From Measurement to Materials: Cost-Aware Right-Sizing of Multi-Tier Edge Computing Infrastructures

Massamaesso Narouwa[1,3], Leo Mendiboure[1] *(IEEE Member)*, Sassi Maaloul[1,2], Hakim Badis[3], Dereje M. Molla[4], Marion Berbineau[4], Rami Langar[3,5]
[1]*COSYS-ERENA, Univ. Gustave Eiffel, 33400 Talence, France*
[2]*CNRS, Institut FEMTO-ST, University of technology (UTBM), 90010 Belfort cedex, France*
[3]*CNRS, LIGM, Univ. Gustave Eiffel, F-77454 Marne-la-Vallée, France*
[4]*COSYS-LEOST, Univ. Gustave Eiffel, F-59666 Villeneuve d'Ascq Cedex, France*
[5]*Software and IT Engineering Department, École de Technologie Supérieure (ÉTS), Montréal, Canada*

***Abstract*—Multi-access Edge Computing (MEC) promises ultra-low end-to-end latency for 5G/6G services, but each additional edge site drives CAPEX and OPEX, making accurate dimensioning essential. Existing studies either optimize runtime placement on a pre-deployed edge tier or size a single layer under synthetic traffic, leaving operators without a principled way to decide where to build servers or how many. We propose a four-phase, measurement-driven workflow that (i) records live radio and transport delays, (ii) converts real applications into machine-readable profiles, (iii) derives tier-specific latency budgets, and (iv) greedily assigns every service to the cheapest network layer that still meets its Service Level Agreement. Applied to FerroMobile, an autonomous railway system with URLLC and eMBB workloads, evaluations demonstrate the relevance of such a framework and the ability of a multi-layer infrastructure to reduce costs while guaranteeing performance.**

***Index Terms*—5G/6G, Multi-Access Edge Computing (MEC), Multi-Tier Placement, Right-Sizing, Cost Optimization.**

I. Introduction

Emerging 5G-class applications, ranging from immersive EXtended Reality and collaborative driving to industrial digital twins, require end-to-end latencies below a few tens of milliseconds and sustained multi-megabit throughput. Once radio, fronthaul, and core delays are aggregated, central clouds seldom reach those figures. Multi-access Edge Computing (MEC) closes the gap by relocating computation from distant hyperscale facilities to infrastructure only one or two network hops away. Thanks to this proximity, MEC becomes the indispensable substrate for forthcoming 5G/6G service classes that cannot tolerate even occasional latency spikes [1].

However, proximity is not free: every edge site consumes capital and operational expenditure, including racks, power, and field support. A simple one-server-per-cell strategy [2] quickly becomes uneconomical in rural or bandwidth-constrained regions. Operators therefore need to right-size their rollouts and purchase only the resources required by applications that truly benefit from edge execution. Otherwise, they will under-use their assets and never achieve profitability.

The prior work is divided into two categories. Dynamic orchestration papers assume the edge tier already exists and optimize live migration, queue balancing, or routing [3]–[7]. Static dimensioning sizes a single edge layer for worst-case demand by robust optimisation or queueing theory [2], [8], [9]. Yet three critical gaps remain: (i) no systematic method decides whether an edge node is needed at a given site; (ii) real applications are replaced by uniform synthetic loads, masking heterogeneous Service Level Agreements (SLAs) and footprints; (iii) cost is treated as a soft weight, whereas in frugal deployments the decisive question is whether to build any server at a site.

We bridge these gaps with a measurement-driven workflow that starts before any hardware purchase. First, radio and backhaul Key Performance Indicators (KPIs) are measured on site; second, application profiles are collected via an automated portal. A tier-scan policy then maps every service to the highest network layer (device, gNB, aggregation, regional cloud, central cloud) that still satisfies its SLA under a configurable admission mode. Finally, a greedy algorithm derives the minimal multi-tier footprint that fits all services within a concrete CAPEX envelope. Evaluations confirm that measurement-driven right-sizing outperforms both cloud-centric and over-provisioned edge deployments.

The remainder of this paper is organized as follows. Section II surveys related work; Section III details the proposed framework; Section IV presents the sizing heuristic; and Section V reports the experimental results.

II. Related Work

Run-time optimisation assumes the edge already exists. Much of the MEC literature starts after the infrastructure is in place. Early papers formulate service–node assignment as an online problem and solve it with integer programming or reinforcement learning [3]–[7]. Although they show performance improvements, they assume unlimited resources: there is no indication of how much peripheral infrastructure is required before anything can be orchestrated.

Single-tier sizing under worst-case demand. Another branch sizes a single MEC layer, usually at the gNB. Khan *et*

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

al. [8] inventory CPU and storage at the RAN for multimedia, whereas Khakimov *et al.* [9] validate a flexible platform on seven base-station servers. Ouyang *et al.* [2] adapt those resources dynamically as users move. Such analyses are valuable first steps but do not tell whether parts of the workload could cost-effectively stay in aggregation nodes or the cloud, nor whether on-board compute may be preferable.

Multi-tier placement without frugal constraints. Some papers move beyond the gNB yet still sidestep cost realism. The authors of [10] couple service needs with RAN latency to choose between gNB and cloud, but they assume that every gNB can host a server. The authors of [11], [12] introduce placement frameworks for generic workloads; however they rely on synthetic latency matrices and disregard radio measurements. Cost enters these models only as a proportional term in the objective function, rather than as a hard budget that may rule out deploying any server at a given site.

Existing gaps. Across all categories three gaps persist: i) no measurement-based "should-we-deploy?" test, ii) no per-application footprint catalogue, iii) and cost viewed as a soft penalty. The present work closes these gaps through a measurement-driven, budget-aware right-sizing process.

III. Measurement-Driven Right-Sizing Framework

Edge planning often relies on rule-of-thumb over-provisioning. We replace guesswork with a traceable loop that transforms field measurements into a tier-aware Bill of Materials (BoM) through four steps (Fig. 1): **Instrument**, **Catalogue**, **Audit**, **Dimension**. Each step produces a version-controlled artefact, enabling continuous re-planning when traffic, radio conditions or the service catalogue evolve.

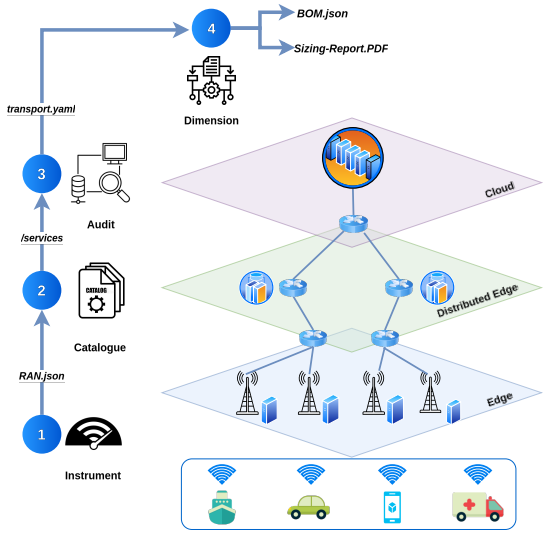


Fig. 1: Overview of the Proposed Framework

Phase 1 - Instrument: Characterising the Radio Edge

Purpose. Field campaigns repeatedly show that the radio hop contributes the largest, and the least predictable, share of end-to-end delay [13]. Sizing edge capacity without first bounding that variability can lead either to chronic under-provisioning (latency overshoots during peaks) or to costly over-provisioning (idle servers most of the day). Phase 1, therefore, establishes a quantitative envelope for every cell before any placement decision is attempted.

Operation. Five KPIs (one-way delay, RTT, jitter, BLER, and user throughput) directly affect application responsiveness and transport efficiency. Measurements run in three windows:

- Busy-hour window corresponds to the interval (e.g., 60 minutes) that carries the highest traffic load. This is the 99^{th}-percentile one-way delay that serves as a worst-case guard band for sizing.
- Steady-state window, providing both the average and the 95^{th}-percentile delays, capturing the typical user experience that must be satisfied most of the time.
- Maintenance window, aligned with the next scheduled software roll-out, that records the maximum delay necessary to gauge the operational headroom required to absorb configuration churn without breaching SLAs.

Data come either from live counters plus TWAMP/`iPerf3` micro-bursts, or from a calibrated digital twin (e.g. a setup based on SRSRAN/OAI + ns-3/Mininet as proposed in [13]).

For each cell c, traces are collapsed into $\{\texttt{min}, \texttt{mean}, P_{95}, P_{99}\}$ for delay, jitter and BLER, then stored in `RAN.json` as `{cell-ID : {min, mean, p95, p99}}`.

Keeping both tail (P_{99}) and typical (mean, P_{95}) values lets the placement engine switch between modes (cf. Phase 4) without new measurements, while cell-level granularity enables different anchor tiers in rural and urban areas.

Artefact. `RAN.json` — mapping cell-ID and $\{\texttt{min}, \texttt{mean}, P_{95}, P_{99}\}$.

Phase 2 - Catalogue: Service Profiles Construction

Purpose. Edge dimensioning is only as good as the input demand model. Phase 2 turns a heterogeneous set of applications into uniform, machine-readable descriptors that expose, up-front, the latency ceilings, resource footprints, scaling semantics, and location constraints that the placement engine will later consume. The resulting catalogue (YAML files) becomes the single source of truth for all subsequent automation and can be version-controlled just like source code.

Operation. Such an operation can be based on a self-service web form (operator portal) that guides each application owner through a four-step questionnaire; every field is validated on the fly and written to a draft YAML:

1) *Latency & reliability*: The owner chooses one of four round-trip ceilings ($<$10 ms, 10–50 ms, 50–100 ms, $>$100 ms) and flags whether occasional loss, re-ordering, or jitter spikes can be tolerated.
2) *Steady-state footprint*: CPU cores, RAM (MiB), persistent storage (GiB) and mean throughput (Mbit/s) are either entered manually or imported from the latest production telemetry snapshot.
3) *Burst factor & scaling policy*: The largest observed peak-to-mean ratio is binned into low ($\leq 2\times$), medium ($\leq 5\times$),

or high (>5×); the owner then selects a scaling model (horizontal, vertical, or fixed) so that the sizing engine can reserve head-room only where it is needed.

4) *Placement constraints*: Optional fields capture data-sovereignty or tenancy rules (e.g. "must stay in country A", "never co-locate with service B").

Each service receives two labels: latency bucket (UL <10 ms, LL 10–50 ms, ML 50–100 ms, or NC >100 ms) and resource band (light/medium/heavy; optional accelerator flag). The pair ⟨latency bucket, resource band⟩ becomes the primary key that the placement algorithm uses when grouping services.

Artefact. One YAML per service under `services/`.

Phase 3 – Audit: Quantifying the Core & Back-Haul

Purpose. Relocating compute to the edge only pays off when the wired segment between the gNB and the selected hosting tier adds less delay than the application can tolerate. Phase 3, therefore, benchmarks the live transport network and translates its behaviour into hard, tier-specific budgets that the sizing engine must obey.

Operation. The method that can be used to make such a system analysis is based on three steps:

1. *Topology crawl*: The operator's inventory or SDN controller exports a graph of gNBs with Local DataCenters (LDC), aggregation routers (ARs) with regional data centres (RDCs) and the central data centre (CDC). Each link is annotated with technology (fibre, microwave), nominal capacity, and protection scheme (ring, dual-homed, best-effort). The result is a machine-readable skeleton against which probes can later be mapped.
2. *Active probing*: Every directed hop is sampled with short RTT bursts (e.g. 5 pkts at 500 pps every 15 minutes for one week), covering at least one maintenance window.
3. *Envelope synthesis*: For each hosting tier ℓ ({gNB, AR, RDC, CDC}) we retain the slowest acceptable path (e.g. the 99$^{\text{th}}$-percentile RTT observed over the week). The outcome is a conservative wired budget $d(\ell)$ that remains constant throughout the sizing cycle.

Artefact. `transport.yml` — associates each tier with its conservative one-way delay.

Phase 4 – Dimension: Tier Assignment and Capacity Roll-Up

Purpose. The three previous phases have established: (i) cell–specific radio envelopes, (ii) per–service latency budgets and resource footprints, and (iii) tier–scoped wired delays. Phase 4 fuses these inputs into an actionable deployment plan that states where each service instance will run and how much CPU, RAM and storage must be purchased at every site.

Operation. Such an objective can be achieved in three steps:

1. *Choose an admission mode:* MAX (Strict SLA – uses the 99$^{\text{th}}$-percentile radio delay $D^{99}_{\text{RAN}}(c)$), MEAN (Balanced - uses the mean delay $D^{\text{mean}}_{\text{RAN}}(c)$.), MIN (Stress test - uses the minimum delay $D^{\text{min}}_{\text{RAN}}(c)$.), i.e. which radio percentile $D^{\text{mode}}_{\text{RAN}}(c)$ to enforce, trading cost for robustness.
2. *Run the latency-guided scan* (Alg. 1): for every pair (s, c) it finds the left-most tier whose wired delay still fits the SLA and, through SELECTSITE (Alg. 2), maps that instance to a concrete location (ie which server) that respects both budget and residual capacity.
3. *Apply the safety margin* α: the per–site counters $\mathbf{R}_{v,c}$ returned by Alg. 1 are inflated once to protect against unexpected traffic bursts and temporary server outages and rounded to the nearest server Stock Keeping Units (SKU) before being written to `BoM.json`.

Algorithm 1 Latency-guided greedy placement

Initialise $\mathbf{R}_{v,c} \leftarrow 0$ for every site v and cell c
for all cell c **do**
  sort $\mathcal{S}$ by ascending $D_{\max}(s)$
  for all service s **do**
    find left-most tier $\ell^\star$ s.t. $D^{\text{mode}}_{\text{RAN}}(c) + d(\ell^\star) \le D_{\max}(s)$
    $v \leftarrow$ SELECTSITE$(\ell^\star, s, c)$
    $\mathbf{R}_{v,c} \mathrel{+}= \mathbf{f}(s)$
  end for
end for
for all pair (v, c) **do**
  $\mathbf{R}_{v,c} \leftarrow \alpha\, \mathbf{R}_{v,c}$
end for
return $\mathbf{R}_{v,c}$ (raw Bill of Materials)

Artefacts. 1) `BoM.json` – a machine-readable table $\langle v, c\rangle \mapsto \mathbf{R}_{v,c}$ for procurement; 2) *Sizing-Report.pdf* – one line per $\langle s, c, \ell^\star, v, \text{mode}\rangle$ for transparent engineering.

The rule-set above is executed verbatim by the greedy procedure of Section IV. Changing the admission mode or the factor α triggers a full re-run in seconds.

IV. Greedy Multi-Tier Sizing Algorithm

Phase 4 defines the policy; this section shows how that policy is implemented.

A. Inputs carried over from the framework

The algorithm consumes the three JSON/YAML artefacts created in Phases 1–3 as well as the operator's choices::

- *Service library*: each YAML file gives the service's latency budget $D_{\max}(s)$ and its footprint $\mathbf{f}(s) = \langle \text{CPU}, \text{RAM}, \text{STO}\rangle$.
- *Radio table*: for every cell c it provides the radio envelope $D^{\text{mode}}_{\text{RAN}}(c)$ selected by the MAX/MEAN/MIN flag.
- *Transport table* it lists the deterministic one-way delay $d(\ell)$ between a gNB and each hosting tier $\ell \in \{\text{device}, \text{gNB}, \text{AR}, \text{RDC}, \text{CDC}\}$.
- *Head-room factor* α: a scalar safety margin introduced in Phase 4 to handle temporary failures (e.g. breakdown).

B. Latency-guided scan

The outer procedure (Alg. 1) performs a single pass over the set of radio cells. At the start of each pass all services are sorted by their latency ceiling $D_{\max}(s)$. With tighter SLAs examined first, a critical workload can never be displaced

to a higher tier by a laxer one, thus ensuring that the final mapping is latency–monotonic by construction. The algorithm then scans the tier list from left to right (device → CDC). The first tier whose wired delay, when added to the cell-specific radio envelope, still fits inside $D_{\max}(s)$ becomes the anchor layer $\ell^\star(s,c)$. Because $D_{\text{RAN}}^{\text{mode}}(c)$ is different for every cell, the same application may anchor at the gNB tier in a rural area yet remain at aggregation in a dense city, delivering the budget savings that motivated edge deployment in the first place. After the anchor layer is found, the helper SELECTSITE picks the concrete site and the corresponding resource counter $\mathbf{R}_{v,c}$ is updated. Finally, the head-room factor α inflates every counter, providing a tunable safety margin against forecasting errors while preserving linear complexity: the entire pass costs $O(|\mathcal{C}|\,|\mathcal{S}|\,|\mathcal{L}|)$ and needs no iteration.

Algorithm 2 Site selection inside the anchor tier

Require: tier ℓ, service s, cell c
Ensure: physical site v that instantiates ℓ
1: $\mathcal{V} \leftarrow$ *sites implementing tier* ℓ
2: remove $v \in \mathcal{V}$ if $\text{capex}_{\Delta}(v)$ exceeds remaining budget
3: remove $v \in \mathcal{V}$ if $\mathbf{f}(s) \not\preceq \text{freeRes}(v,c)$
4: **return** $v = \arg\min_{v\in\mathcal{V}} \text{PrefScore}(v)$

C. Helper SELECTSITE

Once the anchor layer is known, Alg. 2 answers the practical question: "Which concrete site should host this instance?". The routine mirrors real-life procurement practice in three lightweight filters. First, i) Budget feasibility: any site whose incremental CAPEX exceeds the remaining project envelope is discarded first. By applying the hard financial constraint upfront, we guarantee that every placement returned by the algorithm is immediately procurable. Then, ii) Resource fit: the surviving sites are screened against the free resource vector already reserved for other services in the same cell. This avoids silent over-commit and keeps the procedure stateless; no backtracking is ever required. Finally, iii) Policy-driven tie-break: if several candidates still qualify, the site with the lowest preference score is chosen. The score defaults to incremental CAPEX but can be repointed, via a single line of configuration, to energy, carbon, or any operator metric. Because the technical constraints have already been enforced, swapping the ranking never invalidates a feasible plan while giving planners full freedom to pursue secondary goals.

These three filters are sufficient because they satisfy in order the only mandatory conditions for a viable deployment: it must be affordable, it must fit, and, among equally good options, it should align with the operator's optimisation preference. Each call inspects at most $|\mathcal{V}_\ell|$ sites, so the global complexity remains dominated by the outer scan.

V. EVALUATION

A. Scenario and Workload

The experiments carried out here are conducted within the context of FerroMobile, an industrial project that aims to deploy autonomous rail-road vehicles on secondary lines in France to revitalize them [14]. The initial deployments specify numerous elements that enable us to reproduce a realistic execution environment: vehicles run on a 50-km single-track line with a design fleet of 60 cars. Vehicles cruise at 60–70 km/h (up to 90 km/h), and exchange control and infotainment traffic with a Traffic-Management-System (TMS). The six bidirectional TMS services retained for this study are listed in Table I; they cover both URLLC (e.g. *remote driving*, 10 ms) and eMBB (e.g. *passenger information*, 100 ms) workloads. CPU/RAM footprints are taken from past experiments conducted by researchers interested in similar applications or minimum specifications for given application classes (e.g. data server, video server), while performance objectives (throughput, latency, packet loss) are taken from the project specifications themselves. The YAML profiles produced in Phase 2 (cf. Section III) are therefore grounded in an operational vehicular stack rather than synthetic toy apps.

B. Experimental setup

All experiments are executed on a discrete–event emulation platform that couples the open–source srsRAN 5G stack to GNU radio and custom scripts that reproduce the FerroMobile use-cases. This platform was described and released in [13]. The track is modelled as a 50-km linear corridor served by ten gNBs (5km radius each); vehicles progress at 60–90 km/h following a Gauss–Markov mobility model. Each gNB (1 to 10 ms from user) feeds an AR; three ARs share one RDC (5 to 20 ms from user); and a CDC resides 20-50 ms away over the transport network, matching the "Base-station → CDC" range described in different documents proposed by ETSI [15]. FerroMobile specifications also allow the average (3) and maximum (5) number of vehicles that can be connected to the same base station at the same time to be defined based on the total number of vehicles, gNB coverage and mobility

TABLE I: Per-service footprints and QoS budgets used by the sizing engine

Application	**CPU** (peak cores)	**RAM** (peak MiB)	**Latency** target RTT (ms)	**Bandwidth** sustained (Mbit/s)	**Loss** target (%)	**Payload** bytes
Remote driving*	2.0	400	**10**	20 – 30	0.0001	–
Event warning*	0.5	200	**50**	0.20	0.0001	140
Infotainment (audio/video mix)	0.5 – 1.5	200 – 500	**50**	0.5 / 2.0	0.10	–
Traffic data collection	1.0	240	**100**	0.20	0.001	140
Passenger information system	1.0	240	**100**	1.0	0.001	800
Arrival / departure update	0.5	200	**100**	0.20	0.001	140

* Safety-critical (URLLC) service mapped to the UL bucket.

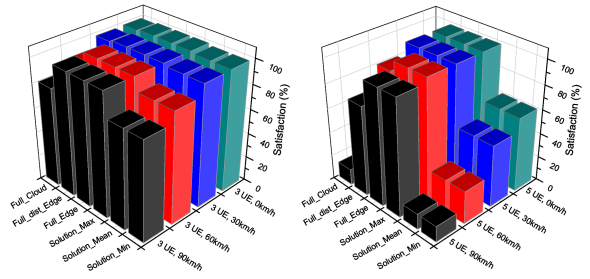
conditions. Wireless latency traces are captured thanks to our platform with *srsUE* and exported as the `RAN.json` artefact. RAM/CPU resources usage are computed based on the FerroMobile data.

C. Competing Placement Strategies

To highlight where the proposed workflow pays off we benchmark it against three intuitive baselines that embody the main deployment mind-sets seen in practice:

- Full-Edge: Every container is instantiated on the gNB-side server that serves the originating radio cell.
- Full-Dist-Edge: All services are consolidated on the first aggregation router; no compute is left at the gNB.
- Full-Cloud: The whole workload is back-hauled to the central data-centre (CDC).

The measurement-driven workflow was also exercised under its three admission policies: i) Workflow-MAX (Tight dimensioning - P_{99} radio delay); ii) Workflow-MEAN (Balanced option - average delay); iii) Workflow-MIN: (Opportunistic variant - best observed delay).



(a) 3 UEs (b) 5 UEs

Fig. 2: Speed vs. Satisfaction ($D_{\max}(s)$ = 100 ms).

D. Metrics and methodology

Two primary performance indicators are used:

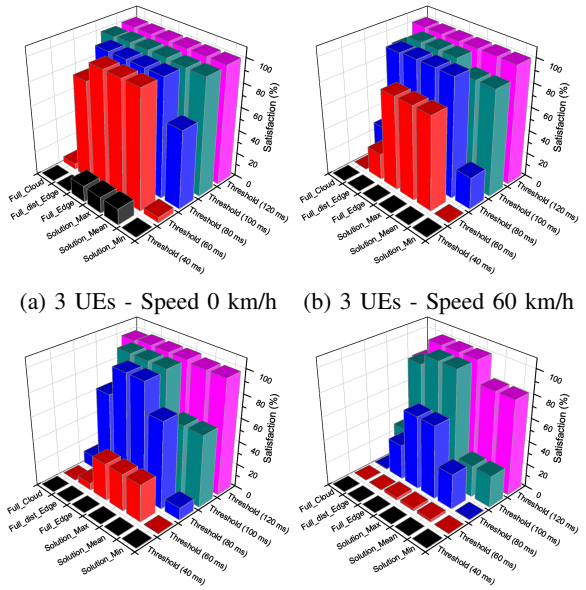
1) Application-level satisfaction: $\mathcal{S} = \frac{|\{r|D(r)\leq D_{\max}(s(r))\}|}{|\{r\}|}$ where each request r is satisfied when its measured round-trip time $D(r)$ does not exceed the budget $D_{\max}(s)$ declared for its service s (Table I). This ratio reflects the effective end-to-end QoS delivered to end users.
2) Provisioned resources: After placement, the BoM yields the total CPU and RAM that the operator must procure: $\mathrm{CPU}_{\Sigma} = \sum_{v,c} R^{\mathrm{CPU}}_{v,c}$ and $\mathrm{RAM}_{\Sigma} = \sum_{v,c} R^{\mathrm{RAM}}_{v,c}$. Smaller values translate directly into lower CAPEX/OPEX.

To determine the evolution of these metrics in the FerroMobile context, we considered different variable parameters:

- User load: average (3) vs. peak (5) vehicles per gNB.
- Mobility: static (0km/h), cruise (60km/h) and worst-case (90 km/h).
- Application mix: the six services described Table I.

E. Results and discussion

Satisfaction under mobility: Figure 2a–2b report the global satisfaction when the latency budget is fixed to 100 ms and the convoy speed ranges from stand-still to 90 km/h. Across the entire speed range, Full-Edge and Workflow-MAX behave almost indiscernibly: both keep the satisfaction ratio above 97% because sizing on the cell-specific 99$^{\text{th}}$ percentile fully cushions the additional delay caused by mobility. Full-Cloud fares substantially worse; once the convoy accelerates, back-haul queuing pushes the round-trip time beyond the URLLC budget and satisfaction plunges below 60% irrespective of UE density. This is also true for the Workflow-Mean and Workflow-Min solutions, whose performance decreases significantly with speed. Full-Dist-Edge is an intermediate solution in this context, maintaining a relatively high level of performance regardless of the parameter modified.



(a) 3 UEs - Speed 0 km/h (b) 3 UEs - Speed 60 km/h

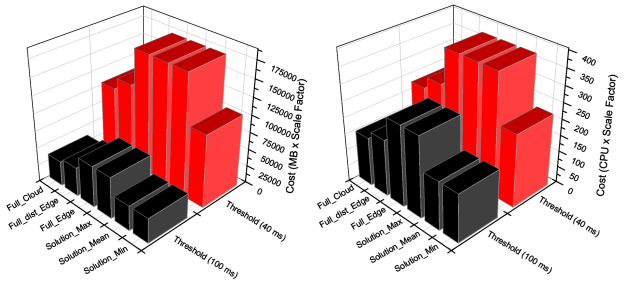
(c) 5 UEs - Speed 0 km/h (d) 5 UEs - Speed 60 km/h

Fig. 3: Threshold and UE number vs. App. satisfaction.

Sensitivity to latency budgets: When the latency budget is relaxed to 120 ms all strategies, even Full-Cloud, satisfy a high number of requests (around 90% for Cloud option); however, as the threshold tightens the differences become manifest. With three UEs per cell (Figure 3a, 3b), Full-Edge, Workflow-MIN and Workflow-MAX collapses below a 60 ms budget whereas the cloud option already fails at 100 ms. At five UEs (Figure 3c, 3d) the gap widens: Full-Cloud exhibits low performance even for the 120ms limit. The most effective solutions in this scenario are also the same as in case 3UE, although they drop to 40 ms, which can be explained by the relatively high communication latencies that currently exist in environments such as SRSRAN. However, being in an identical environment, it is still possible to draw conclusions about the performance level of the different solutions.

Provisioned resources: The cost analysis confirms the qualitative picture (Figure 4b, 4a). A full gNB server at every site (Full-Edge) demands 350 core-units and nearly 20 GiB of RAM to host the six services, whereas Workflow-MEAN halves the CPU bill (175 cores) and cuts memory by

nearly 50% (11.4 GiB) without sacrificing QoS for an average number of UEs (3UEs). Switching from Worflow-MEAN to Workflow-MIN trims a further 25% of CPU and 25% of RAM but, as we have seen, it introduces significant performance losses in 5UE cases or for high latency constraints. By contrast, consolidating everything in the CDC indeed minimises RAM but yields unacceptable user experience; the cloud solution is therefore ruled out for FerroMobile.



(a) Total RAM Cost (MB-Units) (b) Total CPU Cost (Core-Units)

Fig. 4: Competitor and Threshold vs. Provisioned resources.

Key take-aways: Different practical conclusions:

- Full-Edge is economically unsustainable: Placing each container at the gNB level ensures excellent service quality, but requires significantly more cores and memory than any other option. When the latency budget exceeds a few tens of milliseconds this additional footprint no longer improves user experience, so the extra servers translate into pure, and entirely avoidable, CAPEX/OPEX.
- Workflow-MEAN strikes the best cost–performance balance: Sizing on the average per-cell delay delivers satisfaction ratios that stay high, while halving the number of cores and DIMMs to purchase compared to Full-Edge. In addition, MEAN relies on much lighter measurement campaigns than MAX (no tail probing) and on "typical-day" traffic traces that operators already collect for capacity planning, which eases adoption.
- Workflow-MIN is acceptable only for elastic eMBB traffic: Because it exploits the best observed radio latency, the MIN policy yields the leanest Bill of Materials. However, the data show that its QoS drops sharply as soon as either the UE count or the cruising speed rises. This makes the policy attractive for throughput-oriented services (e.g. infotainment bursts) but risky for URLLC streams.
- A single aggregation tier is not a silver bullet: Full-Dist-Edge performs respectably under relaxed budgets,but it falls apart when it comes to URLLC services. The result confirms that shaving one transport hop is insufficient: per-BS edge services remain necessary for ultra low latency services. Multi-tier right-sizing remains necessary to unlock both performance and cost efficiency.

VI. Conclusions

This work introduced a measurement-driven framework that guides operators from in-situ KPI collection to a latency-compliant, budget-aware Bill of Materials. By embedding network measurements into the placement logic, the workflow closes three gaps of the MEC literature: i) it decides where (or whether) to build an edge node, ii) accounts for heterogeneous application footprints, iii) and enforces hard CAPEX ceilings.

The FerroMobile case study demonstrated the concrete benefits of the approach. Defining a multilevel architecture based in our framework, prior to infrastructure deployment, offers a promising avenue for optimizing both the performance and resource utilization. Future work will extend the framework with energy and carbon objectives, integrate micro-services management, and validate the methodology on additional verticals such as AR/VR streaming and smart-grid control.

VII. Acknowledgement

The work presented was performed in the framework of the French FLEXMOVE project supported by the ADEME agency within the FRANCE 2030 program.

References

[1] M. Narouwa, L. Mendiboure, and al., "Enabling network technologies for flexible railway connectivity," *IEEE Access*, 2024.
[2] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, 2018.
[3] S. Bolettieri, R. Bruno, and E. Mingozzi, "Application-aware resource allocation and data management for mec-assisted iot service providers," *Journal of Network and Computer Applications*, vol. 181, 2021.
[4] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "A sample average approximation-based parallel algorithm for application placement in edge computing systems," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 198–203.
[5] R. Li, Z. Zhou, and al., "Joint application placement and request routing optimization for dynamic edge computing service management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, 2022.
[6] N. Waqar, S. A. Hassan, and al., "Computation offloading and resource allocation in mec-enabled integrated aerial-terrestrial vehicular networks: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, 2022.
[7] Q. Liu, R. Mo, X. Xu, and X. Ma, "Multi-objective resource allocation in mobile edge computing using paes for internet of things," *Wireless networks*, vol. 30, no. 5, pp. 3533–3545, 2024.
[8] M. A. Khan, E. Baccour, and al., "A survey on mobile edge computing for video streaming: Opportunities and challenges," *IEEE Access*, vol. 10, pp. 120 514–120 550, 2022.
[9] A. Khakimov, I. A. Elgendy, and al., "Flexible architecture for deployment of edge computing applications," *Simulation Modelling Practice and Theory*, vol. 114, 2022.
[10] K. Poularakis, J. Llorca, and al., "Service placement and request routing in mec networks with storage, computation, and communication constraints," *IEEE/ACM Transactions on Networking*, vol. 28, 2020.
[11] Y. Chen, S. Zhang, and al., "Locus: User-perceived delay-aware service placement and user allocation in mec environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 7, 2021.
[12] H.-W. Tseng, T.-T. Yang, and al., "An mec-based vnf placement and scheduling scheme for ar application topology," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021.
[13] M. Narouwa, L. Mendiboure, S. Maaloul, N. B. Dia, H. Badis, D. M. Molla, M. Berbineau, and R. Langar, "Mobility in 5g emulators: Implementation and evaluation," in *2025 IEEE 28nd international symposium on real-time distributed computing (ISORC)*. IEEE, 2025.
[14] D. M. Molla, C. Hadji, S. Maaloul, L. Mendiboure, M. Berbineau, and H. Badis, "Evaluation of v2x technologies for the connectivity of small autonomous vehicles on secondary railway lines," *IEEE Access*, 2025.
[15] G. Brown, P. Analyst, and H. Reading, "Ultra-reliable low-latency 5g for industrial automation," *Technol. Rep. Qualcomm*, vol. 2, 2018.