Edge-Aware Zero-Touch Slice Reconfiguration for Cross-Layer Resource Optimization in 5G Networks

Raul Barbosa*§, Paulo Duarte*, João Silva*, João Gameiro §,
Marco Araújo*§¶, Susana Sargento†§, Petia Georgieva†‡, Pedro Rito†§
*Capgemini Engineering, Porto, Portugal
†Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, Portugal
‡Institute of Electronics and Informatics Engineering of Aveiro (IEETA)
§Instituto de Telecomunicações, Universidade de Aveiro, Campus Universitário de Santiago, 3810-193, Aveiro, Portugal
¶Portucalense University, Research on Economics, Management and Information Technologies, REMIT, Porto, Portugal

***Abstract*—Modern 5G networks require intelligent, autonomous orchestration to sustain ultra-low latency services and dynamic workloads. This paper presents a fully automated, cross-layer orchestration framework that integrates machine learning (ML)-based edge load forecasting with coordinated slice reconfiguration across edge, core, and RAN domains. A multivariate Random Forest regression model, trained on synchronized edge and RAN metrics, predicts CPU overloads 60 seconds in advance, enabling proactive, closed-loop orchestration—covering slice migration, UPF/SMF reassignment, and PRB profile adjustment through 3GPP-compliant interfaces, all without human intervention. The framework is validated on a real private 5G standalone (SA) testbed under diverse workload and stress scenarios. Forecast-triggered actions complete in 4.78 seconds on average, with service disruption under 1 second, demonstrating both responsiveness and compatibility with existing 5G deployments. This work delivers a practical, standards-compliant demonstration of predictive, zero-touch orchestration. Results confirm the effectiveness of forecast-driven, cross-layer adaptation for preserving service quality, and the approach is designed for scalable application to multi-UE, mobile, and large-scale 5G environments.**

***Index Terms*—ZSM, Slice Reconfiguration, Edge Computing, Machine Learning, Cross-Layer Optimization**

I. Introduction

As 5G networks evolve to meet the demands of dynamic services and ultra-low latency applications, zero-touch management becomes essential to minimize manual intervention and ensure continuous Quality of Service (QoS). This vision, aligned with Zero-Touch Network and Service Management (ZSM), relies on closed-loop orchestration that adapts to real-time network conditions.

Edge computing plays a central role in this shift by bringing computation closer to end-users, enabling latency-sensitive applications such as industrial automation and real-time media streaming [1], [6]. However, the proximity and resource constraints of edge nodes introduce orchestration challenges, particularly under fluctuating workloads and tight timing constraints [2], [7].

This paper presents a fully automated orchestration framework for 5G networks that proactively forecasts edge CPU load and triggers cross-layer reconfiguration. A multivariate Random Forest regression model, trained on synchronized edge and RAN metrics (including lagged and rolling features), predicts CPU overloads 60 seconds in advance. The orchestration logic then selects the closest underloaded edge node and reassigns the UE to its corresponding slice, combining UPF/SMF reassignment via the AMF with PRB profile reallocation at the RAN. This design enables runtime (Day-2) adaptation of both compute and radio resources, ensuring service continuity under dynamic conditions.

To minimize disruption, the system pre-instantiates applications at the target edge node, enabling fast reallocation and sub-second UE handover. The orchestration logic leverages standards-compliant 3GPP interfaces and runs in a real private 5G SA testbed configured with stress scenarios and realistic workload patterns. The architecture is designed for scalability, supporting extension to multi-UE, mobility, and large-scale distributed deployments.

The main contributions of this work are:

- A real-time ML forecasting module trained on synchronized edge and RAN metrics for proactive orchestration;
- A fully automated control pipeline that selects the optimal target slice based on predicted load and location;
- Joint compute and radio resource adaptation through coordinated slice migration and PRB reconfiguration;
- A preparation mechanism that ensures rapid migration and low service disruption;
- Experimental validation on a real 5G infrastructure under diverse workloads, with an architecture designed for scalability and mobility support.

The remainder of the paper is organized as follows: Section2 reviews related work. Section3 presents the system design. Section4 details the orchestration behavior and evaluation results. Section5 concludes and outlines future directions.

II. Related Work

As 5G networks evolve toward intelligent and autonomous operation, recent research has increasingly focused on predictive orchestration and real-time resource management. While forecasting, slice adaptation, and edge intelligence have been

explored individually, few studies address their integration in a unified, closed-loop system capable of operating in real-world 5G environments. This section reviews prior contributions across four foundational areas: (i) ML-based forecasting for edge systems, (ii) slice orchestration in 5G, (iii) edge-5G integration, and (iv) zero-touch management.

ML for Edge Load Forecasting: Accurate short-term forecasting is critical for proactive orchestration in dynamic edge environments. Liu et al. [2] used deep LSTMs to capture CPU load dynamics in MEC settings. Djigal et al. [7] provided a survey of ML techniques for congestion prediction, emphasizing their role in early overload detection. Hua et al. [4] reviewed edge-AI strategies, highlighting lightweight and privacy-preserving solutions such as federated learning. However, these efforts focus on forecasting in isolation and lack orchestration integration or real-time control mechanisms.

Slice Orchestration in 5G: Slice management has emerged as a key enabler of differentiated QoS in 5G. Tsourdinis et al. [1] used LSTM models for RAN-aware slicing; Jain et al. [6] applied deep reinforcement learning for core slice allocation; Vidhya et al. [9] used multi-agent DRL for VNF migration; and Dutta et al. [5] leveraged federated learning to forecast slice-level load. Still, these solutions often focus solely on core-layer control or simulation environments, without addressing edge or RAN dynamics in operational deployments.

Edge–5G Integration: The convergence of edge computing and 5G is essential for supporting low-latency, high-reliability services. Zhou and Chen [12] discussed edge intelligence in IoT-centric 5G systems. Pasupuleti [11] examined edge AI in real-time healthcare and industry applications, while Satyam et al. [10] outlined orchestration challenges in constrained edge environments. These works highlight the role of edge-RAN co-design, but typically lack dynamic orchestration mechanisms driven by system forecasts.

Zero-Touch Management (ZSM): ZSM envisions fully autonomous networks capable of self-configuration and self-optimization across edge, core, and RAN domains [13]. This vision is especially relevant given the increasing complexity introduced by NFV, SDN, MEC, and slicing. However, practical implementations that unify ML-based forecasting with real-time orchestration using standard 3GPP interfaces remain limited.

Positioning of This Work: In contrast to existing solutions, our approach integrates real-time forecasting with operational orchestration across all network layers—edge, core, and RAN—in a testbed-validated deployment. The framework uses real-world edge and RAN metrics to trigger slice reallocation and PRB adaptation with no human intervention. It closes the loop between prediction and reconfiguration, aligning with the ZSM vision while demonstrating end-to-end compatibility with standard 5G components. This positioning highlights the novelty of combining forecasting, orchestration, and standards-compliant interfaces in a unified deployment. Table I summarizes key functional gaps addressed by our cross-layer design.

TABLE I: Comparison of capabilities across representative prior works and the proposed cross-layer architecture.

Capability	[2]	[1]	[6]	[5]	**This Work**
Edge Forecast	✓	–	–	–	✓
Edge Metrics	✓	–	–	–	✓
RAN Metrics	–	✓	–	–	✓
Core Slice	–	–	✓	✓	✓
RAN Slice	–	✓	–	–	✓
Cross-layer	–	–	–	–	✓
Testbed	–	✓	–	–	✓

III. System Design

The proposed architecture enables intelligent, zero-touch orchestration by forecasting edge load using real-time edge and RAN metrics and autonomously reconfiguring network slices across core and RAN domains. Initial provisioning is handled by an End-to-End Orchestrator (MANO), ensuring that each slice is correctly instantiated and aligned with service-level objectives. Building upon this baseline, our system focuses on Day-2 operations—monitoring runtime conditions and dynamically reassigning UEs to the most suitable slice in response to predicted congestion.

This cross-layer design achieves closed-loop orchestration where machine learning forecasts directly inform configuration changes without human intervention. In particular, it supports proactive core migration and RAN-level PRB adjustments, minimizing service disruption and maintaining QoS. The orchestration pipeline operates in two phases:

- **Offline phase:** data collection and model training using real testbed metrics.
- **Runtime phase:** real-time forecasting triggers autonomous slice and bandwidth reallocation.

This section details both the system architecture and data acquisition setup. Fig. 1 provides an overview of the integrated orchestration logic.

Edge Node Monitoring and Forecasting. Each edge node runs latency-sensitive applications (Apps) such as optical character recognition (OCR), and continuously monitors system-level metrics (CPU, memory, socket states) and RAN metrics (RSRP, SNR, latency, throughput). A local QoS Prediction module processes this data in real time to forecast the CPU load (`node_load1`) into the future.

Although the approach supports configurable forecasting windows, a short time horizon was chosen in this work, but it needs to accommodate the full reconfiguration pipeline: decision-making, slice configuration, UE deregistration, and re-registration. Shorter horizons may not provide sufficient time for orchestration, while longer ones may compromise accuracy under dynamic workloads. The forecasting model was trained using data from the real testbed described below.

APP Orchestrator and Slice Reconfiguration. When the forecasting module predicts an overload at a given edge node, a notification is sent to the APP Orchestrator — a central control component responsible for initiating the slice migration process. Upon receiving this alert, the orchestrator evaluates

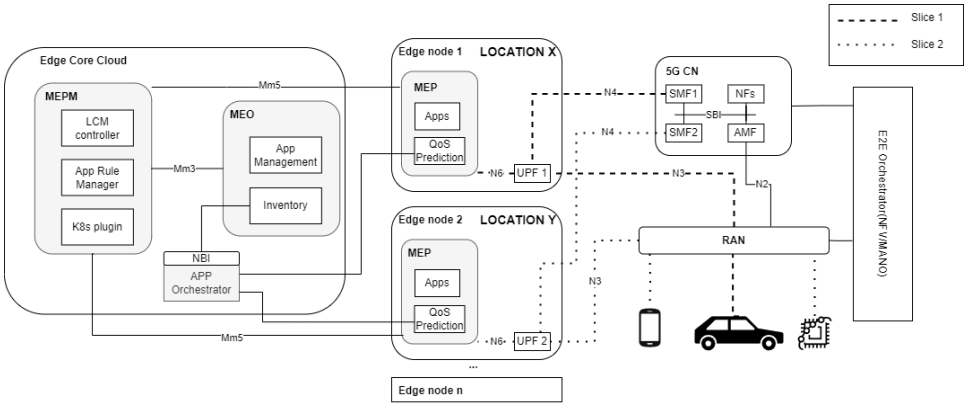


Fig. 1: System architecture: integration of edge forecasting, orchestration logic, and 5G Core interfaces.

the availability of other edge nodes, and decides whether to offload the affected UE to a different slice associated with a less loaded edge location.

Unlike traditional solutions that require manual intervention or fixed policies, the orchestrator automatically updates the slice configuration associated with the UE. Specifically, it modifies the SST (Slice/Service Type) value linked to the IMSI of the UE in the 5G Core's configuration backend (either directly in the database or via the WebUI). This operation ensures that, when the UE re-registers, the new session will be established under the newly selected slice with updated SMF and UPF entities.

Core Triggering and UE Re-registration. To finalize the slice migration, the orchestrator sends a request to the AMF to trigger a UE re-registration procedure. Once the AMF receives the request, it initiates the de-registration of the UE from the current session and prompts a new registration cycle, resulting in the establishment of a fresh PDU session under the updated slice configuration. This mechanism ensures that the transition to a new slice — including the underlying SMF and UPF — is performed transparently and without requiring UE-side changes. The entire operation leverages existing 5G service-based interfaces (SBI), preserving compatibility with standard network function implementations such as Open5GS.

Preparation of the Target Edge Node. To ensure minimal downtime during slice migration, the target edge node must be prepared in advance to host the application workloads. This preparation is handled by the *App Management* module within the MEC Orchestrator (MEAO/M), which manages the deployment of services to edge nodes using standard container-based orchestration. Each node is onboarded via the MEC Platform Manager (MEPM), which uses cluster configuration files to validate available resources (CPU, memory, bandwidth) and to orchestrate application lifecycle events (e.g., deployment, updates). Applications are pulled from a centralized App Store and deployed via a cloud connector that configures forwarding rules through the UPF, ensuring proper data plane redirection post-migration. The MEC Platform (MEP) on the edge node provides the runtime environment for the migrated applications, exposing lifecycle and registration services to the App Management module. By preparing the target edge node prior to forecast-triggered reconfiguration, the system enables seamless UE handover and application continuity across slices.

RAN-Level Bandwidth Adaptation. In parallel with core-level migration, the system reconfigures the PRB allocation associated with the UE. Each slice (SST/SD) is mapped to a specific PRB quota at the gNB. Upon re-registration, the UE receives the appropriate configuration, and the RAN enforces the new bandwidth profile. This ensures that resource adaptation occurs in both compute and radio domains, closing the orchestration loop.

Data Collection Setup. To train the forecasting model, we deploy a controlled 5G Standalone (SA) environment using Open5GS (Docker), a CableFree gNB, and a lightweight K3s edge cluster with one master and one worker node. The master node hosts a Prometheus server to collect and store performance metrics from the nodes and deployed applications. The worker node runs two key applications: an EasyOCR server for text recognition and an iPerf server. Additionally, an NG-RAN metrics server interacts with the CableFree gNB's remote API and gathers traffic metrics. The UE is a Raspberry Pi4 with a SIMCom Modem, which acts as an interface between the Raspberry and the 5G network. The UE generates dynamic workload via three client applications- traffic simulation with OCR, Iperf, and Ping for edge node performance monitoring. Fig. 2 illustrates the topology of the data collection setup. The system is designed to introduce varying degrees of resource contention over time, enabling the capture of edge CPU load fluctuations and radio performance degradation. Metrics are collected at two levels: **Edge:** CPU load, memory usage, socket stats, and network I/O from Kubernetes endpoints. **RAN:** Throughput, MCS, CQI, RI, latency, and signal metrics exported from the gNB.
The resulting dataset serves as the foundation for the feature engineering and model training pipeline presented in Section IV. The trained and validated model is integrated into the runtime architecture to enable proactive orchestration.

Scalability and Generalization Considerations. Although

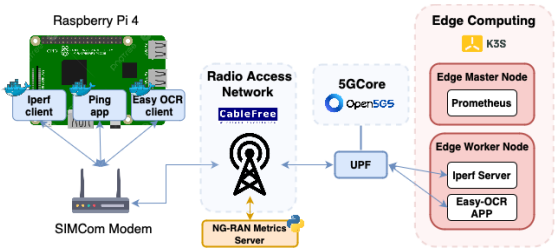


Fig. 2: Topology of the controlled data collection setup used to train the forecasting model.

the testbed focuses on a single UE scenario for controlled validation, the architecture is designed to scale across multiple UEs and edge nodes. The forecasting module and orchestration logic are modular and stateless, enabling horizontal scaling via distributed deployment. Future extensions will incorporate UE mobility, handover scenarios, and load balancing strategies across a larger edge federation. In scenarios where multiple slices experience simultaneous predicted overload, the orchestrator applies a prioritization policy based on QoS class indicators (5QI) and service-level agreements (SLAs). Slices carrying latency-sensitive traffic are reassigned first, while less critical flows may be rate-limited or queued until capacity becomes available. Fairness constraints are enforced to avoid starvation of lower-priority slices.

IV. Forecasting and Evaluation

The trained model operates in real time, enabling a closed-loop orchestration mechanism where ML-driven forecasting directly triggers slice reconfiguration. This approach aligns with ZSM principles, emphasizing proactive control and minimal human intervention. The pipeline is designed with modularity in mind, separating forecasting, decision-making, and orchestration to allow scalable deployment and future extensibility. The full system is validated under realistic conditions on a private 5G infrastructure supporting multiple slices, edge nodes, and UEs. We describe the steps involved in data preparation, model training, orchestration triggering, and the obtained results.

A. Feature Engineering and Forecasting

The forecasting task is formulated as a multivariate regression problem to predict the edge CPU load (`node_load1`) at $t + 60$ seconds. This lookahead window was selected as a practical trade-off: it provides sufficient time to execute the orchestration pipeline (forecasting, orchestration decision, application migration, and UE re-registration) while maintaining prediction accuracy under dynamic workloads. The horizon is configurable and can be adapted in future deployments based on latency constraints or workload volatility.

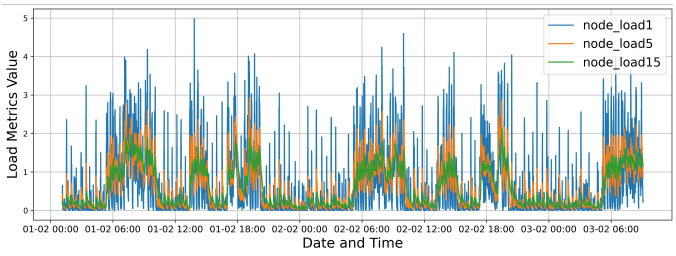
Data Integration and Preprocessing: two data sources are combined: edge-level metrics collected from Kubernetes endpoints, and RAN metrics retrieved from the CableFree gNB. After timestamp alignment, the combined dataset is obtained with synchronized edge and RAN information. An initial step removes variables with low information gain, such as unique identifiers (e.g., `rnti`, `ran_ue_id`), constant metrics (e.g., `dl_err`, `ul_err`), irrelevant logs (e.g., `language`). The reduced dataset retains only metrics with potential predictive relevance, such as load, memory, TCP activity, bitrate, retransmissions, latency, and signal quality.

Missing Values: A targeted imputation strategy is applied based on variable type and expected behavior:

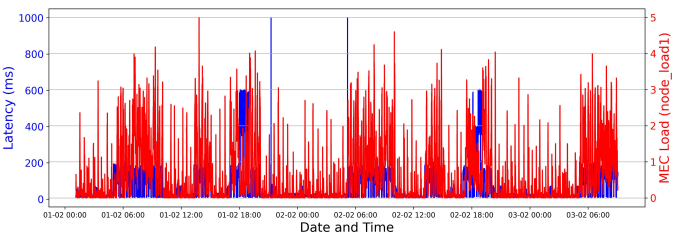
- Continuous RAN metrics with low missing rates (e.g., `ul_mcs`, `turbo_decoder_avg`) are filled with the mean.
- Log-derived metrics with sparse activity (e.g., bandwidth, `parallelism`, `congestion_iperf`) are forward-filled to simulate persistent behavior during activity.
- OCR detection logs (e.g., `detection_repetitions`, `nr_detections`) are filled with zeros, assuming that the absence of a record implies no detection.
- Categorical variables (e.g., `congestion_ocr`) are set to ”UNKNOWN” to preserve data integrity without introducing artificial structure.

Outlier Analysis: Most outliers (e.g., in `node_load1`, `latency`) reflect genuine overloads and are retained. Only `turbo_decoder_avg` is winsorized to cap extreme, non-physical spikes.

Exploratory Trend Analysis: To characterize the node load behavior over time, the `node_load1` metric is plotted alongside 5-minute and 15-minute averages, as shown in Figure 3 (a). Although not used as a prediction target, UE latency is analyzed as a reference for QoS degradation. Figure 3 (b) shows that, while not identical, latency and load trends are aligned, confirming the operational impact of CPU saturation.



(a) Temporal evolution of edge CPU load.



(b) Temporal correlation between UE latency and edge load.

Fig. 3: Exploratory analysis of edge load and latency trends.

Feature Selection via Correlation: Spearman correla-

tion was applied, retaining features with $|\rho| \geq 0.20$ to `node_load1_t+60`. The selected set (Table II)—including CPU load history, memory usage, TCP socket activity, throughput, retransmissions, and radio link quality—was used to train the forecasting model. All continuous variables were normalized using `RobustScaler`.

TABLE II: Features with correlation $\geq$ 0.20 to `node_load1_t+60`.

Feature	Correlation
node_load1	0.809
node_load1_rolling60	0.761
node_load5	0.682
node_load1_lag60	0.624
node_load15	0.593
ul_phr	-0.389
node_sockstat_TCP_alloc	0.363
p_ue	0.342
dl_tx	0.329
ul_tx	0.328
dl_bitrate	0.328
node_sockstat_TCP_mem	0.320
ul_bitrate	0.318
latency_rolling60	0.298
dl_retx	0.280
latency_lag60	0.256
latency	0.215

B. Forecasting Model

The forecasting task is formulated as a multivariate regression problem to predict the edge CPU load (`node_load1`) over a configurable short-term horizon. In our evaluation, we use a 60-second lookahead as a representative configuration, though this interval can be adapted based on system responsiveness and application requirements. We evaluate a range of regression models encompassing linear baselines, ensemble-based learners, gradient boosting methods, and a deep learning approach. The objective is to identify a model with high predictive accuracy and generalization capacity for real-time deployment. Each model is trained on 60% of the data, with 20% used for validation and 20% reserved for testing. Hyperparameter optimization is performed using `RandomizedSearchCV`. Table III summarizes the performance metrics across training and test sets, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R^2).

CatBoost and Random Forest emerged as the top-performing models. CatBoost achieved the best training accuracy (RMSE = 0.3206, R^2 = 0.8681), while Random Forest generalized better, with the lowest test RMSE (0.3842) and highest R^2 (0.7716). Considering the trade-off between accuracy, stability, and runtime performance, Random Forest was selected for deployment in resource-constrained edge environments. The final Random Forest configuration, obtained via `RandomizedSearchCV`, used 400 trees, maximum depth of 10, a minimum of 10 samples per leaf, and the square-root rule for feature selection at each split.

Figure 4 shows how the model predictions follow the actual `node_load1_t+60` trend, capturing both rapid surges and stable intervals. The model runs every 60 seconds and triggers reconfiguration when predictions exceed a dynamic threshold based on historical distribution—removing reliance on static, manual rules and enabling adaptive orchestration in response to real-time load dynamics.

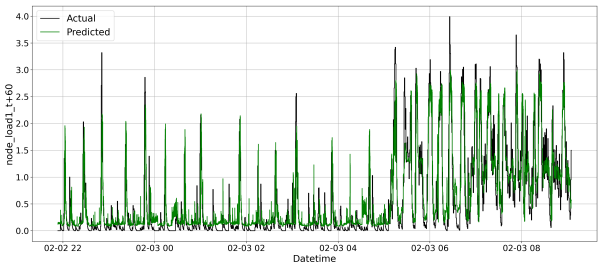


Fig. 4: Forecasted vs. actual `node_load1_t+60` values.

C. Slice Reconfiguration Logic

The orchestration trigger is activated when the predicted CPU load exceeds 75% of the node's capacity. This percentile-based threshold was empirically chosen to provide a safety margin for orchestration actions to complete before reaching saturation, while avoiding excessive migrations under normal transient load variations. When an overload is predicted, the App Orchestrator autonomously initiates slice reallocation. It prepares the new edge node with containerized services, updates the UE's SST in the 5G Core to point to a different UPF/SMF and PRB profile, and triggers re-registration via a standard AMF request. The UE then reconnects under the new slice, restoring service continuity with minimal delay.

This fully autonomous reallocation process enables coordinated compute and radio adaptation with no human intervention—fulfilling the core tenets of zero-touch orchestration by reacting to predicted conditions in real time.

D. System Behavior and Results

Two experiments validate the end-to-end orchestration in a real 5G setup, focusing on (i) responsiveness to predicted overload and (ii) QoS impact of slice selection.

1) Responsiveness to Forecast Trigger: When overload is predicted, the QoS module triggers the full pipeline: edge instantiation, slice update, and UE re-registration. The process averaged 4.78 seconds (0.983s of disruption), outperforming simulated VNF migration times (5.4s) [9] and remaining well within the 60s forecast window.

To evaluate the impact of slice migration on QoS, we conducted two complementary experiments whose results are summarized in Figure 5.

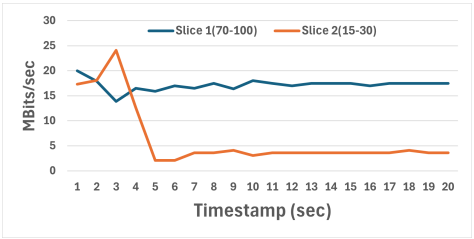
2) Impact on Throughput: The UE was tested on two slices: high PRB (70–100) and low PRB (15–30). As seen in Fig. 5a, the high-PRB slice sustained >15 Mbit/s, while the low-PRB slice dropped to 3–5 Mbit/s. This confirms the effectiveness of PRB enforcement and the architecture's ability to maintain throughput via forecast-based reassignment.

TABLE III: Forecasting model performance on training and test sets.

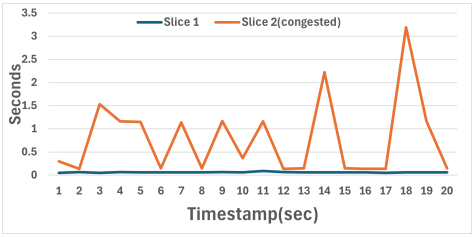
Model	Train RMSE	Test RMSE	Train MAE	Test MAE	Train R^2	Test R^2
Linear Regression	0.4649	0.4383	0.3046	0.2795	0.7226	0.7280
Ridge Regression	0.4653	0.4391	0.3055	0.2803	0.7222	0.7017
Random Forest	0.3302	**0.3842**	0.2082	0.2454	0.8601	**0.7716**
XGBoost	0.3466	0.3869	0.2188	0.2429	0.8458	0.7683
LightGBM	0.3596	0.3896	0.2270	0.2463	0.8340	0.7651
CatBoost	**0.3206**	0.3848	**0.2030**	0.2396	**0.8681**	0.7708
LSTM	0.3977	0.3916	0.2418	**0.2382**	0.7971	0.7627

3) Impact on Latency: In a separate experiment, latency was monitored while the UE consumed an app hosted on two edge nodes: one operating normally and another under artificial stress. As shown in Figure 5b, the overloaded node induced significant latency spikes (over 3 seconds), while the lightly loaded node maintained stable latency below 0.2 seconds. This confirms that edge congestion directly affects application responsiveness and supports the use of `node_load1` as a proactive signal for slice reallocation.

Together, these results demonstrate that the proposed cross-layer approach can mitigate service degradation by dynamically adapting both compute and radio resources. While prior works have addressed edge load forecasting, slice orchestration, and RAN resource allocation, they typically treat these dimensions independently, using simulated environments or focusing on a single control layer. In contrast, our system closes the loop by combining forecast-driven triggers with automated slice migration and differentiated PRB enforcement across edge, core, and RAN layers, implemented on a standard-compliant 5G testbed.



(a) Throughput in slices with different PRB allocations.



(b) Latency impact under edge node load conditions.

Fig. 5: QoS variation across slices

V. Conclusion and Future Work

We presented a proactive orchestration framework for private 5G networks, where real-time edge load forecasting predicts CPU overloads 60 seconds in advance and triggers fully automated slice migration and RAN reconfiguration across edge, core, and RAN layers. Validation on a real 5G SA testbed showed the reconfiguration pipeline completes well within the forecast window (avg. 4.78 s) with sub-second service disruption. Differentiated PRB profiles sustained QoS under load, demonstrating the benefits of joint compute/radio adaptation. Future work will address multi-UE and mobility scenarios, high-scale performance, and RAN Intelligent Controller (RIC) integration, as well as assess forecasting model generalizability and orchestration scalability in heterogeneous deployments.

References

[1] T. Tsourdinis, et. al, "AI-driven Service-aware Real-time Slicing for beyond 5G Networks," in *Proc. IEEE INFOCOM Workshops*, 2022, pp. 1–6, doi: 10.1109/INFOCOMWKSHPS54753.2022.9798391.

[2] Z. Liu, et al., "Load Prediction in Edge Computing Using Deep Auto-Regressive Recurrent Networks," in *Proc. IEEE ICC*, 2023, pp. 809–814, doi: 10.1109/ICC45041.2023.10278924.

[3] E. Kartsakli et al., "AI-Powered Edge Computing Evolution for Beyond 5G Communication Networks," in *Proc. EuCNC/6G Summit*, 2023, pp. 478–483, doi: 10.1109/EuCNC/6GSummit58263.2023.10188371.

[4] H. Hua et al., "Edge Computing with Artificial Intelligence: A Machine Learning Perspective," *ACM Comput. Surv.*, vol. 55, no. 9, Article 184, pp. 1–35, 2023, doi: 10.1145/3555802.

[5] N. Dutta, et al., "Federated learning framework for prediction based load distribution in 5G network slicing," in *Proc. IC3*, 2024, pp. 421–426, doi: 10.1145/3675888.3676085.

[6] A. Jain, et al. "AI-Driven Dynamic Network Slicing for Resource Optimization in 5G Networks: Implementation and Performance Evaluation," in *Proc. CYBERCOM*, 2024, pp. 720–724, doi: 10.1109/CYBERCOM63683.2024.10803129.

[7] H. Djigal, et al. "Machine and Deep Learning for Resource Allocation in Multi-Access Edge Computing: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2449–2494, 2022, doi: 10.1109/COMST.2022.3199544.

[8] O. Jouini et al., "A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions," *Technologies*, vol. 12, no. 6, Article 81, 2024, doi: 10.3390/technologies12060081.

[9] V. P. et al., "Dynamic network slicing based resource management and service aware Virtual Network Function (VNF) migration in 5G networks," *Computer Networks*, vol. 259, pp. 111064, 2025, doi: 10.1016/j.comnet.2025.111064.

[10] Satyam, P., et al., "AI-Enabled Edge Computing Models: Trends, Developments, and Future Implications," in *Proc. ICECAA*, 2023, pp. 63–67, doi: 10.1109/ICECAA58104.2023.10212294.

[11] M. K. Pasupuleti, "AI-Enabled Edge Computing: Revolutionizing IoT with Real-Time Optimization," in *Next-Gen Edge Systems*, Nov. 2024, pp. 29–64, doi: 10.62311/nesx/46687.

[12] Y. Zhou, et al., "Edge Intelligence: Edge Computing for 5G and the Internet of Things," *Future Internet*, vol. 17, no. 3, Article 101, 2025, doi: 10.3390/fi17030101.

[13] P. Simoens, et al., "Zero-touch network and service management: A survey," *Computer Networks*, vol. 205, pp. 108693, 2022, doi: 10.1016/j.comnet.2022.108693.