International Journal of Advanced Engineering and Technology Research
ISSN: 3106-8677 (Print), ISSN: 3106-8685 (Online) | Vol. 3, No. 3, 2026

A Queueing-Predictive Task Offloading Framework for Joint Latency and Energy Optimization in Smart City Edge–Cloud IoT Networks

Uma Bennett [1, *], Serena Zhou [2], Daniel Jackson [1]

[1] School of Computing, University of Georgia, USA

[2] Department of Computer Science, Virginia Tech, USA

[*] **Corresponding author**: (Email: f039215@uga.edu)

Abstract: Smart-city Internet of Things (IoT) platforms run safety-critical alarms, near-real-time video analytics and delay-tolerant batch jobs on a shared edge–cloud infrastructure. Keeping enough edge capacity powered for peak demand wastes energy for most of the day, whereas aggressive consolidation pushes work to distant clouds and erodes latency guarantees. This paper presents PLEO, a queueing-predictive latency–energy offloading framework that couples two control loops. A slow loop forecasts zone-level demand with quantile gradient-boosted trees and sizes the powered capacity of every edge site by minimizing a joint latency–energy cost under an M/G/m waiting-time chance constraint, waking processing units ahead of demand and releasing them only beyond a break-even horizon. A fast loop predicts the completion time and marginal energy of every task at the local site, at neighbouring sites and in regional clouds, and selects the cheapest destination that meets the slice deadl ine. PLEO builds on a reference edge–cloud synergy architecture whose topology, slices and reactive QoS orchestration rules are implemented as a baseline. In simulations driven by the real PeMSD8 loop -detector dataset (170 detectors in 10 edge zones, 84 million tasks per week), PLEO lowers infrastructure energy per task by 9.2% relative to the reactive orchestration baseline, 11.0% relative to edge-only and 7.6% relative to cloud-only processing, meets every critical-slice deadline and completes 99.82% of all tasks on time. Ablations show that queueing-aware capacity planning, rather than forecast accuracy, is the main source of the gain.

Keywords: Edge–cloud computing, task offloading, energy efficiency, capacity provisioning, queueing theory, load forecasting, smart cities.

1. Introduction

Smart-city platforms increasingly run on dense Internet of Things (IoT) deployments in which roadside sensors, cameras and wearables feed services ranging from emergency response to traffic management [1]. Edge computing brings processing close to these data sources and closes response loops that a remote cloud cannot close in time [2–4], and recent surveys identify the coordination of edge and cloud resources as a central open issue for urban deployments [5], [6]. The services that share this infrastructure are strongly heterogeneous: an alarm from a patient monitor must be handled within tens of milliseconds, video analytics for incident detection tolerates a few hundred milliseconds, and bulk synchronization can wait for seconds. Network slicing gives each class an isolated logical network with its own service-level objectives [7], but the compute layer still has to decide where every task runs and how much edge capacity is kept powered.

These two decisions pull in opposite directions. Servers are far from energy-proportional, and an idle accelerator node still draws a large fraction of its peak power [8, 9]. Urban demand, in contrast, is strongly periodic: in the loop-detector data used in this paper, the mean traffic of the afternoon peak hour is about 4.7 times that of the night-time trough. Sizing edge sites for the peak therefore wastes energy for most of the day, whereas consolidating work in regional clouds adds wide-area round trips that critical slices cannot afford. Energy and latency are thus inseparable: a powered edge unit pays off only if it is kept sufficiently busy, and it is kept busy only if the offloading policy routes the right work to it.

The computation-offloading literature offers effective per-task decision schemes, including game-theoretic offloading [10], Lyapunov-based dynamic offloading [11], joint offloading and resource allocation across multiple edge servers [12], collaborative edge–cloud latency minimization [13] and deep-reinforcement-learning controllers [14]. Most of these works assume a fixed set of powered servers and optimize device-side energy. Data-center right-sizing [15], [16] and edge autoscaling [17], on the other hand, adapt the number of active servers but abstract away where individual tasks are executed. Smart-city deployments need a controller that sizes the powered capacity of each edge site with an explicit slice-level latency risk, places each task using a prediction of its latency and energy at all candidate sites, and is validated on real city demand.

This paper proposes PLEO, a queueing-predictive latency–energy offloading framework built on two coupled control loops. Its contributions are as follows.

A two-timescale control framework in which a slow loop forecasts zone demand with quantile gradient-boosted trees and chooses the powered capacity of each edge site by minimizing a joint latency–energy cost under an M/G/m waiting-time chance constraint, with lead-time wake-up and break-even scale-in, while a fast loop predicts every task's completion time and marginal energy at local, neighbouring and cloud sites and selects the cheapest destination that satisfies the slice deadline and forecast-based admission guards.

A trace-driven evaluation built on the public PeMSD8

loop-detector dataset, which drives a ten-zone city workload, combined with an event-driven simulator that models processing-unit power states, preemptive slice priorities, metro and wide-area delays, and transport energy.

A comparison with cloud-only and edge-only processing, a reactive slice-aware orchestration policy and a myopic joint optimizer, including trade-off, load-stress, wake-up-time and ablation studies. PLEO reduces infrastructure energy per task by 9.2% relative to the reactive orchestration policy while completing 99.82% of all tasks on time, and the ablations show that queueing-aware capacity planning, rather than forecast accuracy, drives this gain.

Section II reviews related work, Section III presents the system model, Section IV describes PLEO, Sections V and VI report the experimental setup and results, and Section VII concludes the paper.

2. Related Work

2.1. Computation Offloading in Edge–Cloud Systems

Surveys of mobile edge computing cover the communication and computation models that underpin offloading decisions [3, 4]. Chen et al. formulate multi-user offloading as a game with a Nash-equilibrium solution [10]; Mao et al. use Lyapunov optimization to schedule offloading under energy-harvesting constraints [11]; Tran and Pompili jointly optimize offloading decisions, transmit power and server resources across multiple edge servers [12]; and Ren et al. split workloads between edge and cloud to minimize latency [13]. Learning-based schemes such as DROO approximate the combinatorial offloading decision with a deep network trained online [14]. These approaches either keep the server capacity fixed or focus on device-side energy. PLEO targets infrastructure energy, i.e., the idle and dynamic power of edge sites together with cloud and transport energy, and treats the powered capacity as a decision variable.

2.2. Capacity Provisioning and Energy Proportionality

Because idle servers consume a large share of their peak power [8, 9], switching capacity on and off is one of the most effective energy levers. Lin et al. characterize the offline structure of dynamic right-sizing under switching costs and derive a 3-competitive lazy online algorithm [15], and Gandhi et al. show that setup times make reactive scaling fragile unless spare capacity is managed carefully [16]. Xu et al. learn joint offloading and autoscaling policies for a renewable-powered edge site with reinforcement learning [17]. PLEO borrows the switching-cost reasoning for its scale-in rule but derives the required capacity from queueing theory [18], [19], which yields an explicit per-site latency guarantee instead of a utilization threshold. PRESAGE provides a valuable systems-level example of linking predictive uncertainty with elastic resource decisions and service-level objectives [29].

2.3. Forecasting Urban Demand

Traffic data from the Caltrans Performance Measurement System (PeMS) are a standard benchmark for urban demand modelling [20], and spatio-temporal graph networks such as STGCN and ASTGCN deliver state-of-the-art accuracy on them [21, 22]. For control at the edge, lightweight models are attractive: gradient-boosted decision trees [23, 24] trained with the pinball loss of quantile regression [25] provide calibrated upper quantiles at negligible inference cost, which is what a risk-aware capacity planner needs.

3. System Model and Problem Formulation

3.1. Edge–Cloud Architecture

Our model builds upon the edge–cloud synergy model (ECSM) of Zhang et al. [26], which organizes smart-city processing into device, edge and cloud tiers coordinated by a feedback control loop and studies a reference deployment of ten edge nodes and three regional cloud servers. We adopt this deployment and its slice taxonomy directly. Slice S1 carries healthcare alarms and emergency-vehicle telemetry and is never preempted; S2 carries real-time traffic analytics and incident detection; S3 carries best-effort bulk uploads and historical synchronization. Each zone is served by edge site, which hosts processing units (PUs), i.e., accelerator-backed inference workers. A PU is asleep, waking, idle or busy. Edge sites are interconnected by metro links, and each of them reaches the regional clouds through a shared backhaul followed by a wide-area path (Fig. 1).

3.2. Task and Slice Model

Task belongs to slice, originates in zone at time, carries input bits and output bits, and requires seconds of processing on a reference PU. The work is log-normally distributed with slice mean and coefficient of variation, and every slice has an end-to-end deadline. Tasks of slice arrive in zone as a non-homogeneous Poisson process with rate. Inside a site, S1 is served with preemptive-resume priority, S2 has non-preemptive priority over S3, and tasks of the same slice are served in arrival order.

3.3. Latency Model

A task placed at site experiences the end-to-end latency

Where is the access delay with uplink rate; is zero for local execution, equals the metro delay plus for a neighbouring site, and comprises backhaul queueing, and the wide-area delay for a cloud; is the waiting time at the site; is the speed of its PUs; and returns the result to the device.

3.4. Energy Model

The power drawn by edge site is

Where counts powered PUs (idle, busy or waking), counts busy PUs, and, and are the idle, busy and sleep powers. Waking a PU takes seconds, during which it draws idle power without serving tasks. Cloud regions are elastic multi-tenant pools, so only the dynamic energy of a task is attributed to the city, scaled by the power usage effectiveness:

Where is the per-bit energy of the backhaul and wide-area path; forwarding to a neighbour costs per bit [27]. The infrastructure energy integrates (2) over all edge sites and adds cloud and transport energy. Device uplink energy is identical for all policies, because every task traverses the same access link, and is therefore excluded.

3.5. Problem Formulation

The controller chooses a destination for every task when it reaches its local site, and the number of powered PUs of every site, knowing that newly woken PUs become available only after. With completed tasks, the goal is to

Where normalizes energy by the cloud energy of an

average task and and weight the two objectives. Placement and capacity decisions are coupled through the queues, the arrival rates are non-stationary and unknown, and the two decisions act on different time scales; we therefore decompose the problem into the two loops described next.

4. The PLEO Framework

4.1. Overview

Fig. 1 summarizes PLEO. The reactive controller of ECSM [26] runs a fast loop that rebalances slice shares and task placement and a slow loop that updates scaling decisions, protected by hysteresis and a minimum warm capacity. PLEO retains this cadence and both guardrails but derives every decision from predictions: the slow loop (every = 60 s) predicts demand and latency risk to size the powered capacity of each site, and the fast loop (per task, with telemetry refreshed every = 1 s) predicts the completion time and marginal energy of each task at every candidate site.

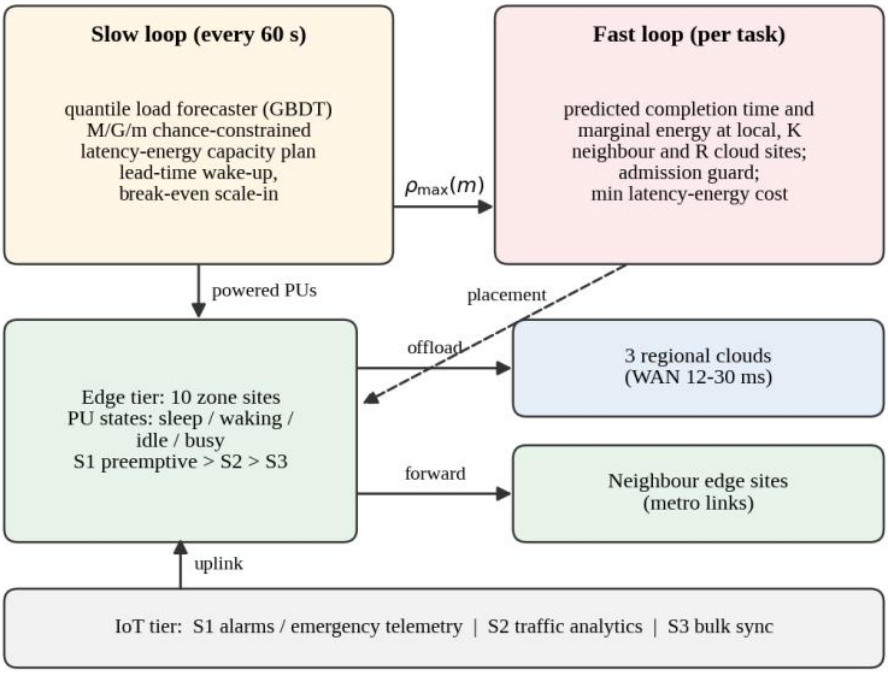


Figure 1. Architecture of PLEO. The tiers and slices follow the reference edge–cloud synergy model; the slow loop sizes the powered capacity of every edge site from quantile forecasts and an M/G/m chance constraint, and the fast loop places each task using its predicted completion time and marginal energy.

4.2. Quantile Load Forecasting

At the end of every 5-min slot, a forecaster predicts the vehicle count of each zone for = 1 and 2 slots ahead. The features comprise the last twelve observations, the values of the target slot one day and one week earlier, the time of day (sine and cosine), the day of the week, a weekend flag, the zone identifier and short-window statistics; targets are normalized by the zone's training mean so that a single global model serves all zones. We train histogram-based gradient-boosted trees [28] with a squared loss for the conditional mean and with the pinball loss

for the quantiles [25]. Slot forecasts are interpolated onto the 60-s control grid and converted into slice rates, e.g., for S2, where is the number of analytics tasks per vehicle.

4.3. Chance-Constrained Capacity Planning

For a site with powered PUs and offered load (in Erlang), the Erlang-C formula gives the probability that an arriving task has to wait in an M/M/m queue [18]:

Service demands in our system are much less variable than exponential ones, so we apply the two-moment correction for M/G/m queues [19], which scales waiting times by and yields the tail approximation

With mean service demand. Let be the waiting budget of S2, i.e., its deadline minus the mean uplink and return delays and the 95th percentile of its service demand. The largest utilization that keeps the probability of exceeding this budget below is

Which grows with the site size: equals 0.16, 0.71 and 0.89 for = 1, 5 and 15. For each candidate, the planner fills the capacity with the forecast demand of S1, S2 and S3 in this order and sends the overflow to the nearest cloud. With served load, total task rate and the predicted latency rate, which uses the Erlang-C mean waiting time for local tasks, the planning cost is

And. Two horizons turn this static choice into a provisioning policy. Capacity is added whenever the largest over the lead horizon exceeds the committed capacity, so that newly woken PUs are ready when the demand arrives. Capacity is released only if it is not needed again before the break-even time

Beyond which keeping a PU idle costs more energy than waking it again, in the spirit of switching-cost-aware right-sizing [15]; the scale-in target is therefore the largest over the horizon. As in ECSM, scale-in additionally requires two consecutive epochs and never goes below a minimum warm capacity = 1. PLEO plans with the 0.9-quantile forecast.

4.4. Predictive Task Placement

When task of slice reaches its local site, PLEO evaluates the local site, its nearest neighbours and the clouds. For a candidate with active PUs, queued work of equal or higher priority and residual work of the tasks in service, the predicted latency and marginal energy are

Where indicates forwarding to a neighbour; for clouds, follows (3) with replaced by. The marginal energy of an edge

site deliberately excludes idle power, which the slow loop has already committed. The task is sent to

Where contains the candidates whose predicted latency meets after the time already spent in the uplink, with safety margin. A neighbour is admitted for S2 only if its spare capacity

Exceeds = 0.5 Erlang, where is the mean forecast of the neighbour's own demand and is the work already forwarded to it. This guard prevents several overloaded sites from flooding the same neighbour at once. If is empty, the task goes to the candidate with the smallest predicted latency.

4.5. Complexity

The slow loop evaluates at most capacities for each of a few horizon steps per site, i.e., operations per minute, and the fast loop evaluates candidates per task. Both costs are negligible: our single-core implementation simulates and controls about 1.4 million tasks per second.

5. Experimental Setup

5.1. Dataset and Workload

We use PeMSD8, the public release of Caltrans PeMS District 8 (San Bernardino) data prepared by Guo et al. [20], [22]: flow, occupancy and speed of 170 loop detectors at 5-min resolution from 1 July to 31 August 2016. The files were obtained from the authors' public repository, and their SHA-256 checksums are stored with our code. We group the detectors into ten zones by spectral clustering of the road graph, which yields zones of 5 to 33 detectors and a correspondingly uneven demand. The real data drive the dominant slice and its bursts: every vehicle passing a detector generates one S2 analytics task (= 1), and a detector whose speed drops below 60% of its typical speed for that slot of day and day type adds S1 alerts at 0.5 tasks/s. A synthetic base load of wearable alarms (0.02 tasks/s per detector) completes S1, and S3 is a synthetic constant stream of 0.02 tasks/s per detector. The first 42 days serve for training, the next 7 for validation and the last 13 (19–31 August) for testing. Each edge site is dimensioned for the 95th percentile of its training-period load at 80% utilization, which gives between 3 and 15 PUs per site and 75 PUs in total. Fig. 2 shows the resulting S2 load and the forecasts for the busiest zone.

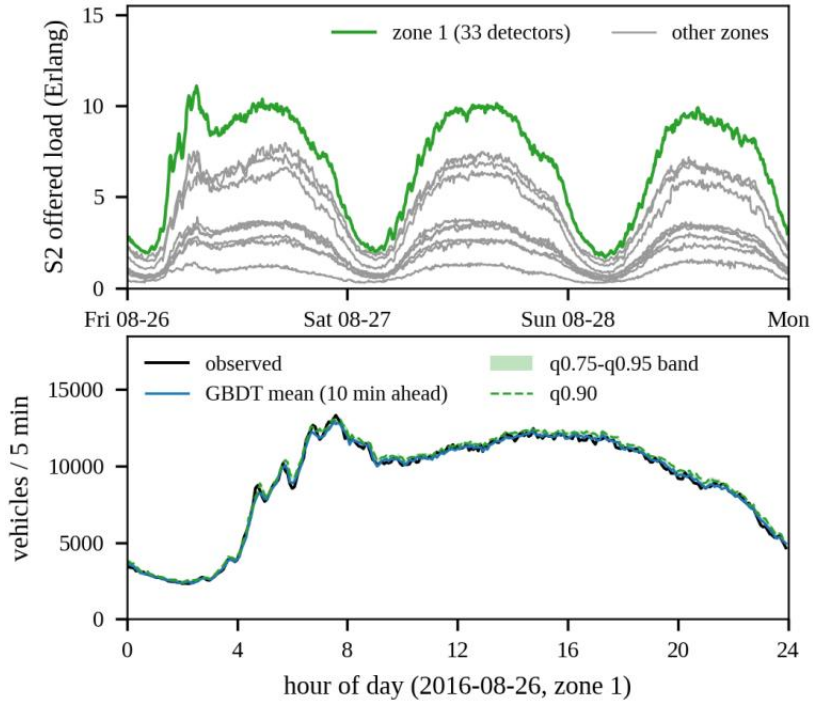


Figure 2. Real workload drivers derived from PeMSD8. Top: S2 offered load of the ten zones over three test days. Bottom: observed traffic of zone 1 on 26 August 2016 with the 10-min-ahead GBDT mean forecast and quantile band.

5.2. Simulator and Parameters

We implemented an event-driven simulator in Python with Numba that models every task, PU power state, queue, metro link, backhaul and wide-area path; Table I lists its parameters. The power figures are representative of accelerator-backed edge workers that are not energy-proportional, clouds are elastic pools with a PUE of 1.2, and the per-bit transport energies are conservative values relative to measured network electricity intensities [27]. With these values, an edge PU becomes cheaper than the cloud only above the break-even utilization = 0.57, which makes the energy trade-off non-trivial.

Table I. Simulation Parameters

Parameter	Value
Edge sites / PUs	10 sites; = 15, 12, 11, 9, 5, 6, 5, 5, 4, 3 (75 PUs)
Cloud sites	3 regions × 64 PUs; one-way WAN delay 12/20/30 ms plus exponential jitter (mean 5%)
Network	Access 2 ms; metro 0.5 ms + 10 µs/km at 10 Gb/s; backhaul 1 Gb/s; downlink 100 Mb/s
Uplink rate	Uniform in [5, 20] Mb/s (S1) and [40, 120] Mb/s (S2, S3)
S1 critical	= 10 ms, = 0.3, 1 kB, = 40 ms
S2 analytics	= 250 ms, = 0.3, 150 kB, = 600 ms
S3 best effort	= 2 s, = 0.5, 2 MB, = 20 s
Edge PU power	= 25 W, = 65 W, = 3 W, = 60 s
Cloud energy	= 70 W per busy PU, = 1.2, = 1
Transport energy	= 2 nJ/bit, = 10 nJ/bit
Control	= 60 s, = 1 s, = 3, = 1, hysteresis 2 epochs
PLEO	= 0.9, = 0.1, = 0.5, = 0.1
Reactive scaler	Scale out above utilization 0.75 towards 0.6; scale in below 0.35

5.3. Baselines

We compare five policies. Cloud-only and Edge-only are the two reference architectures against which ECSM was evaluated [26]; Edge-only keeps all PUs powered and processes every task at its local site. ECSM-R implements the QoS-based orchestration policy of ECSM: S1 stays at the local site; S2 is processed locally if the site is healthy, otherwise at the nearest healthy neighbour, otherwise in the cloud; S3 batch work is sent to the cloud. A site is healthy if fewer S1 and S2 tasks are queued than it has active PUs and its smoothed utilization is below 0.9. The slow loop scales out when the measured utilization exceeds 0.75, aiming at 0.6, and scales in below 0.35 after two consecutive epochs, with one PU of minimum warm capacity. Because ECSM specifies these rules in policy terms, ECSM-R instantiates them with common autoscaling thresholds, which Section VI-D sweeps so that the comparison does not hinge on a single setting. MJ (myopic joint) applies PLEO's placement rule (12)–(14) with measured instead of forecast rates together with the reactive scaler of ECSM-R, which isolates the value of PLEO's capacity planning. The PLEO hyper-parameters were selected on one validation day (15 August 2016).

5.4. Metrics

For each slice we report the mean, 95th (P95) and 99th (P99) percentile end-to-end latency and the deadline-satisfaction ratio (DSR), i.e., the fraction of tasks completed within, which corresponds to the slice reliability metric of the reference model. Energy is the infrastructure energy per completed task. The main experiments cover seven test days (25–31 August 2016, 84.0 million tasks per run) with three random seeds after a one-hour warm-up. Across seeds, the standard deviation is at most 0.03 J per task and 0.1 ms for the S2 mean latency, so we report means. The sweeps use 26–27 August and one seed.

6. Results

6.1. Forecast Accuracy

Table II compares the forecasters on the 13 test days. Gradient-boosted trees lower the weighted absolute percentage error (WAPE) from 2.47% (persistence) to 1.83% at a 5-min horizon and from 3.48% to 2.05% at 10 min, and they also beat ridge regression on the same features. Seasonal baselines are much weaker because they ignore the day-to-day level of traffic. The quantile models are well calibrated: the empirical coverage of the 0.75, 0.9 and 0.95 quantiles is 0.76, 0.90 and 0.94 at = 1, and the 0.9-quantile model roughly halves the pinball loss of the mean model (16.5 vs. 34.7).

Table II. Zone-Level Forecast Accuracy on the Test Period

Model	MAE_5	$RMSE_5$	$WAPE_5$ (%)	MAE_{10}	$RMSE_{10}$	$WAPE_{10}$ (%)
Persistence	97.2	137.7	2.47	136.6	201.1	3.48
Seasonal (weekly)	153.1	241.0	3.90	153.1	241.0	3.90
Historical avg.	243.7	350.1	6.20	243.7	350.1	6.20
Ridge regression	85.5	123.1	2.18	101.1	147.1	2.57
GBDT (mean)	71.9	102.1	1.83	80.6	116.2	2.05

6.2. Overall Comparison

Table III summarizes the seven-day results. Every policy that uses the edge meets all S1 deadlines with a mean S1 latency of 14.8 ms, whereas Cloud-only meets only 54.3% of them because the wide-area round trip consumes most of the 40-ms budget. Edge-only keeps all 75 PUs powered at an average utilization of 0.53, below the break-even point, so it is the most energy-hungry policy (24.92 J per task) and consumes even more than Cloud-only. It also has the worst S2 tail (P99 of 622 ms), because 2-s S3 jobs occupy PUs non-preemptively at busy sites. ECSM-R avoids this blocking by sending all S3 work to the cloud, which keeps the S2 tail low but adds 345 MJ of cloud energy per week; its 58.3 powered PUs run at a utilization of only 0.56, close to break-even, and its energy per task (24.44 J) is slightly above that of Cloud-only.

Table III. Seven-Day Results (Mean of Three Seeds)

Method	S1 DSR (%)	S2 mean (ms)	S2 P95 (ms)	S2 P99 (ms)	All DSR (%)	Energy (J/task)	Powered PUs
Cloud-only	54.3	297.4	436	521	98.39	23.99	0.0
Edge-only	100.0	288.9	460	622	98.86	24.92	75.0
ECSM-R	100.0	280.0	430	523	99.78	24.44	58.3
MJ	100.0	273.9	416	502	99.85	23.00	62.2
PLEO	100.0	278.4	424	512	99.82	22.18	55.6

PLEO consumes the least energy, 22.18 J per task, i.e., 9.2% less than ECSM-R, 11.0% less than Edge-only, 7.6% less than Cloud-only and 3.6% less than MJ. For the ten sites this amounts to 190 MJ (53 kWh) per week compared with ECSM-R. The saving comes from running fewer and busier PUs: PLEO powers 55.6 PUs on average at a utilization of 0.69, compared with 62.2 PUs for MJ, which cuts edge idle energy by 9.2%, and it executes 74% of the S3 work on edge capacity that is powered anyway. At the same time, PLEO lowers the S2 mean, P95 and P99 latencies of ECSM-R by 0.6%, 1.5% and 2.1% and completes 99.82% of all tasks on time. MJ, which keeps 6.6 more PUs powered on average, attains the lowest S2 latency (273.9 ms, 1.6% below PLEO) at 3.7% higher energy, a different operating point of the same trade-off that Section VI-D examines.

6.3. Temporal Behaviour

Fig. 3 illustrates one weekday. All adaptive policies follow the diurnal demand, and PLEO does so with the smallest powered capacity: over the whole week it keeps 4.7% fewer PUs powered than ECSM-R and 10.7% fewer than MJ. The S2 latency of the adaptive policies differs by only a few milliseconds over the day, while Cloud-only is consistently slower because of the wide-area round trip. The price of PLEO's tighter provisioning is a higher number of capacity changes, 18.5 per site and day compared with 4.1 for MJ; the energy of every wake-up is included in the results.

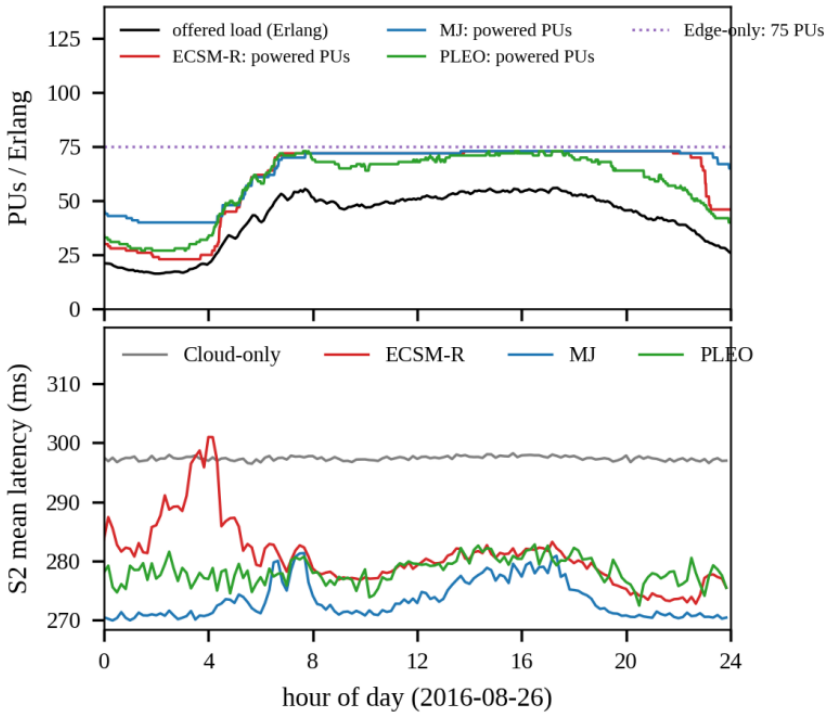


Figure 3. Behaviour on 26 August 2016 (main experiment, seed 1). Top: total offered load and powered edge PUs. Bottom: S2 mean latency in 10-min bins.

6.4. Latency–Energy Trade-Off

To compare the policies at equal latency, Fig. 4 sweeps the target utilization of the reactive policies from 0.5 to 0.8 and the risk budget of PLEO from 0.02 to 0.4. Raising moves MJ along a steep front from 23.35 J (S2 P95 of 414 ms) to 21.53 J (469 ms). ECSM-R degrades abruptly at = 0.8: its health rule pushes S2 work away from sites whose capacity it has already shrunk, and its DSR falls to 98.05%. PLEO's sweep is nearly flat in latency: its energy ranges from 22.36 to 21.82 J while the S2 P95 latency stays between 418 and 426 ms, because higher utilization limits let powered edge capacity absorb more of the forwarded work. Every ECSM-R configuration and MJ at = 0.7 are dominated by a PLEO configuration. At = 0.4, PLEO needs 3.6% less energy than MJ at its default = 0.6 with an S2 P95 latency only 2 ms higher; MJ reaches lower energy only at = 0.8, at the cost of a 12.2% higher P95 latency and a DSR 0.32 percentage points lower. The advantage stems from the per-site utilization limit (9): a single city-wide target is too conservative for large sites and too aggressive for small ones, whereas the chance constraint lets large sites run hot and keeps small sites safe. The default = 0.1, chosen on validation data, lies in the middle of PLEO's front.

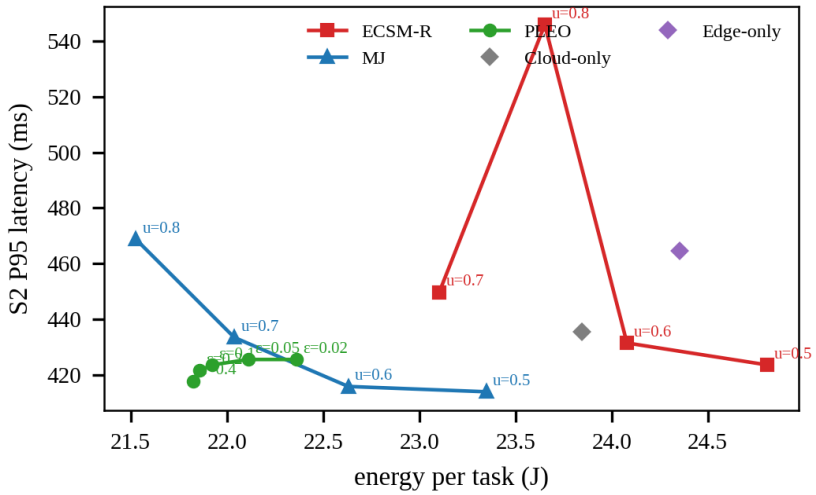


Figure 4. Latency–energy trade-off on 26–27 August 2016. Reactive policies are swept over the target utilization and PLEO over the risk budget; diamonds mark the reference architectures.

6.5. Load Stress

Fig. 5 scales all arrival rates by 1.0, 1.2 and 1.4. Edge-only cannot absorb the peaks: 6.1% of its tasks miss their deadline at 1.2× and 29.4% at 1.4×, where S2 queues build up to a mean latency of 6.3 s. Cloud-only keeps its energy constant, but its S1 DSR falls to 48.4% as the shared backhaul becomes busier. Among the policies that keep the DSR above 99%, PLEO consumes the least energy at every load level (21.92, 21.28 and 20.86 J per task). At 1.4× it matches the DSR of MJ (99.73% vs. 99.73%) with 2.0% less energy, and compared with ECSM-R it reduces energy by 4.7% and the S2 mean and P99 latencies by 6.1% and 5.8%.

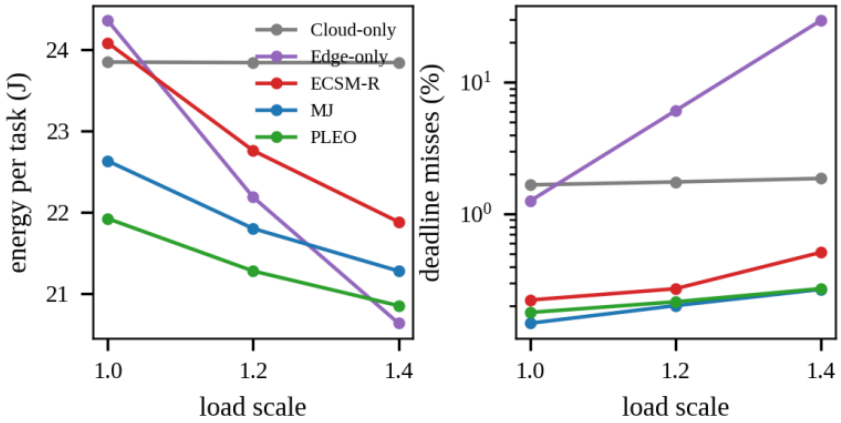


Figure 5. Energy per task (left) and share of tasks missing their deadline (right, logarithmic scale) when the arrival rates of all slices are scaled.

6.6. Ablation and the Role of Forecasting

Table IV isolates PLEO's components on the seven test days (one seed per variant). Replacing the M/G/m correction by the plain M/M/m bound makes the planner keep 58.2 instead of 55.6 PUs powered and raises energy by 1.4%, which confirms that an accurate queueing model is what allows sites to run closer to their limits. An admission reservation that keeps S3 work off the capacity needed by S1 and S2 yields the lowest S2 latency (P99 of 498 ms) but sends most S3 work to the cloud and costs 6.2% more energy, which makes it a useful latency-first variant. The break-even rule leaves energy unchanged but reduces capacity changes by 11.9%. TRIAD demonstrates the importance of measuring hardware execution latency and memory-use trade-offs under explicit real-time operating constraints [30].

Table IV. Ablation of PLEO Components (25–31 August 2016)

Variant	Energy (J/task)	S2 mean (ms)	S2 P99 (ms)	All DSR (%)	Powered PUs	Scalings per site-day
PLEO (full)	22.18	278.4	512	99.82	55.6	18.5
Persistence forecast	22.13	280.3	516	99.81	54.6	23.7
Mean forecast	22.13	280.1	516	99.81	54.6	19.2
Oracle forecast	22.13	280.0	516	99.81	54.6	19.6
M/M/m bound	22.49	275.7	507	99.84	58.2	18.7
No break-even rule	22.18	278.5	512	99.82	55.6	21.0
+ S3 reservation	23.55	272.3	498	99.87	55.6	18.5

The forecasting results are the most instructive. Replacing the 0.9-quantile GBDT forecast by its mean, by a persistence forecast or even by the true future rates (oracle) changes energy by at most 0.2% and the S2 mean latency by at most

1.9 ms; the quantile margin simply trades a little energy for slightly lower latency. Likewise, varying the wake-up time from 10 s to 300 s changes the energy of MJ and PLEO by less than 0.3% (Fig. 6). Zone-level demand in PeMSD8 changes by roughly 1% per minute even during the morning ramp, so the slack built into the chance-constrained plan absorbs any provisioning lag, and forecast accuracy is not a bottleneck. PLEO's gains therefore come from how it sizes capacity and places tasks; forecasting becomes valuable only for faster demand changes, such as event-driven surges, which the aggregated PeMSD8 traffic does not contain.

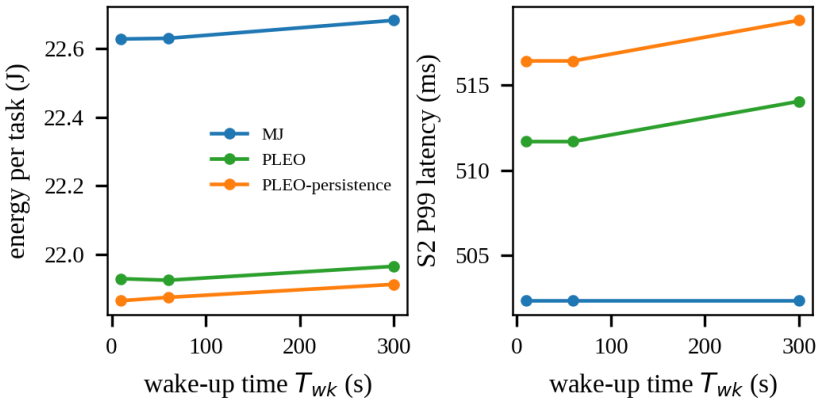


Figure 6. Sensitivity to the PU wake-up time on 26–27 August 2016.

6.7. Discussion and Limitations

Three observations generalize beyond this case study. First, edge capacity is energy-efficient only above a break-even utilization (0.57 here); policies that keep edge PUs powered below it, such as Edge-only and, on average, ECSM-R, lose to a pure cloud deployment on energy while winning on latency. Second, the latency budget of a slice translates into a utilization limit that depends on the site size, so capacity should be planned per site rather than with a city-wide threshold. Third, short-term forecasting of aggregated urban demand adds little to a well-designed queueing-aware controller. The study also has limitations: it is simulation-based, the S1 base load and S3 are synthetic, the power figures are representative rather than measured, PLEO changes capacity more often than the reactive baselines, and learning-based controllers such as deep reinforcement learning are left for future comparison.

7. Conclusion

This paper presented PLEO, a queueing-predictive framework that jointly optimizes latency and energy in smart-city edge–cloud networks by coupling chance-constrained capacity planning with predictive per-task placement. Built on a reference edge–cloud synergy architecture and evaluated with real PeMSD8 traffic data, PLEO reduced infrastructure energy per task by 9.2% compared with reactive slice-aware orchestration and by 7.6% and 11.0% compared with cloud-only and edge-only processing, met all critical-slice deadlines, and completed 99.82% of all tasks on time under nominal load and 99.73% under 40% higher load. The analysis further showed that queueing-aware capacity planning, rather than forecast accuracy, governs the latency–energy trade-off for city-scale demand. Future work will evaluate PLEO on event-driven workloads, compare it with learning-based controllers and validate it on an edge testbed.

References

[1] Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. IEEE Internet of Things Journal, 1(1), 22–32. https://doi.org/10.1109/JIOT.2014.2306328

[2] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[3] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63–72.

[4] Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials, 19(3), 1628–1656. https://doi.org/10.1109/COMST.2017.2682318

[5] Khan, L. U., Yaqoob, I., Tran, N. H., Kazmi, S. M. A., Dang, T. N., & Hong, C. S. (2020). Edge-computing-enabled smart cities: A comprehensive survey. IEEE Internet of Things Journal, 7(10), 10200–10232. https://doi.org/10.1109/JIOT.2020.2987070

[6] Trigka, M., & Dritsas, E. (2025). Edge and cloud computing in smart cities. Future Internet, 17(3), 118. https://doi.org/10.3390/fi17030118

[7] Foukas, X., Patounas, G., Elmokashfi, A., & Marina, M. K. (2017). Network slicing in 5G: Survey and challenges. IEEE Communications Magazine, 55(5), 94–100. https://doi.org/10.1109/MCOM.2017.1600951

[8] Barroso, L. A., & Hölzle, U. (2007). The case for energy-proportional computing. Computer, 40(12), 33–37. https://doi.org/10.1109/MC.2007.443

[9] Fan, X., Weber, W.-D., & Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. In Proceedings of the 34th Annual International Symposium on Computer Architecture (pp. 13–23). https://doi.org/10.1145/1250662.1250665

[10] Chen, X., Jiao, L., Li, W., & Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking, 24(5), 2795–2808. https://doi.org/10.1109/TNET.2015.2487344

[11] Mao, Y., Zhang, J., & Letaief, K. B. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications, 34(12), 3590–3605. https://doi.org/10.1109/JSAC.2016.2611964

[12] Tran, T. X., & Pompili, D. (2019). Joint task offloading and resource allocation for multi-server mobile-edge computing

networks. IEEE Transactions on Vehicular Technology, 68(1), 856–868. https://doi.org/10.1109/TVT.2018.2881191

[13] Wang, J., Ge, Y., Liu, J., & Jiang, B. (2024). Early warning of liquidity stress in U.S. banks with seasonally aligned frailty hazards and delayed-feedback conformal alarms. Journal of Trends in Finance and Economics, 1(1), 85–100.

[14] Huang, L., Bi, S., & Zhang, Y.-J. A. (2020). Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Transactions on Mobile Computing, 19(11), 2581–2593. https://doi.org/10.1109/TMC.2019.2928811

[15] Lin, M., Wierman, A., Andrew, L. L. H., & Thereska, E. (2013). Dynamic right-sizing for power-proportional data centers. IEEE/ACM Transactions on Networking, 21(5), 1378–1391. https://doi.org/10.1109/TNET.2012.2226216

[16] Gandhi, A., Harchol-Balter, M., Raghunathan, R., & Kozuch, M. A. (2012). AutoScale: Dynamic, robust capacity management for multi-tier data centers. ACM Transactions on Computer Systems, 30(4), 14. https://doi.org/10.1145/2382553.2382556

[17] Xu, J., Chen, L., & Ren, S. (2017). Online learning for offloading and autoscaling in energy harvesting mobile edge computing. IEEE Transactions on Cognitive Communications and Networking, 3(3), 361–373.

[18] Harchol-Balter, M. (2013). Performance modeling and design of computer systems: Queueing theory in action. Cambridge University Press. https://doi.org/10.1017/CBO9781139226424

[19] Whitt, W. (1993). Approximations for the GI/G/m queue. Production and Operations Management, 2(2), 114–161.

[20] Chen, C., Petty, K., Skabardonis, A., Varaiya, P., & Jia, Z. (2001). Freeway performance measurement system: Mining loop detector data. Transportation Research Record, 1748, 96–102. https://doi.org/10.3141/1748-12

[21] Teng, D. (2025). TRIAD: Deadline- and reliability-aware scheduling for post-earthquake structural assessment workflows. World Journal of Engineering Research, 3(6), 55–66.

[22] Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 1, pp. 922–929). https://doi.org/10.1609/aaai.v33i01.3301922

[23] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

[24] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems (Vol. 30, pp. 3146–3154).

[25] Koenker, R., & Bassett, G. (1978). Regression quantiles. Econometrica, 46(1), 33–50. https://doi.org/10.2307/1913643

[26] Zhang, S., Qiu, L., & Zhang, H. (2025). Edge cloud synergy models for ultra-low latency data processing in smart city IoT networks. International Journal of Science, 12(10), 1–16.

[27] Aslan, J., Mayers, K., Koomey, J. G., & France, C. (2018). Electricity intensity of internet data transmission: Untangling the estimates. Journal of Industrial Ecology, 22(4), 785–798. https://doi.org/10.1111/jiec.12630

[28] Chen, J., Zhang, X., & Han, X. (2023). Explainable identification of tax-related financial reporting anomalies in U.S. public companies: Cross-fitted tax-consistency residuals and Owen-value Shapley propagation. World Journal of Economics and Business Research, 1(1), 33–45.

[29] Wang, Y., Zou, X., Lu, Z., Wei, K., & Rhee, M. (2026). PRESAGE: Uncertainty-aware predictive elastic scheduling for cloud analytics and GenAI inference. IEEE Access. https://doi.org/10.1109/ACCESS.2026.3734498

[30] Zhang, F., Shen, A., & Huang, S. (2026). TRIAD: Co-designing mixed-precision quantization, operator fusion, and on-chip memory reuse for real-time vision–language–action inference on automotive NPUs. Results in Engineering. https://doi.org/10.1016/j.rineng.2026.113673