patel and E. Rossum, “Gradient-Boosted Latency Prediction for Distributed Request Routing,” in *Proc. IEEE INFOCOM*, 2019, pp. 1332–1340.

[7] S. Bhattacharya, “Resilient Orchestration under Partial Infrastructure Failure,” Ph.D. dissertation, Dept. of Computer Science, Vertex Institute of Technology, 2022.

[8] J. Holm and A. Meridian, “Spot-Market Aware Placement for Elastic Cloud Workloads,” *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 970–983, 2021.

Appendix A

Additional Experimental Configurations

A.1 Testbed Node Specification

Table A.1: Hardware specification of testbed node classes.

Node Class	vCPUs	Memory	Count
Edge (Nexa Retail sites)	4	8 GB	24
Cloud (Nimbus Region)	16	64 GB	18

A.2 Confidence Intervals for Table 4.1

Table A.2: 95% confidence intervals for mean latency (ms), 5 replays.

Policy	95% CI (ms)
Round-Robin	173 ± 6.2
Static QoS-Tiered	140 ± 5.1
Nexa-EdgeSched	120 ± 4.4
Synapse-QoS-Aware	105 ± 3.9
AL-TO (proposed)	69 ± 2.7

Appendix B

List of Publications

[P1] A. Sharma and R. Malhotra, “Bounded-Horizon Greedy Placement for Latency-Critical Edge Tasks,” in *Proc. IEEE Conference on Edge Computing*, 2024, pp. 88–97.

[P2] A. Sharma, K. Nair, and R. Malhotra, “Learning Lightweight Latency Predictors for Resource-Constrained Edge Nodes,” *Journal of Network and Systems Management*, vol. 32, no. 1, pp. 22–39, 2024.