Enabling Near Real-Time Slicing in Open-Source O-RAN Testbed using Dynamic Resource Allocation

Iker Hernández, Zaloa Fernández
Fundación Vicomtech Basque Research
and Technology Alliance, San Sebastián, 20009, Spain
Email: {ihernandez, zfernandez}@vicomtech.org

Alfonso Gómez, Andrea Campos, Juan Jose Fernández
Gamma Solutions, Cáceres, 10004, Spain
Email: {alfonso.gomez, andrea.campos,
juanjose.fernandez}@diggia.com

***Abstract*—The Open Radio Access Network (O-RAN) architecture is a fundamental paradigm for 5G and Beyond, promising programmable, service-driven resource management. This flexibility is essential to satisfy diverse Quality of Service (QoS) requirements through network slicing. However, realizing efficient closed-loop control in O-RAN infrastructures remains an open research challenge. Conventional implementations often lack the granular scheduler interfaces required to map application-layer demands to radio resources in real-time. To address this gap, this paper introduces a device-centric dynamic slicing framework specifically designed for private 5G Non-Public Networks (NPNs). This framework is built upon an extended E2 Radio Access Network Control (RC) Service Model (SM) and dedicated xApps. This mechanism augments the scheduler's capabilities, enabling dynamic Physical Resource Block (PRB) limit allocation at the Transmission Time Interval (TTI) level. This logic has been integrated into an end-to-end 5G Standalone (SA) testbed combining OpenAirInterface (OAI), FlexRIC, real-time RAN telemetry via a monitoring xApp, and UE-embedded Deep Packet Inspection (DPI) for traffic classification. The proposed solution remains compatible with O-RAN specifications and is experimentally validated, demonstrating effective Service Level Agreement (SLA) enforcement with deterministic algorithmic processing ($\approx 30\mu s$) and minimal scheduling overhead ($< 0.2\%$ of the TTI budget), confirming its viability for latency-sensitive industrial environments and demonstrating significant isolation improvements over conventional Proportional Fair scheduling.**

***Keywords*—O-RAN, Network Slicing, xApp, OpenAirInterface, FlexRIC, Resource Allocation.**

I. Introduction

The strict determinism required by Industry 4.0 applications fundamentally conflicts with the architectural design of standard public mobile networks. Traditional Radio Access Network (RAN) schedulers utilize algorithms like Proportional Fair (PF), which optimize for aggregate spectral efficiency by prioritizing users with favorable channel conditions. While effective for mobile broadband, this approach violates the critical Service Level Agreements (SLAs) of industrial Operational Technology (OT), encompassing systems such as Programmable Logic Controllers (PLCs) or robotic actuators, where isochronous control loops require guaranteed packet delivery regardless of channel quality [1]. Consequently, the network objective must shift from maximizing cell throughput to strictly enforcing per-machine resource isolation.

This necessity drives the adoption of Non-Public Networks (NPNs) in critical sectors. Recent techno-economic studies identify seaport terminals, automotive factories, and power plant campuses as the primary candidates for private 5G deployments, as these environments require strict resource isolation and reliability levels (up to 99.9999%) that standard public slicing cannot consistently guarantee [2].

The Open Radio Access Network (O-RAN) architecture has emerged as a key enabler for such customization, decoupling control logic from hardware execution. The capabilities of the Near-Real-Time RAN Intelligent Controller (Near-RT RIC) and the O-RAN architecture itself have been extensively studied in recent literature [3], [4]. However, existing research on xApps has predominantly focused on maximizing aggregate network throughput or spectral efficiency, as demonstrated in [5]. While beneficial for consumer broadband, these optimization strategies often overlook the connection stability required by critical OT traffic.

Moreover, realizing this potential in practice faces a fundamental architectural conflict stemming from the timescale mismatch between the O-RAN control plane and the radio interface. While the Near-RT RIC is designed to operate on a timescale of 10 ms to 1 s, industrial OT applications require deterministic enforcement at the Transmission Time Interval (TTI) level (sub-millisecond). Direct closed-loop control over the E2 interface introduces transport jitter that often violates these strict timing deadlines. Consequently, conventional implementations resort to static resource partitioning to guarantee isolation [6], avoiding the latency risk but resulting in severe spectral inefficiency when industrial traffic is bursty or idle. Furthermore, standard E2 Service Models (SM) lack the granular interfaces required to implement dynamic logic [7]. Specifically, implementations of these interfaces prevent the application of precise Physical Resource Block (PRB) quotas through the E2 interface due to a lack of direct write attributes exposed within the Medium Access Control (MAC) scheduler. This limitation hinders the realization of true closed-loop automation regardless of the underlying platform.

To bridge this gap, a split-logic architecture that fundamentally decouples global policy adaptation from local radio enforcement is proposed. By introducing a lock-free shared state mechanism within the gNB, high-frequency execution environment has been enabled where resource quotas are applied at the TTI level, effectively shielding the scheduler from the inherent transport jitter of the O-RAN E2 interface. This approach transforms the RAN into an active enforcement point, providing the deterministic stability required to host the modular control logic and high-level optimization agents explored in this work.

The main contributions of this work include:

- First, the development of a closed-loop deterministic enforcement mechanism driven by an extended E2 RAN Control SM (RC SM) [8] and a Slice Manager xApp. This framework advances beyond static partitioning limitations by exposing internal scheduler variables to an asynchronous external control. The solution injects a Dynamic Resource Partitioning logic into the MAC, effectively overriding standard fairness algorithms to strictly enforce PRB quotas at the Transmission Time Interval (TTI) level, regardless of the control plane's transport latency.
- Second, the integration of a device-centric architecture that couples Deep Packet Inspection (DPI) with a dedicated Metric Exporter xApp. Unlike core-centric traffic classification approaches that can become a bottleneck in Industrial IoT environments with multiple heterogeneous User Equipments (UEs) and concurrent application flows, this architecture shifts intelligent slice-selection decisions to the UE. This ensures immediate slice-specific mapping, preventing low-latency control traffic from being queued behind high-bandwidth data while continuously feeding the RAN control loop with real-time telemetry.
- Third, the validation on an end-to-end open-source O-RAN testbed with Commercial Off-The-Shelf (COTS) User Equipments (UE), deploying the solution using OpenAirInterface and FlexRIC. Owing to the modular and technology-agnostic nature of the setup, the experimental results are readily transferable to other O-RAN-compliant testbeds, demonstrating the feasibility of deterministic slicing on physical hardware beyond simulation-based studies.

The remainder of this paper is organized as follows. Section II details the proposed full-stack O-RAN framework and its operational planes. Section III describes the experimental setup, including the hardware topology and the implemented software architecture. Section IV presents the performance evaluation and experimental results. Finally, Section V concludes the paper and outlines future work.

II. Proposed O-RAN Slicing Framework

Building upon the industrial requirements outlined in Section I, this paper presents a full-stack O-RAN slicing framework designed to enable deterministic resource management. The framework is designed to transform the RAN from a passive entity into an active enforcement point that guarantees strict resource isolation while allowing for dynamic capacity adaptation. The proposed framework relies on the O-RAN Alliance reference architecture [9], centering on the capabilities of the Near-RT RIC. This platform decouples control logic from hardware execution, hosting custom xApps that interact with the RAN via the standard E2 Interface.

The high-level system operation is illustrated in Figure 1. The diagram captures the end-to-end traffic flow, from application-aware classification at the UE, through dynamic resource partitioning in the RAN, to logical separation at the 5G Core (5GC), while mapping these stages to three hierarchical operational planes: the Data Plane (detailed in Section II-A) for physical isolation, the Metrics Plane (Section II-B) for real-time observability and the Control Plane (Section II-C) for closed-loop actuation.

Drawing upon recent advancements in network isolation and throughput optimization [10] [11] [12], and considering the architectural requirements for critical traffic handling discussed in [1], we adopted this hierarchical design to maximize transport efficiency while maintaining deterministic isolation boundaries within the 5G ecosystem.

Unlike static partitioning approaches [6], [7], our proposal utilizes an adaptive contiguous allocation mechanism driven by continuous RAN monitoring. The central controller dictates the exact volume of resources required for a slice based on real-time network metrics (eg., channel quality, resource utilization and connection stability), while the local scheduler enforces this quota via strict contiguous block reservation. Crucially, this actuation is coupled with a device-centric traffic steering strategy that classifies application flows at the edge, ensuring that the resource enforcement correlates exactly with the specific Quality of Service (QoS) demands of the industrial endpoints.

A. The Data Plane

The Data Plane constitutes the execution path for user traffic, designed to ensure strict logical separation from the industrial endpoint, through the packet core, and up to the respective external network.

1) Device-Centric Traffic Steering: The architecture necessitates a device-centric steering mechanism to map application flows to their corresponding network slices at the source to leverage the multi-slice capabilities of the UEs. Functionally aligned with the 3GPP UE Route Selection Policy (URSP) standard [13], this component is responsible for intercepting outgoing traffic and applying routing rules based on application identifiers (e.g., App ID, Destination IP/Port). By enforcing this classification at the user equipment, the framework ensures that critical traffic is encapsulated into the correct Protocol Data Unit (PDU) session before entering the radio interface. This strictly isolates high-priority frames from best-effort background data before they reach the gNB schedule, a prerequisite for effective per-slice resource enforcement.

2) Core Network Separation (User Plane): Traffic exiting the RAN is encapsulated into GTP-U tunnels corresponding to the selected slice. These tunnels terminate at isolated User Plane Functions (UPFs), extending the logical separation established at the radio layer through the backhaul.

B. The Metrics Plane

Situated above the execution layer is the Metrics Plane, responsible for synthesizing a holistic view of the state of the network. To address the latency constraints of legacy interfaces, the architecture integrates a dedicated Metric Exporter xApp subscribing to an extended E2 RC SM for granular, TTI-level statistics. The standard E2-KPM model was excluded to avoid the significant ASN.1 serialization overhead associated with high-frequency reporting (1 kHz), which introduces substantial latency in real-time controllers [7]. Instead, a

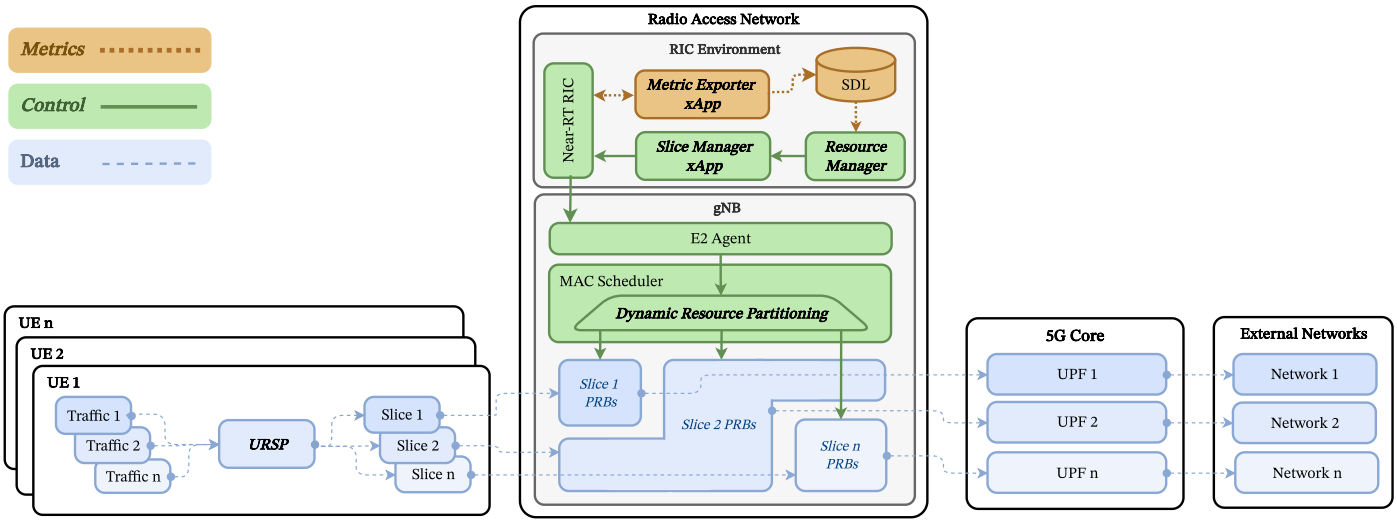

Fig. 1. High-level architecture of the proposed O-RAN slicing framework.

lightweight custom SM enables raw telemetry extraction with minimal computational load on the E2 Agent, achieved by stripping auxiliary protocol overhead in favor of a direct-mapping serialization strategy. This ensures data freshness, allowing the system to instantaneously detect slice saturation and trigger the control loop without the risk of actuation errors caused by stale network states. This telemetry is continuously synchronized with the Shared Data Layer (SDL), effectively decoupling the strict real-time constraints of data acquisition from the asynchronous inference logic."

C. The Control Plane

The Control Plane is responsible for the dynamic dimensioning of slice partitions based on the feedback provided by the Metrics Plane.

1) Dynamic Resource Partitioning (MAC Layer): The critical enforcement point resides within the gNB MAC scheduler. Standard O-RAN implementations often struggle to map asynchronous E2 control messages to the strict real-time constraints of the scheduler without introducing locking latency. To address this, a split-logic resource partitioning mechanism is introduced.

As detailed in Algorithm 1, this mechanism decouples the control plane from the data plane. The *Asynchronous Config Handler* (Procedure 1) operates on the E2 Agent thread, parsing incoming `MAC_CTRL_REQ` payloads and updating a lock-free shared configuration structure with the specific `prb_start` and `num_prbs` dictated by the Resource Manager.

The *Synchronous TTI Scheduler* (Procedure 2) executes within the strict TTI interrupt context. It performs a lock-free read of this shared state at the start of every slot to dynamically calculate the resource partition boundaries, ensuring the MAC thread is never blocked by E2 updates. For Best-Effort traffic, the system calculates a dynamic ceiling ($P_{ceiling}$) based on the highest resource block utilized by any active isolated slice, strictly preventing the scheduler from allocating background traffic within protected partitions.

Algorithm 1 Dynamic Slice Resource Enforcement

Require: Global Slice Config $\mathcal{S}_{cfg}$ (Shared State)
Require: Active User List $\mathcal{U}_{active}$
Procedure 1: Asynchronous Configuration Update
▷ Triggered by E2 Agent thread upon `MAC_CTRL_REQ`
procedure HANDLESLICECONFIG(Msg_{E2})
  $Map \leftarrow$ Parse(Msg_{E2}) ▷ Decodes PRB quotas
  Atomic Update: $\mathcal{S}_{cfg} \leftarrow Map$ ▷ C11 atomic store with seq_cst ordering
end procedure
Procedure 2: Synchronous TTI Scheduler
▷ Executed every TTI in MAC Main Thread
procedure GETPARTITIONLIMITS(User u)
  $i \leftarrow$ GetSliceID(u)
  // Case A: User belongs to an Isolated Slice
  if $\mathcal{S}_{cfg}[i].active$ **and** $\mathcal{S}_{cfg}[i].\texttt{num\_prbs} > 0$ **then**
    return $\{\mathcal{S}_{cfg}[i].\texttt{prb\_start}, \mathcal{S}_{cfg}[i].\texttt{num\_prbs}\}$
  end if
  // Case B: Best-Effort (Dynamic Ceiling)
  // Assumption: Isolated slices are packed contiguously from 0
  $P_{\text{ceiling}} \leftarrow 0$
  for $s \in \mathcal{S}_{cfg}$ **do**
    if $\mathcal{S}_{cfg}[s].active$ **then**
      $End \leftarrow \mathcal{S}_{cfg}[s].\texttt{prb\_start} + \mathcal{S}_{cfg}[s].\texttt{num\_prbs}$
      $P_{\text{ceiling}} \leftarrow \max(P_{\text{ceiling}}, End)$
    end if
  end for
  $\text{Size}_{\text{rem}} \leftarrow \text{BWP}_{\text{total}} - P_{\text{ceiling}}$
  return $\{P_{\text{ceiling}}, \max(0, \text{Size}_{\text{rem}})\}$
end procedure

Enforcing strict contiguous allocation limits the scheduler's ability to exploit frequency diversity across the entire band. However, this trade-off is intentional, as the determinism of guaranteed contiguous isolation in industrial OT environments outweighs the marginal spectral efficiency gains of distributed resource allocation.

2) The Resource Manager: This component functions as a modular inference engine responsible for the precise dimen-

sioning of the radio interface. Architecturally, it is algorithm-independent; it serves as a generic container capable of hosting diverse logic, ranging from deterministic heuristics to Deep Reinforcement Learning (DRL) agents.

However, unlike legacy schedulers that abstract capacity into aggregate bitrates, this component focuses on Spatio-Spectral Isolation. Its primary directive is to translate high-level connectivity metrics (e.g., buffer occupancy and retransmissions), retrieved from the SDL into rigid physical coordinates, specifically, the optimal `prb_start` and `num_prbs`. This output is not merely a suggestion but a deterministic mandate that drives the partitioning mechanism, ensuring that the chosen algorithm, regardless of its complexity, can physically guarantee that critical traffic flows are mapped to clean, uncontested spectrum.

3) Asynchronous Control Loop Architecture: To ensure system stability regardless of transport fluctuations, the control plane is designed using a decoupled asynchronous model. Inside the Near-RT RIC, the Slice Manager xApp pushes resource quotas (`target_prb_count`) taken from the Resource Manager as state updates rather than synchronous per-packet commands. On the RAN side, the MAC scheduler performs a non-blocking read of this shared configuration state at the start of every TTI. This design decouples the strict real-time requirements of the radio scheduler (Operating in μs) from the variable latency of the E2 transport interface, ensuring that the scheduler never stalls while waiting for external controller input.

While the O-RAN architecture standardizes the A1 interface for policy guidance, its reliance on HTTP-based transaction models introduces transport overhead incompatible with TTI-level actuation. Consequently, the control plane is designed utilizing a persistent, unidirectional WebSocket connection from the Resource Manager down to the xApp. This architectural decision explicitly diverges from the standard to eliminate the connection establishment latency inherent in stateless protocols, ensuring that dynamic PRB quotas are synchronized with the Slice Manager xApp with minimal transport jitter.

III. Experimental Setup

To validate the proposed framework, we deployed a fully operational end-to-end O-RAN testbed. This implementation translates the abstract architecture defined in Figure 1 into the deployed setup shown in Figure 2. This section details the physical infrastructure, the software stack, and the methodological design criteria applied to bridge the gap between standard open-source components and strict real-time requirements required to achieve deterministic actuation latency.

The experimental validation focuses on a Smart Maintenance scenario, a representative Industry 4.0 use relying on a private 5G NPN. In this topology, a field operator equipped with a 5G Supervisor Terminal requires seamless access to high-resolution digital CCTV surveillance cameras while simultaneously receiving high-priority safety alarms from nearby machinery and sending time-sensitive control messages. Unlike static sensor setups, this user is mobile and consumes concurrent, contending traffic flows. The framework must dynamically isolate the safety-critical alarm packets and

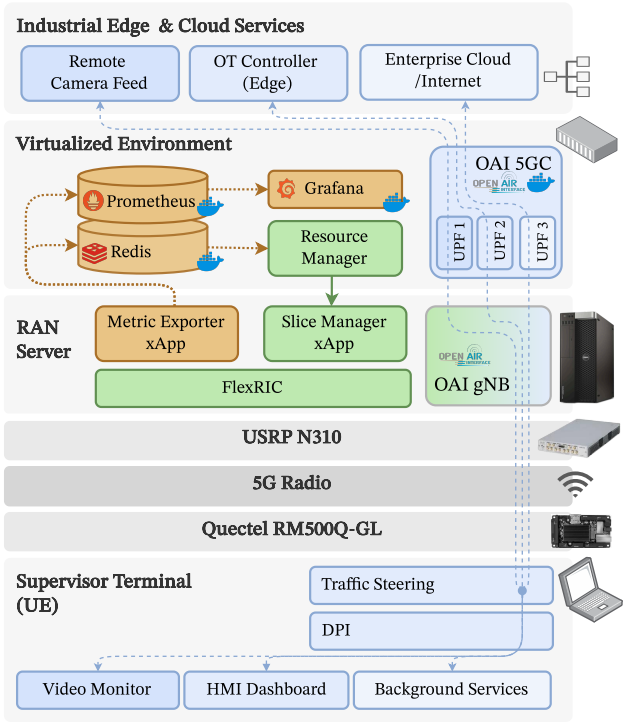

Fig. 2. High-level architecture of the implementation testbed.

CCTV video stream from the background traffic without manual intervention.

A. Physical Infrastructure

To emulate a representative industrial environment, user traffic is generated using a 5G Supervisor Terminal equipped with a commercial UE Quectel RM500Q-GL 5G Module. Unlike software-simulated UEs, the integration of this physical module introduces real-world modulation and demodulation processing times, ensuring realistic latency measurements and channel behavior. Critically, this module supports multi-slice connectivity via a single International Mobile Subscriber Identity (IMSI), enabling simultaneous PDU sessions across distinct network slices. On the network side, the RAN is powered by a high-performance server connected to a Universal Software Radio Peripheral (USRP) N310 Software Defined Radio (SDR). The RF front-end operates in Band n78 (3.5 GHz) with a 40 MHz bandwidth, providing sufficient spectral capacity to support multiple concurrent slice partitions without physical saturation.

B. Software Architecture

The software stack is built upon open-source standards, modified to support the write-access requirements of the Control Plane described in Section II-C.

OpenAirInterface (OAI) v2.2.0 serves as the gNB foundation. To implement the logic defined in Algorithm 1, the `pre_processor` function has been modified within the `gNB_scheduler_dlsch.c` and `gNB_scheduler_ulsch.c` modules. Specifically, the Dynamic Resource Partitioning logic has been implemented immediately prior to the standard Proportional Fair iterator,

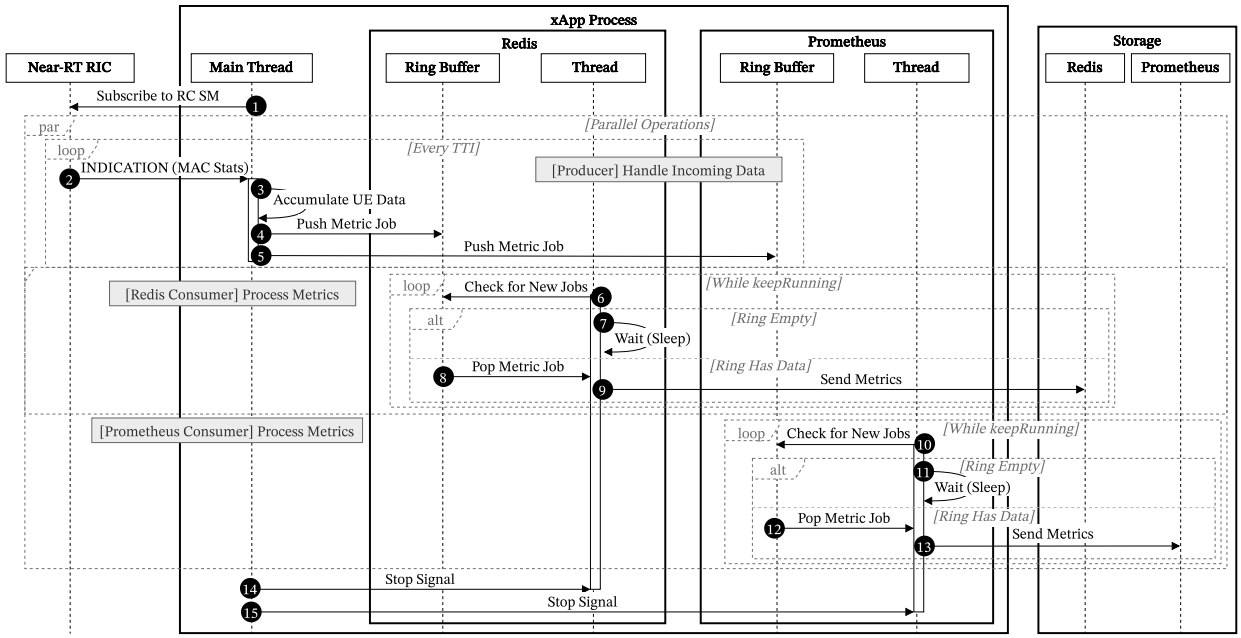

Fig. 3. Sequence diagram of internal xApp metric collection.

ensuring that the resource grid ($rballoc\_mask$) is logically partitioned before any user candidates are evaluated. Thread safety between the asynchronous E2 agent and the real-time MAC scheduler has been maintained via efficient synchronization mechanisms detailed in Section III-E.

FlexRIC has been selected as the E2 termination point due to its lightweight, C-based architecture designed for low-latency adaptation. The custom Slice Manager and Metric Exporter xApps have been implemented in C, communicating with the RAN via the extended RC SM.

The system relies on the OAI 5GC deployed in a Service-Based Architecture (SBA), where distinct UPF containers are instantiated for each network slice to guarantee strict traffic isolation at the GTP-U level.

Regarding the monitoring layer, Remote Dictionary Server (Redis) has been employed as a high-performance, in-memory datastore acting as an ephemeral metric cache. To guarantee determinism, we have used Redis Pipelining to batch telemetry updates. This completely decouples the xApp from the Resource Manager, ensuring that high-frequency telemetry export never blocks the critical E2 processing thread. Finally, a parallel monitoring stack using Prometheus and Grafana has been deployed solely for validation and historical logging, ensuring that visualization tasks do not impact the real-time resource allocator. Finally, extending this non-blocking design to the actuation path, shared state between the high-speed MAC scheduler and the E2 Agent is managed via the shared state mechanism described in Section II-C.

C. Traffic Profiles and Classification

To emulate an industrial floor, we orchestrated heterogeneous traffic flows representing distinct OT behaviors, ranging from isochronous control loops to high-bandwidth video feeds. Table I details the specific mapping of these industrial use cases to their respective network slices, including the traffic generation tools, protocols, and QoS profiles utilized for the evaluation.

TABLE I. FACTORY SUPERVISION TRAFFIC PROFILES, TRAFFIC GENERATION AND SLICE CONFIGURATION

Parameter	Slice 1	Slice 2	Slice 3
Service Tier	eMBB	URLLC	Best-Effort
Use Case	CCTV Surveillance	Machine Alarms	Admin. Data (Blueprints)
Network Source	Remote Cloud NVR	Local OT Controller (Edge)	Remote Enterprise Cloud
Data Rate	20–50 Mbps (4K Video)	$\approx$ 6 KB/s (64B/10ms)	Intermittent (Bulk)
QoS Profile	GBR (High BW)	Low Latency ($<$ 10ms)	Non-GBR (Background)
Traffic Tool	FFmpeg (RTSP)	iPerf3 (UDP)	wget (HTTP)
Simulated Data Rate	$\approx$ 5-10 Mbps	2Mbps	$\approx$ 20Mbps
PRB Allocation	Dynamic	Hard (10 PRB)	Proportional Fair

For the practical realization of the traffic steering mechanism described in Section II-A, the unavailable commercial URSP implementation has been bypassed in favor of a kernel-level solution. The nDPI deep packet inspection library [14] has been deployed directly on the UE host. This setup mimics an Industrial Gateway proxying legacy sensors, allowing deep inspection without modifying constrained endpoints. As discussed in [15], this module acts as a functional proxy for the 3GPP standard, providing the necessary accuracy to distinguish between industrial protocols and video streams in real-time. It tags packets at the OS level and routes them to the specific network interfaces associated with the active slice PDU sessions. To ensure classification accuracy and a consistent baseline for the architectural evaluation, we enforced a conservative re-classification period of 10 seconds.

D. Control Logic Implementation

To validate architectural latency and closed-loop stability, the Resource Manager implements a deterministic Proportional Controller strategy. While the component is designed as a model-agnostic container capable of hosting complex AI/ML agents, this streamlined heuristic is sufficient to verify the mechanical enforcement of constraints. The logic processes real-time usage telemetry, retrieved from the SDL with minimal overhead, through an Exponential Moving Average (EMA) filter to smooth transient jitter, comparing the result against a target utilization setpoint of 75%. A hysteresis deadband is applied to prevent oscillation, triggering reallocation only when the error magnitude exceeds a significant threshold. Upon determining a necessary adjustment, the manager constructs a configuration payload containing the active status flag, the target slice ID, and the synchronized uplink and downlink PRB quotas (`dl_num_prbs` and `ul_num_prbs`).

This directive is transmitted to the Slice Manager xApp, which acts as the secure gateway to the RAN. Upon reception, the xApp executes a strict validation routine, verifying that the requested Slice IDs map to active contexts and that the aggregate PRB allocation adheres to spectral grid safety limits (e.g., < 275 blocks). Validated constraints are then encapsulated into a custom `rc_ctrl_req_data_t` structure using the `ACTION_ID_SLICE_CONFIG` opcode and asynchronously broadcast via the E2 interface. This ensures that new resource boundaries are applied by the gNB scheduler in the subsequent TTI without inducing blocking latency. Although the architecture supports TTI-level control, a 10 ms cycle has been selected for metric retrieval in the SM to align with Near-RT RIC standards while proving sufficient for the target traffic stability.

E. Methodological Design Criteria for Real-Time Actuation

A primary challenge in O-RAN experimentation is ensuring that the Metrics Plane described in Section II-B does not induce latency in the Control Plane described in Section II-C. To satisfy the strict timing budget of the Near-RT RIC, we have adopted a zero-blocking design philosophy governed by the following two methodological criteria:

1) Elimination of Synchronous Disk I/O: To prevent thread suspension during telemetry export, we have enforced a strict memory-only data path. Standard distributions of the FlexRIC controller utilize a synchronous SQLite backend for metric persistence. However, experimental profiling revealed that the blocking write operations of this relational DB introduced CPU latency spikes incompatible with TTI-level synchronization. To resolve this, we have modified the build configuration to disable the relational backend entirely during production runs.

2) Decoupling of Acquisition and Transmission: To further isolate high-frequency RAN telemetry from latency, the Metrics Exporter xApp employs a Producer-Consumer pattern rooted in dual thread-safe Ring Buffers, as illustrated in Figure 3.

The Producer (Main Thread) is responsible for capturing `E2_INDICATION(MAC_STATS)` messages at the TTI level. Instead of forwarding raw data, it performs temporal aggregation, averaging throughput over a 100-tick window to smooth instantaneous jitter and applying a max-hold strategy for retransmission statistics.

Concurrently, the Consumer (Worker Thread) implements I/O batching, grouping up to 50 JSON commands into a single network transmission. This architecture significantly reduces Round-Trip Time (RTT) overhead while simultaneously executing non-blocking HTTP POST requests to the Prometheus visualization backend.

IV. Evaluation

To assess the performance of the proposed O-RAN slicing framework, the experimental evaluation has been designed to validate the three architectural planes defined in Section II: (1) the functional correctness of the closed-loop algorithm under dynamic congestion scenarios, demonstrating the effective decoupling of the architecture modules, (2) the ability of the software stack to operate within strict real-time constraints (Metrics & Control Planes) and (3) the accuracy and isolation of the device-centric traffic steering (Data Plane).

A. Dynamic Service Assurance and Congestion Recovery

The experimental protocol validates the system's ability to execute runtime reconfiguration without service interruption. As detailed in Table I, Slice 1 (eMBB) serves as the primary validation target, transitioning from standard Proportional Fair contention to dynamic PRB allocation at $t \approx 100$ s, while Slice 2 maintains a static reservation. During the baseline phase ($t = 0$–100 s), system-wide contention resulted in aggregate retransmissions exceeding 200 events. Upon activation, the Slice Manager successfully injected PRB quotas via the E2 interface, which the scheduler immediately enforced without gNB restart or service disruption. As shown in Figure 4, this actuation instantly stabilized Slice 1 throughput and confined residual congestion solely to the Best-Effort partition.

The immediate elimination of contention-induced retransmissions for protected slices, alongside an 85.5% reduction in throughput variance, functionally validates the proposed split-logic architecture. These results confirm that the lock-free shared state mechanism successfully decouples the control plane from the radio scheduler, ensuring that asynchronous E2 directives are translated into rigid TTI-level enforcement regardless of transport jitter.

B. Control Plane Latency

To ensure compliance with Near-RT RIC timing constraints (10ms – 1s) [9], we have evaluated the impact of the xApp on the critical control loop. Based on the timing logs aggregated from the 5 experimental runs of 5 mins (1500 metrics), the measurements reveal a sharp distinction between transport latency and processing overheads.

Measurements reveal a clear distinction between transport and processing latencies. The E2 transport (Network RTT) averaged 0.64 ms with significant jitter (up to 2.58 ms), while the internal xApp logic remained deterministic ($30\,\mu s$), with a highly deterministic range of 0.02 – 0.06 ms. Consistent with the design principles in Section II-C, this transport jitter did not impact the scheduler's stability. The MAC successfully enforced the most recent valid state in the subsequent TTI,

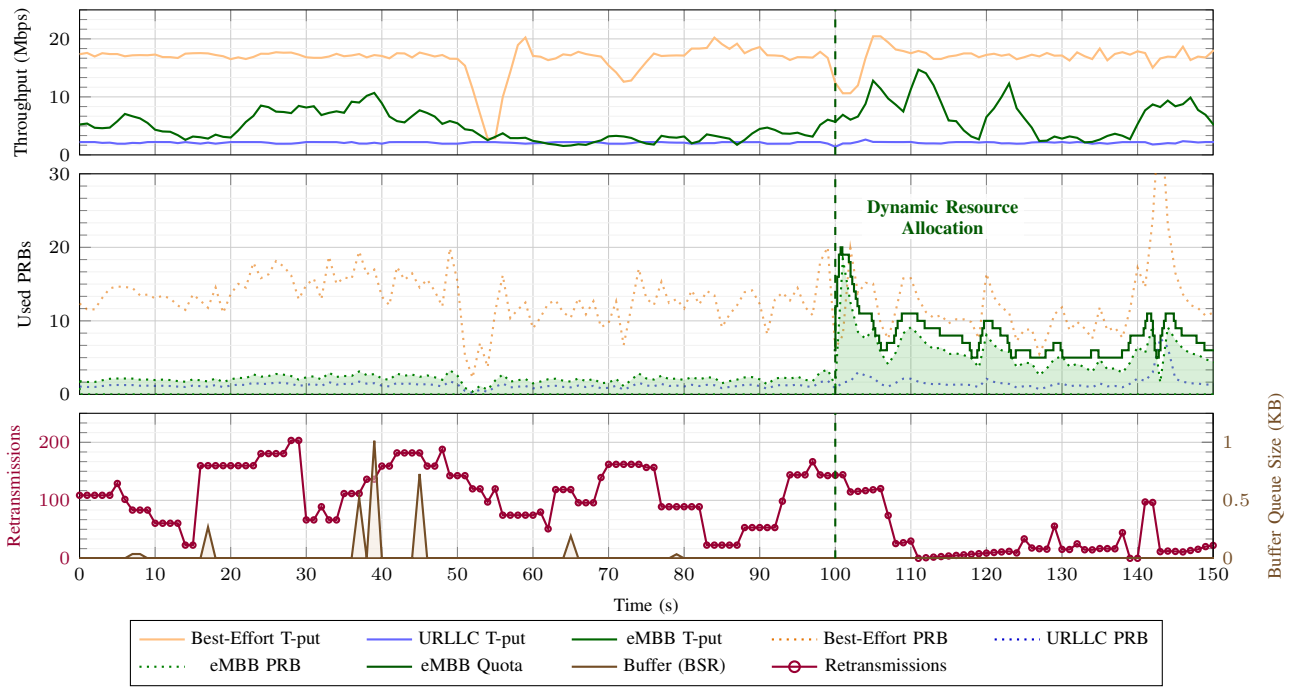

Fig. 4. Performance metrics across three operational phases. Top: Per-slice throughput (Mbps). Middle: PRB allocation with eMBB usage (green fill) and quota limit (red line). Bottom: UE-level retransmissions (purple, left axis) and buffer queue size (KB) (brown, right axis) and .

confirming that the radio execution loop is effectively isolated from E2 transport fluctuations.

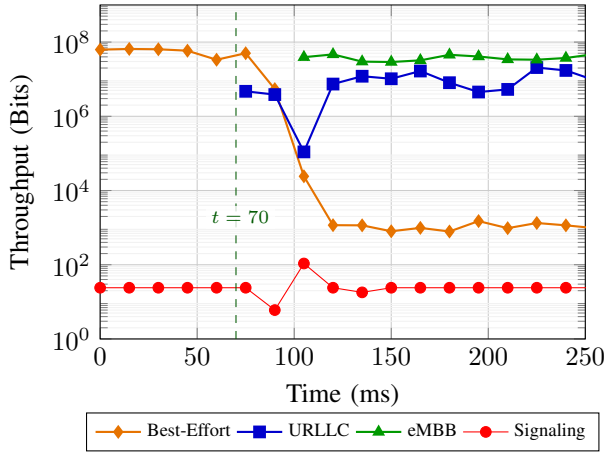

Fig. 5. Logarithmic evolution of slice throughput demonstrating the dynamic activation of the traffic steering mechanism.

C. Scheduler Execution Efficiency

The computational overhead introduced by the custom resource partitioning logic within the MAC scheduler has been profiled under high-load conditions to ensure compliance with strict TTI deadlines. The results demonstrate that the isolation algorithm is computationally negligible. The execution time follows a tightly concentrated distribution with a mode of approximately 250 ns (0.25 μs) and a weighted mean of 367 ns. Even considering the worst-case outliers (99^{th} percentile), the execution time remains below 900 ns. Against the standard 5G TTI budget of 500 μs (for 30 kHz Sub Carrier Spacing (SCS)), this overhead constitutes less than 0.2% of the available processing time. Since complexity is linear $O(N)$ with slices (typically $N \leq 8$) and invariant to the number of active UEs, the execution time remains bounded, ensuring scalability even in dense environments. Consequently, the execution time is deterministically bounded, posing no threat to the scheduler's stability even in saturation scenarios.

D. Metrics Pipeline Performance

The computational overhead and data freshness of the telemetry export pipeline have been profiled to validate the asynchronous design's effectiveness. Redis pipelined writes execute in 0.011 ms (11 μs), adding virtually no delay to the worker thread. In contrast, synchronous Prometheus HTTP POST operations incur 1.78 ms latency, a 160× disparity. This confirms that offloading visualization tasks to dedicated threads is mandatory to preserve the sub-millisecond processing budget of the E2 Agent.

Table II quantifies the impact of the dual-ring optimization on data freshness (ingress-to-buffer latency). The single-ring implementation exhibited tight coupling between pipelines: fast Redis operations were blocked by slow Prometheus HTTP requests, leading both paths to share the same high latency profile (median 1.732 ms). Decoupling the pipelines resolved this bottleneck. Redis latency dropped to 0.046 ms (97.4% improvement), effectively eliminating jitter in the control plane. In contrast, Prometheus latency increased to 3.125 ms, an expected trade-off since the synchronous worker blocks on network I/O, causing metrics to queue in the ring buffer. This latency increase is acceptable, as Prometheus is used solely for visualization and does not affect the real-time control loop, allowing the critical path to remain minimal.

TABLE II. IMPACT OF DUAL-RING OPTIMIZATION ON DATA FRESHNESS (MEDIAN [Q1 – Q3])

Metric	Single (ms)	Dual (ms)	Improv.
Redis Age	$1.732_{[0.03,2.12]}$	$0.046_{[0.04,0.56]}$	**97.4%**
Prom. Age	$1.733_{[0.03,2.12]}$	$3.125_{[1.25,4.62]}$	-80.3%*

*Latency increase due to non-critical buffering

E. Device-Centric Uplink Traffic Steering

Figure 5 validates the dynamic response of the Data Plane. Initially, all unclassified traffic is mapped to the Best-Effort slice. As the traffic classifier identifies the traffic type, the system enforces the steering policy, triggering a handover of flows to their respective QoS-defined slices.

The logarithmic scale reveals two critical behaviors: Effective Isolation, visible at $t = 70$ ms where throughput spikes in URLLC and eMBB while the Best-Effort Slice drops; and (2) Stability, as Signaling traffic (red line) remains unaffected by these transitions.

This confirms that the mechanism decouples connectivity from service assurance. Crucially, since nDPI classification operates asynchronously, the scheduler acts on cached tags, preserving sub-millisecond scheduling latency independent of the 10 second detection window of the traffic classifier.

V. CONCLUSIONS AND FUTURE WORK

This paper presented a split-logic O-RAN slicing framework designed for private 5G NPNs that decouples Near-RT RIC policy adaptation from MAC-layer enforcement via lock-free shared state, enabling dynamic PRB partitioning independent of E2 transport jitter.

Experimental validation on an end-to-end 5G SA testbed confirmed the architectural viability of this approach. Rather than focusing solely on throughput gains, the results demonstrate operational stability, specifically the ability to execute live policy transitions without service interruption and to maintain sub-millisecond actuation latency ($30\,\mu$s xApp processing) regardless of background load. The observed elimination of contention-induced retransmissions serves as a proof-of-concept for the isolation logic, verifying that standard open-source RAN components can be retrofitted to support deterministic industrial SLAs.

While this validation utilized a single-UE topology due to open-source RAN stability constraints, the architectural principles are designed to scale. Future work will evaluate multi-UE scenarios (N>20) on commercial O-RAN platforms, extend to per-UE resource allocation, integrate predictive AI/ML models for proactive allocation, and validate operational deployment considerations, including RIC failover mechanisms and xApp lifecycle management, through field trials with industrial partners.

ACKNOWLEDGMENT

This work has been supported by the Spanish Ministry of Economic Affairs and Digital Transformation under project "Investigación y desarrollo de sistemas avanzados de análisis de datos para la gestión de la red mediante técnicas de machine learning y arquitecturas open-source para la integración en redes privadas 5G+/6G", with number TSI-064200-2022-20 as part of the "UNICO I+D 6G 2022" program, and funded by the European Union through the Recovery and Resilience Facility (RRF). We would like to thank Sergio Morato and Jose Luis Martín-Javato from Gamma Solutions for their contribution to this work.

REFERENCES

[1] A. Perdigão, J. Quevedo, and R. L. Aguiar, "Automating 5g network slice management for industrial applications," *Computer Communications*, vol. 229, p. 107991, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366424003384

[2] N. Ioannou, D. Kokkinis, D. Katsianis, and D. Varoutas, "5g ran slicing for industry 5g verticals: A comparative techno-economic analysis of private network deployment versus network slicing," ser. ITS 33rd European Conference 2025: "Digital innovation and transformation in uncertain times", Edinburgh, UK, 29th June – 1st July 2025. Calgary: International Telecommunications Society (ITS), 2025. [Online]. Available: https://hdl.handle.net/10419/331278

[3] K. Alam, M. A. Habibi, M. Tammen, D. Krummacker, W. Saad, M. D. Renzo, T. Melodia, X. Costa-Pérez, M. Debbah, A. Dutta, and H. D. Schotten, "A comprehensive tutorial and survey of o-ran: Exploring slicing-aware architecture, deployment options, use cases, and challenges," *IEEE Communications Surveys Tutorials*, vol. 28, pp. 1637–1678, 2026.

[4] J. Luis Herrera, S. Montebugnoli, D. Scotece, L. Foschini, and P. Bellavista, "A tutorial on o-ran deployment solutions for 5g: From simulation to emulated and real testbeds," *IEEE Communications Surveys Tutorials*, vol. 28, pp. 1709–1748, 2026.

[5] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Federated deep reinforcement learning for resource allocation in o-ran slicing," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 958–963.

[6] H. Zhang *et al.*, "Resource allocation for network slicing in open RAN: A hierarchical learning approach," *IEEE Transactions on Cognitive Communications and Networking*, Jan 2025, early Access.

[7] R. Schmidt, N. Nikaein, R. Knopp, D. Egelseer *et al.*, "Flexric: an sdk for next-generation sd-rans," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2021, pp. 411–425.

[8] O-RAN Alliance, "O-RAN near-real-time ran intelligent controller e2 service model (E2SM) ran control," O-RAN Alliance, Technical Specification O-RAN.WG3.E2SM-RC-v01.03, 2023.

[9] O.-R. Alliance, "O-ran architecture description," O-RAN Alliance, Technical Specification O-RAN.WG1.O-RAN-Architecture-Description-v07.00, 2022.

[10] S.-P. Yeh, S. Bhattacharya, R. Sharma, and H. Moustafa, "Deep learning for intelligent and automated network slicing in 5g open ran (oran) deployment," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 64–70, 2024.

[11] D. Johnson, D. Maas, and J. Van Der Merwe, "Nexran: Closed-loop ran slicing in powder -a top-to-bottom open-source open-ran use case," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, ser. WiNTECH '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 17–23. [Online]. Available: https://doi.org/10.1145/3477086.3480842

[12] S. Niknam, A. Roy, H. Dhillon, S. Singh, R. Banerji, J. Reed, N. Saxena, and S. Yoon, "Intelligent o-ran for beyond 5g and 6g wireless networks," 12 2022, pp. 215–220.

[13] 3GPP, "Policy and charging control framework for the 5g system (5gs); stage 2," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.503, 2023.

[14] "ndpi, [online]," https://github.com/ntop/nDPI, accessed: 2025-01-19.

[15] Á. Gabilondo, Z. Fernández, Á. Martín, M. Zorrilla, P. Angueira *et al.*, "Dynamic mobile network slicing through vehicular traffic analysis," *IEEE Open Journal of Vehicular Technology*, 2025.