EdgeRIC: Empowering Realtime Intelligent Optimization and Control in NextG Cellular Networks

Woo-Hyun Ko, *Texas A&M University;* Ushasi Ghosh, *University of California San Diego;* Ujwal Dinesha, *Texas A&M University;* Raini Wu, *University of California San Diego;* Srinivas Shakkottai, *Texas A&M University;* Dinesh Bharadia, *University of California San Diego*

https://www.usenix.org/conference/nsdi24/presentation/ko

This paper is included in the Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation.

April 16–18, 2024 • Santa Clara, CA, USA

978-1-939133-39-7

EdgeRIC: Empowering Real-time Intelligent Optimization and Control in NextG Cellular Networks

Woo-Hyun Ko[*1], *Ushasi Ghosh*[*2], *Ujwal Dinesha*[1], *Raini Wu*[2], *Srinivas Shakkottai*[1] *and Dinesh Bharadia*[2]

[1] *Texas A&M University, TX, USA,* [2] *UC San Diego, CA, USA*

{whko, ujwald36, sshakkot}@tamu.edu *{ughosh, rainiwu, dineshb}@ucsd.edu*

Abstract

Radio Access Networks (RAN) are increasingly softwarized and accessible via data-collection and control interfaces. RAN intelligent control (RIC) is an approach to manage these interfaces at different timescales. In this paper, we introduce EdgeRIC, a real-time RIC co-located with the Distributed Unit (DU). It is decoupled from the RAN stack, and operates at the RAN timescale. EdgeRIC serves as the seat of real-time AI-in-the-loop for decision and control. It can access RAN and application-level information to execute AI-optimized and other policies in real-time (sub-millisecond). We demonstrate that EdgeRIC operates as if embedded within the RAN stack. We showcase RT applications called μApps over EdgeRIC that significantly outperforms a cloud-based near real-time RIC ($>$ 15 ms latency) in terms of attained system throughput. Further, our over-the-air experiments with AI-based policies showcase their resilience to channel dynamics. Remarkable, these AI policies outperform model-based strategies by 5% to 25% in both system throughput and end user application-level benchmarks across diverse mobile scenarios.

1 Introduction

As we move into the age of NextG applications, cellular networks need to be versatile and must cater to a wide array of application-specific requirements concerning throughput, latency, and reliability. The "one size fits all" cellular network approach is fading, raising the need to be replaced by an adaptive, application-specific model. Modern applications can often furnish granular details about their operational context to aid such adaptation. A streaming app can reveal buffer status, AR/VR applications share viewing angles, robotic controllers offer positional data, and industrial IoT devices indicate data freshness. These capabilities can enable tailored network responses. For instance, in a video streaming network environment, rather than merely maximizing system throughput, it might be paramount to prioritize users nearing buffer exhaustion to prevent video playback halts. Different apps have varied response time needs; while a video might handle some delay, a VR game demands near-instant reactions.

[*]These authors contributed equally to this work.
The source code for this project is available at https://github.com/ushasigh/EdgeRIC-A-real-time-RIC.git

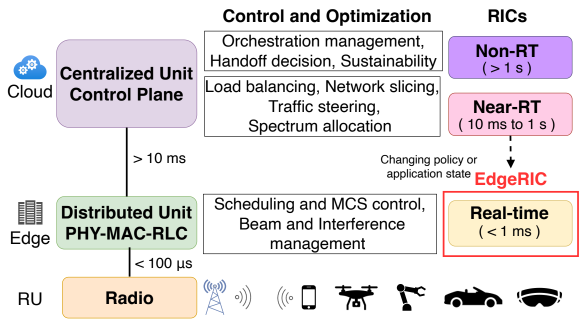

Control and Optimization
RICs
Cloud
Centralized Unit Control Plane
Orchestration management, Handoff decision, Sustainability
Non-RT (> 1 s)
Load balancing, Network slicing, Traffic steering, Spectrum allocation
Near-RT (10 ms to 1 s)
> 10 ms
Changing policy or application state
EdgeRIC
Edge
Distributed Unit PHY-MAC-RLC
Scheduling and MCS control, Beam and Interference management
Real-time (< 1 ms)
< 100 µs
RU
Radio

Figure 1: Timescales of RAN Intelligent Control for O-RAN. We desire real-time control at a latency $<$ 1 ms.

While application awareness has become critical to the decision-making process of network functions in both wired and wireless (cellular) communications, the problem is far more complex with rapidly changing wireless channels for cellular communications. More specifically, the channel evolves at the timescale of milliseconds. Such changes in the channel make the control and optimization of the link to meet application requirements far more challenging.

The need for cellular networks to adapt to different applications, all while keeping up with the rapidly changing wireless channel, is supported by the cellular industry's pivot towards the standardization of open interfaces for RANs, encapsulated by the term O-RAN. By harnessing softwarization and disaggregation at every layer, O-RAN provides the flexibility to operate the RAN stack across diverse distributed computing platforms and offers enhanced monitoring and control through novel interfaces. Alongside this evolution, the RAN Intelligent Control (RIC) concept has gained prominence. RICs, distinct from the time-critical RAN stack, are designed to seamlessly access both the application layer and RAN-level data. This dual access facilitates cross-layer decision-making and control, bolstered by AI/ML enhancements, to serve a plethora of high-demand applications.

A defining attribute of the RIC is the time scales at which it operates, necessitating control decisions and information access at those time scales. Current RIC architectures, visualized predominantly as centralized control microservices, are placed in the cloud, as depicted in Figure 1. These can be classified into: (i) Near Real-time RIC (near-RT RIC): Provides a feedback loop to the RAN stack in a range of 10ms to 1s. (ii) Non Real-time RIC (non-RT RIC): Operates with a feedback loop timescale exceeding 1s.

Unfortunately, both these RICs inspired by SDN controllers, adopt a centralized, cloud-based control approach, minimizing risk to the RAN stack's essential operations within each TTI. This approach prevents potential disruptions, such as PHY-MAC task latency breaches leading to UE detachment. However, the inherent delay in decision-making, exceeding 10ms due to the wireless channel's rapid variability, creates bottlenecks. Such delays result in broad decision-making strategies like resource slicing, where, for example, a streaming app is allocated to a high-throughput RAN slice. Yet, as network demands grow, this coarse strategy struggles to maintain efficiency, given the fast-paced fluctuations in wireless channels that require TTI-scale responsive decisions for optimal performance.

While O-RAN has enabled unprecedented macro network optimizations, a vast reservoir of potential remains untapped in granular, *real-time* control. To that end, we propose a disaggregated real-time RIC platform called EdgeRIC. EdgeRIC is positioned on edge-compute close to the radio head, synchronizes intelligence with the granularity of RAN functions, but is decoupled with the RAN stack, which enables resilient operation of the RAN stack. EdgeRIC's careful design enables it to synchronize with RAN events and provide decision-control, while ensuring functioning of the tight-constrained RAN stack, even if decisions are not received from EdgeRIC. EdgeRIC facilitates control decisions and network telemetry at the TTI timescale (smallest unit of decision-making available at RAN), which is faster than underlying channel variations. The philosophy of EdgeRIC is to revolutionize the algorithmic control of lower-layer RAN functionalities (PHY-MAC-RLC), integrated with application awareness, thereby realizing the true potential of an AI-driven air interface. To provide an overview, Figure 1 showcases the myriad RAN enhancements that can be made feasible through intelligent control across diverse timescales.

Main Contributions

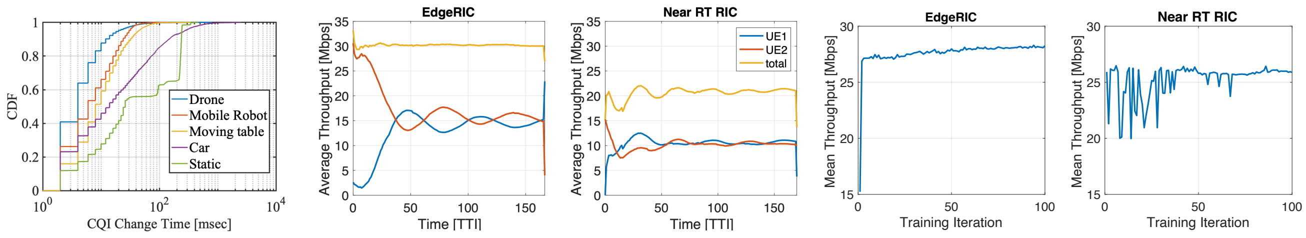
Our contributions are twofold: (i) EdgeRIC: a real-time RIC module, which facilitates real-time RAN telemetry and control. EdgeRIC is driven by the basic observation that real-time control for the cellular stack implies TTI-level sync with the RAN stack—all events happen between TTIs. Our principal contribution is rooted in the architectural and engineering decisions that ensure real-time performance of telemetry and AI-optimized control policies, without ever violating the TTI boundaries. EdgeRIC is strategically situated on edge compute, independent from the RAN stack, and interfaces with it via an O-RAN-like standard. (ii) EdgeRIC emulator: We further introduce the EdgeRIC training module, a cloud-compute-based emulator. This module is instrumental in the design, offline training, and deployment phases of AI-optimized algorithms. These features are enabled by providing a comprehensive full-stack, trace-driven (network and channel) emulation environment that accommodates multiple users and diverse applications.

Finally, we showcase potential benefits of real-time control with an AI-optimized μApp to provide resource allocation decisions at each TTI. We demonstrate (i) throughput increases of 5-10% using RT resource scheduling over near-RT approaches, (ii) up to 15% throughput increases through robust Reinforcement Learning (RL) based RT scheduling, and (iii) up to 30% enhancement in Quality of Experience (QoE) for video streaming via an application-aware RL-based RT scheduling policy over application-agnostic approaches. We also benchmark its performance against a near RT RIC. To the best of our knowledge, no existing RIC platform has demonstrated the benefits of real-time AI-in-the-loop-based RAN control while leveraging cross-layer application information in over-the-air experiments.

2 Motivation for Real-Time RIC

A fundamental value proposition of RIC is that it would enable the RAN to adapt to support heterogeneous applications over a variety of end devices ranging from smartphones, drones, cars, headsets, to sensors. What timescale of monitoring and control would enable application and environment responsive intelligent configuration and control?

Wireless Environment: Our channel measurements at 2.5 GHz across various mobile environments—drones, cars, indoor robots, and human movement—reveal that wireless channels fluctuate within milliseconds, highlighting the need for real-time control in mobile scenarios, illustrated in Figure 2(a). For instance, drone channels showed quality changes within 3-4 ms in over half the cases. Even in low-mobility scenarios, like a robot moving indoors, we observed significant changes within less than 10ms. This demonstrates the importance of designing systems like ours to adapt quickly to the dynamic nature of wireless channels. Additionally, even in stable channel conditions, real-time capabilities enhance RAN functions, such as enabling more aggressive Modulation and Coding Scheme (MCS) selections in the absence of packet drops. As cellular networks evolve towards supporting highly mobile applications with brief channel coherence times, achieving RIC latencies that match the sub-millisecond TTI timescale becomes crucial.



a) Channel qualities change as low as every 3 to 4ms under mobility

b) Realtime scheduling can support a higher system throughput

c) Near RT RL training yields lower throughput with delayed states

Figure 2: NextG networks need real-time intelligence and control.

RAN Control: We next verified the thesis that channel variations require real-time control by implementing a rule-based scheduler that prioritizes users with the largest channel quality index. We ran the experiment over a mix of channels gathered from our mobile experiments and show the throughput obtained in Figure 2(b). The latency experienced by near-real-time RIC with a round-trip latency of 30ms, corresponding to accessing cloud-based compute resources causes the throughput to drop by a third as compared to real-time control using the same scheduler located at edge compute accessed with a round-trip latency of 1ms.

AI Training: Can the RIC can support data collection, training and adaptation of AI/ML policies? Since many RAN management tasks involve feedback control, we are specifically interested in whether reinforcement learning (RL) training and execution can be supported. We conducted experiments on training an RL-based scheduler with a near-RT round-trip delay between state-action-reward of 60ms, versus one that has a round-trip delay of 1ms. As illustrated in Figure 2(c), the correlations between state-action-reward break down at near-RT, leading to lower throughput.

The wireless community is focusing on data-driven approaches for the air interface, enabling intelligent decisions at the TTI timescale. These encompass various tasks such as beam-forming decisions [14] [13] [26], interference management [18] [40], localization [17], channel estimation [21] [23], Modulation and Coding Scheme (MCS) selection [44] [15], power allocation [31], retransmissions and more. As illustrated above, none of these will be possible without real-time RIC co-located with the RAN on edge compute.

3 Related Work

Recent advancements in the development of near real-time (near-RT) RAN Intelligent Controller (RIC) frameworks and xApps have gained significant attention. Scope [4] introduces a containerized method for deploying network elements, supporting real-world emulation, AI/ML data collection, and network control APIs. [5] demonstrates the the integration of deep RL agents for near-RT RIC-based control, built with ColO-RAN [32], an AI/ML framework built upon the Colosseum network emulator [24]. These works focus on RAN resource slicing using xApps and RAN-embedded schedulers assigned to each slice. ORAN E2 [42] presents a software-defined radio testbed featuring an open-source 5G system that interacts with the O-RAN near-RT RIC through standard interfaces, utilizing xApps for RAN slicing. FlexRAN [9] offers a software-defined RAN platform where a master controller communicates with agents embedded in the LTE stack. However, FlexRAN lacks the ability to train or utilize AI-optimized policies while maintaining real-time constraints. FlexRIC [35] addresses this by providing a more modular variant of FlexRAN. It simplifies the 5G near-RT RIC architecture, adhering to the agent-controller approach.

Closer to the TTI-timescale, ChARM [1] presents spectrum selection based on supervised learning over IQ samples collected in real-time, but with control at near-RT RIC. For even finer control, an architecture for integrating distributed applications (dApps) into O-RAN has been proposed, with simulation results on the possible benefits that might be realized via network intelligence at real-time (<10 ms) [7].

Our initial work on EdgeRIC provided an architecture and messaging scheme for enabling real-time RIC (<1 ms), along with a feasibility study and demonstrations of the approach [8, 16]. More recently, an approach entitled Janus [10] also recognizes the significance of real-time intelligence. While both EdgeRIC and Janus aim at real-time measurement and control of the RAN, their architectural choices are fundamentally different. Janus is integrated directly with each Distributed Unit (DU), making it vendor-specific. Thus, Janus requires updates to its hooks and codelets within the RAN software at each site or DU, contingent upon the purchase of Janus by that specific DU. EdgeRIC avoids such tight coupling by following O-RAN compliant messaging interfaces, which decouples EdgeRIC from the RAN stack. EdgeRIC then uses the open source Gym-class interface to connect with an in-memory database and AI/ML algorithms. EdgeRIC's decoupled design allows for robustness by never interfering with time-critical RAN tasks, ease of runtime data gathering, standardized training and runtime updating of AI/ML models, and support across different vendors' equipment via compatibility with universally accepted O-RAN service models. Finally, since EdgeRIC is implemented over the open source

Table 1: Comparison of RIC frameworks

Framework	Connectivity to RAN stack	Monitoring and control	Application awareness	Adaptability to channel fluctuations	Full stack AI training support with real traces	Real World OTA evaluations
FlexRIC [35]	Disaggregated	10ms-1s	✓	×	×	×
ColO-RAN [32]	Disaggregated	10ms-1s	✓	×	×	×
dApps [7]	Disaggregated	6-10ms	×	✓	×	×
Janus [10]	Integrated	<1ms	×	✓	×	✓
EdgeRIC	Disaggregated	<1ms	✓	✓	✓	✓

software srsRAN stack and uses open messaging standards and interfaces, it is available for unfettered experimentation by the research community.

In the applications domain, streaming media has received much attention for AI-optimized control. For instance, AI/ML for choosing video streaming rate selection is considered in [12, 22, 30, 43, 45] from the server's perspective. In contrast, [3] studies optimal policies when the network can be controlled in the context of WiFi-based access. Here, reconfiguration of WiFi flow priorities using AI-optimized policies is shown to improve streaming performance.

In contrast to the above works, EdgeRIC is a simple, lightweight, disaggregated approach towards ensuring that TTI-level synchronized policies can be trained in non-RT and executed in real-time. Specifically, we show that our approach provides the ability to train robust cross-layer optimized policies in non-RT and a guarantee of completing the full feedback loop from sensing, AI-based policy execution and control within each TTI, and are the first to verify our claims while running full stack over-the-air experiments on mobile nodes. Table 1 summarizes comparable frameworks.

4 EdgeRIC Concept Architecture

The EdgeRIC design is primarily driven by the objective of infusing real-time intelligence into network functions and decisions at the network's edge. This approach is particularly important for making informed decisions based on instantaneous channel conditions. By situating decision-making processes closer to the RAN edge, we ensure that the channel metrics utilized are as current and relevant as possible, as opposed to being relayed from a distant cloud infrastructure.

EdgeRIC's architecture extends beyond edge operations to foster a cooperative relationship with cloud systems, enabling smooth data exchange and access to shared databases. It adopts a dual strategy, utilizing real-time edge data alongside cloud analytics to optimize decision-making. This approach aims to merge the promptness of edge processing with the extensive insights of cloud computing, emphasizing the synergy between cloud and edge to enhance cellular network capabilities. This integration effectively combines local responsiveness with global intelligence.

Our architecture design is motivated by two fundamental considerations, namely *(i) Disaggregated Programming Model:* O-RAN is driven by the desire to disaggregate the cellular stack into functional components that can be created by independent developers and instantiated on distributed compute resources. Consequently, EdgeRIC must be modular and decoupled from the RAN components. This will permit simple models of application development, run-time updates, robustness to errors, and bi-directional information sharing with user-defined applications, and *(ii) Real-time RAN Connectivity and Control:* While functionally decoupled from the RAN stack, EdgeRIC must enable messaging with TTI-level sync (< 1ms) with RAN events. This will enable real-time observability of RAN state, such as channel quality or backlog buffers, and decision making and control of the RAN stack each TTI to optimize performance. Consequently, EdgeRIC must be slaved to the TTI clock at the RAN, and messaging and decision making must be lightweight.

In order to realize these goals, EdgeRIC is composed of two modules: (i) the EdgeRIC execution module, which is the seat of μApps for real-time monitoring control of the RAN, as well as information aggregation from user applications, and and (ii) EdgeRIC emulation module, which is used as a full-stack emulator used for training of AI-based and other algorithms prior to instantiating them as μApps in the EdgeRIC execution module. We discuss the architecture and workflow of these modules below.

4.1 Disaggregated EdgeRIC Architecture

The real-time EdgeRIC execution module is illustrated in Figure 3, where we have shown it within the O-RAN architecture. O-RAN consists of a radio unit along with disaggregated microservices that perform the RAN functions. These microservices are divided across edge compute (near the radio) and cloud compute resources, based on the required latency targets. The components of the O-RAN stack are as follows: (i) RF Frontend: Open Radio Unit (O-RU), (ii) Edge Compute: Real-time components at the Open Distributed Unit (O-DU) supporting High-PHY, MAC and radio link control, and (iii) Cloud Compute: Open Centralized Unit (O-CU) with control and management functions. The final element is (iv) Cloud Compute: 5G Core, supporting management, billing

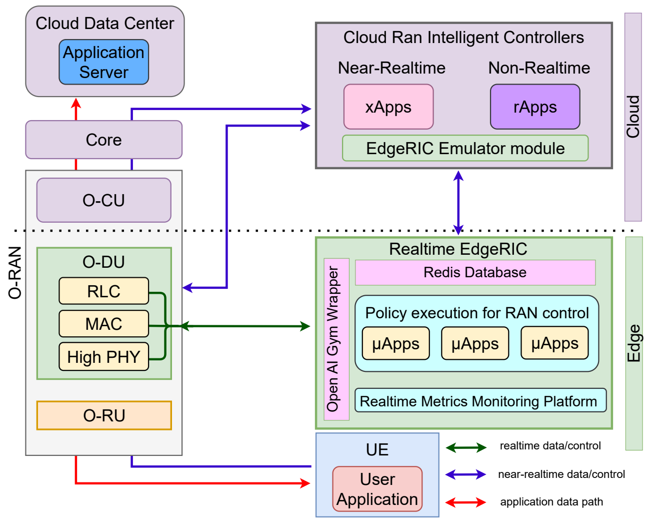


Figure 3: EdgeRIC concept architecture, showing its integration into O-RAN.

and Internet gateway functions.

O-RAN also provides specifications for two cloud-hosted microservices for (i) near-RT RIC, which supports xApps for policy adaptation at near-RT (10ms - 1s), and (ii) non-RT RIC, which supports rApps providing large timescale microservices management and data analytics. The standard provides protocols for the near-RT and the non-RT RIC to communicate with the O-RAN stack and with each other. In particular, the E2 Application Protocol (E2AP) operates over SCTP and provides pub-sub and on-demand messaging between RAN and near-RT RIC at near-RT latency.

EdgeRIC is designed as a microservice for the O-DU, closely integrated with the O-RAN architecture to enhance PHY-MAC level RAN functionalities through real-time. It utilizes μApps for executing real-time policies (with TTI latency), allowing for immediate RAN state adjustments and control. Connection to the O-RAN stack is achieved through a specialized real-time-E2 protocol (RT-E2), aligned with the TTI clock and leveraging IPC for microservice communication, ensuring a latency around 100 μs to meet stringent real-time requirements.

EdgeRIC operates in real-time, running on separate CPU cores at the edge compute cluster to ensure low latency without interfering with the O-RAN PHY-MAC microservices. It allows μApps to dynamically use RAN and application data for decision-making. Supporting integration with protocols like OpenFlow and ROS, EdgeRIC enables cross-layer policies for PHY-MAC control, enhancing system performance without disruption.

4.2 EdgeRIC Functional Components

The EdgeRIC real-time execution module, depicted in Figure 3, features interfaces for real-time communication with the RAN stack and near-real-time interaction with cloud modules. These interfaces connect to an Open AI Gym Wrapper, which abstracts them into a Gym-compatible environment, allowing components to interact with the RAN using Gym methods. This design enables μApp developers to employ either custom or AI/ML-based strategies, benefiting from the compatibility with standardized reinforcement learning frameworks. Consequently, μApps can seamlessly integrate these standardized codeblocks, facilitating efficient development and deployment within the EdgeRIC ecosystem.

The EdgeRIC execution module incorporates an in-memory Redis database for managing real-time RAN metrics and application data. This data supports μApps directly or through cloud processing, enabling cloud-hosted xApps to refine policies or ML models based on the data. These enhancements are fed back to the Redis database for μApps integration, allowing them to start with basic policies and improve them over time with cloud-derived insights, optimizing their performance dynamically.

4.3 EdgeRIC Emulator Module

The EdgeRIC execution module directly enables support for optimization based approaches to modulation, coding and queuing that are designed around well studied, substantiated, and tractable models. For instance, we can immediately instantiate approaches such as proportionally fair [37] or max-weight [41] scheduling across UEs on a per TTI basis with execution in RT, as if embedded within the RAN stack.

Our architecture also supports AI/ML approaches such as Reinforcement Learning (RL), a branch of ML that is explicitly tailored towards learning feedback-control policies. Training such policies is often hard in a real-world system, where user satisfaction is paramount at all times. Hence, we endow EdgeRIC with a full-stack emulator module for training, which can support user applications over trace-based or synthetic channels. The RL workflow is well aligned with the modality of a emulator based non-RT training of a base policy using data collected offline or via an emulator. Such polices can then undergo near-RT adaptation to the current environment, culminating in RT policy execution. Simultaneously, data is gathered at the edge, which is shared with the non and near-RT RIC for accurate training and adaptation. This three-timescale workflow is illustrated in Figure 4.

5 EdgeRIC Implementation

We now describe the implementation of EdgeRIC to satisfy the goals of our concept architecture. The experimental results that we present in this section were collected on two servers: Intel Xeon Gold 5218R CPU @ 2.10GHz, 20 cores and Intel i9 CPU @ 2.4GHz, 12 cores, without using GPUs. We chose the open source Software Radio Systems srsRAN stack [39] as the experimental RAN system for EdgeRIC integration due to its simple, modular codebase, its stability and compatibility with various core networks, 4G and 5G versions, and

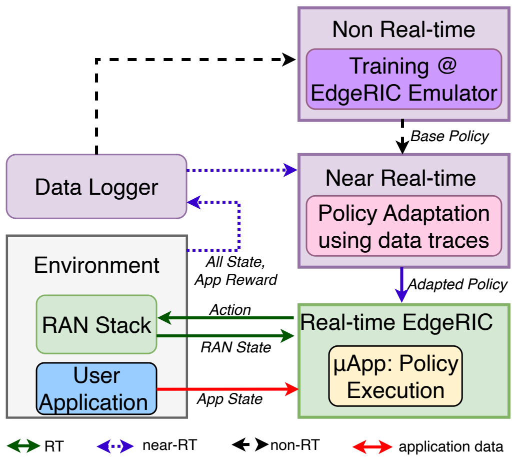


Figure 4: Non-RT policy training on EdgeRIC Emulator, Near-RT policy adaptation and RT policy execution.

the availability of the srsUE codebase. srsRAN runs on the general-purpose Ubuntu OS, which does not provide real-time guarantees.

5.1 EdgeRIC Execution Module

5.1.1 Real-time Connectivity to RAN and messaging

O-RAN specifications provide the E2 interface between the near-RT RIC and the O-DU or O-CU. Specifically, the E2 application protocol (E2AP) operates over SCTP and provides near-RT services for RAN monitoring and control. E2 does not support a real-time connectivity service, i.e., it is not synchronized with the TTI clock at the RAN. Hence, we extend the specification to create a real-time, RAN-synchronized variant that we call RT-E2. RT-E2 supports the following to connect EdgeRIC with the PHY-MAC stack at the O-DU.

RT-E2 TTI-Sync: Our system is synchronized to the TTI-level clock tick from the RAN stack, ensuring that EdgeRIC and the RAN maintain TTI-by-TTI alignment for accurate real-time control actions and reward feedback. The RAN stack uses a TTI counter, referred to as *RANtime*, included in all RT-E2 messages to EdgeRIC. Correspondingly, RT-E2 messages from EdgeRIC to RAN specify the TTI for policy application, ideally set to $RANtime + 1$, to ensure actions match the current RAN state. To prevent asynchrony caused by EdgeRIC's compute time exceeding one TTI, termed "Lazy RIC," EdgeRIC's RAN subscriber only retains the most recent RAN message, tagging policy messages with the latest $RANtime + 1$. The RAN disregards any EdgeRIC message not matching the current *RANtime*. Additionally, the RAN is equipped with a default mechanism to manage any potential Lazy RIC scenarios, ensuring stability despite possible invalid inputs from EdgeRIC.

RT-E2 Report: This is a periodic pub-sub procedure under which a module at the O-DU, such as radio link control may publish information at a given rate. Our default periodicity is one TTI, i.e., information may be generated in real-time. μApps at EdgeRIC may subscribe to the RT-E2 Report service and utilize it for inference and control. Subscription may be blocking in that the μApp will proceed only when new information is available from the RAN.

RT-E2 Policy: This is an event-driven pub-sub procedure under which a μApp at EdgeRIC may publish information to one of the O-DU modules such as UE priorities for resource allocation at the MAC layer. This information is used directly for real-time control at the O-DU. Subscription is non-blocking in that the O-DU subscriber will move on if no new information is available on this procedure, without breaking the tight TTI deadlines required by PHY-MAC.

RT-E2 API Support: RT-E2, synchronized with the RAN, supports messaging over TCP/UDP/SCTP or IPC, adapting to EdgeRIC's hosting environment. It necessitates both blocking and non-blocking pub-sub capabilities and must manage diverse messages, including sync, state, action, reward fields, and UE identities. Opting against modifying the O-RAN E2's limited SCTP-based messaging, we use ZMQ for RT-E2 due to its low latency, minimal overhead, versatile pub-sub modes, and compatibility with IPC or TCP, accommodating our required message formats.

TTI-Level Events: The sequence of events, occurring every TTI is shown in Figure 5, where we see (i) state measurement and transmission from RAN, (ii) reception, processing and response at EdgeRIC, and (iii) final resource allocation at RAN. Note that srsRAN receives each EdgeRIC message well before the TTI boundary, but only reads it in a non-blocking manner at the beginning of each TTI.

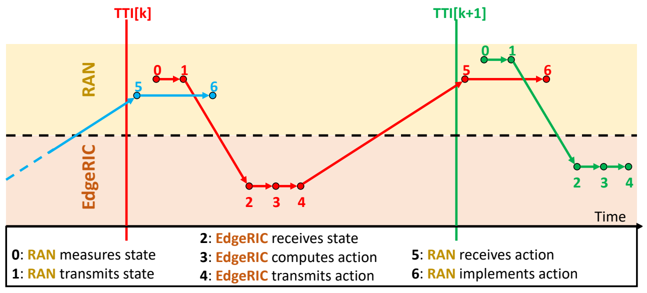


Figure 5: TTI-level events for EdgeRIC to RAN loop.

Real-time operation evaluations: We aim to demonstrate the effectiveness of our synchronization and messaging techniques between the RAN and RT-RIC, crucial for feedback control and RL training success. Through tests with the srsRAN stack and RT-RIC on a server, operating at 10 MHz downlink load and achieving about 37.5 Mbps throughput, we found our system maintains a median round trip latency of just 100 μs. This performance is satisfactory for managing

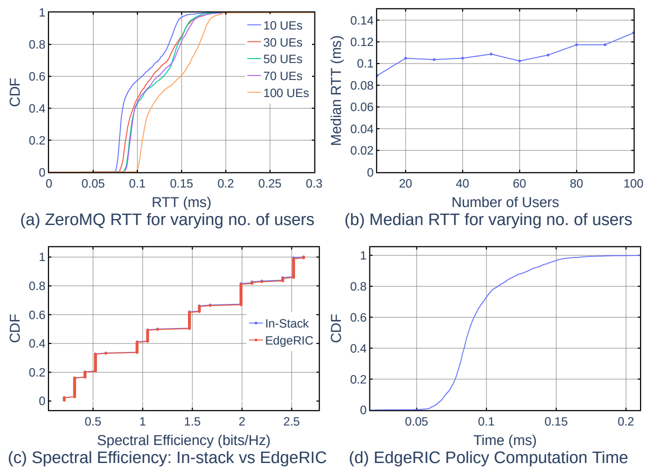


Figure 6: EdgeRIC Feedback latency, spectral efficiency and AI-policy execution times.

information dissemination and control commands for approximately a hundred UEs, showcasing ZeroMQ's suitability for real-time operations as illustrated in Figure 6(a) and (b).

In our study on MAC layer resource block (RB) allocation, the RAN stack communicates UE state information, including RNTIs, CQIs, buffer states, and previous downlink bitrates, to the RT-RIC, which then decides on downlink RB allocations per UE. Figure 6(c) shows that the spectral efficiency achieved by implementing policies through EdgeRIC is on par with integrating them directly into the RAN stack, indicating that our modular approach maintains efficiency while ensuring RAN stack stability. This underscores the effectiveness of our decoupled architecture in balancing policy enforcement with architectural integrity.

5.1.2 Cross-Layer Connectivity and Logging

Our choice of ZMQ for inter-process communication between RAN and EdgeRIC is also extendable to cross-layer application awareness, shown in Figure 4. Since ZMQ can operate over TCP or UDP on an IP network, applications can simply use ZMQ to publish their state information to EdgeRIC. Apart from being lightweight and having APIs in most programming languages, ZMQ also permits client authentication and encryption via CurveZMQ [6] for security.

We also enable EdgeRIC with an in-memory Redis database for data logging and sharing, shown in Figure 4. Redis is a fast, lightweight, key-value store, in which we log data digests, as well as trained models for sharing across the elements of EdgeRIC. An added advantage of Redis is that we can save all traces to drive at experiment conclusion, which allows for post processing and performance analysis.

5.1.3 Integration with OpenAIGym

OpenAIGym is an open source python library that provides a framework for developing an interface to interact with and query the environment by any given algorithm. While it is typically used for developing and comparing RL algorithms, it can be used as a standard approach for realtime policy execution, regardless of whether the policy in question is based on RL. This openness motivates us to develop an OpenAIGym interface connecting the RAN stack, EdgeRIC and the control algorithms in the form of μApps that it hosts. Our OpenAIGym interface allows for swift policy development and freedom of execution of algorithms as desired. Figure 6(d) shows the time taken by EdgeRIC to execute a forward pass of a trained policy network using only CPU, while running a fully loaded RAN. The mean value is less than 100 μs, which implies that TTI-scale execution is straightforward.

5.2 EdgeRIC Emulator Module

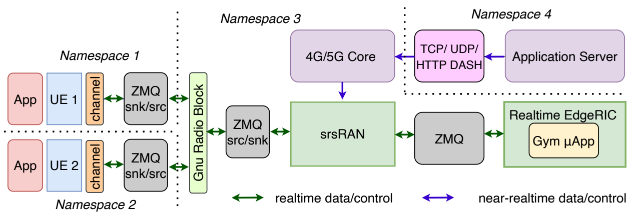


Figure 7: EdgeRIC emulation environment

To bridge the "sim-to-real" gap in RL-optimized control, we've developed the EdgeRIC training module, which forgoes the need for a complex Python simulator for simulating RAN, RIC, and application dynamics. This module leverages actual RAN and RIC codebases with virtual radios and channel simulation techniques, using ZeroMQ for virtual radio interfaces in srsRAN, enabling accurate real-world emulation. It routes complex-valued samples, typical for software-defined radios, through ZeroMQ sockets, incorporating simulated channel effects. A GNU Radio flowgraph with ZMQ Source and Sink blocks allows distribution of these samples to multiple srsUEs, simulating real user equipment. Furthermore, separate IP namespaces for UEs and application servers facilitate running real-world TCP or UDP based applications end-to-end, closely mimicking actual deployment scenarios as shown in Figure 7. This setup ensures minimal sim-to-real disparity, making the EdgeRIC emulator highly effective for real-world applicable policy training and testing.

6 Case Study: An RL based scheduling μApp

In this paper, our illustrative use-case is on realtime downlink resource block allocation (scheduling), which utilizes (i) channel quality information (wireless state), (ii) downlink backlog buffers (RAN state) and (iii) media buffer length for video streaming (application state). We develop an RL-trained realtime scheduling application operating on EdgeRIC, which we refer to as a μApp. We study performance, both from the perspective of throughout maximization under a variety of channel conditions, as well as stall minimization in a video streaming application case. We present the results of real-world over-the-air experiments to validate our approach. In this section, we establish the performance gains achievable with real time RAN control on simulated channel traces.

We use a downlink resource block (RB) scheduling μApp to illustrate an EdgeRIC application, since (i) scheduling has to be done each TTI in realtime, (ii) it requires channel quality information from the wireless link, downlink backlog buffer on a per-UE basis from the RAN, and can utilize application-level information such as the media buffer length in a video streaming for prioritization, and (iii) being such a fundamental problem, has a variety of baseline approaches to compare against RL-based scheduling algorithms. We now describe the design and training of a scheduling μApp.

Weight Based Abstraction of Control: Optimal queueing and wireless resource management often employ structures like threshold [11, 25], index [29, 33, 34], and linear policies [2, 19] for their simplicity and learnability, with some showing properties like monotonicity or concavity [2]. Systems such as [27] develop weights to prioritize flows, ensuring max-min fairness among them. All these structured policies can effectively be represented by assigning relative priorities to the different connected UEs. For example, the so-called Whittle index is a scalar parameter corresponding to the value of resources allocated to a given UE, which can quickly be learned independently of other UEs [28]. Resource allocation may also be done with a fairness metric in mind, such as proportional fairness, where RBs are assigned to a UE based on the ratio of its current as compared to its average channel quality, or max-min fairness [38]. Motivated by these ideas, our general approach for downlink RB allocation is for EdgeRIC to provide values w_i for each connected UE i over realtime information exchange at each TTI. The 5G MAC will then allocate an number of RBs in a manner proportional to w_i over the next TTI. Such an abstraction provides simplicity of actions for the resource allocation policy, while maintaining its ability to attain near-optimal allocations in realtime.

6.1 Training RL on emulator

We utilize the model-free RL algorithm, Proximal Policy Optimization (PPO) [36], for training an agent on optimal resource allocation due to its straightforward implementation and efficiency. Training involves collecting 5,000 samples per iteration, updating the agent's policy neural network via backpropagation, and then using the updated agent to collect another 5,000 samples to evaluate performance and track progress. Each sample corresponds to a transmission time interval (TTI) and includes the environment's current state, the agent's action, and the resulting reward and next state. The RL policies, trainable on the emulator, can focus on RAN-specific scenarios or incorporate application-level data for broader optimizations. We specifically explore downlink throughput enhancement (Section 7.3) and video streaming stall reduction (Section 7.4) as two key use cases.

Table 2: RL Specifications: Throughput Maximization

State $(s[t])$	$B_i[t], CQI_i[t]\ \forall i$
Action $(a[t])$	$w_i[t]\ \forall i$
Reward $(r[t])$	*total throughput*

For the throughput-maximization goal, we utilize RAN-level CQI and UE backlog buffer lengths as state information, with action being the allocation weights for UEs and the reward as total throughput. Training typically reaches reward saturation after 100 iterations, equivalent to 500,000 TTI samples. The RL setup, detailed in Table 2, includes CQI ($CQI_i[t]$) and backlog data ($B_i[t]$) per UE i and the allocation weight for UE i ($w_i[t]$) as actions. Factoring in data collection/transfer to the RL agent and time for actor-critic policy updates, total training completes in about ten minutes.

6.2 Evaluations on emulator

We conducted emulations using synthetic channel traces to assess the potential gains achievable by a real-time agent for policy computation and control. Two metrics we use throughout our study is the downlink system throughput and the downlink backlog buffer lengths at the RAN. We desire to evaluate two basic questions, *(i) How much does performance improve with realtime control as opposed to near real time control* and *(ii) How does RL-based control policy perform compared to basic algorithms?*

We consider three basic algorithms for weight-based resource allocation. In all the below approaches, each UE is assigned a weight $w_i[t]$ at TTI t. The weights are then normalized over all UEs as $\tilde{w}_i[t] = w_i[t]/\sum_j w_j[t]$. The RAN receives the normalized weights $\tilde{w}_i[t]$ from the RIC, and performs an allocation of resource block groups (RBGs) in proportion to the weights, i.e., $R_i[t] = \tilde{w}_i[t]R_{total}[t]$, where $R_i[t]$ is the assignment to UE i, and $R_{total}[t]$ is the number of RBGs available in TTI t. While some approaches call for an absolute prioritization of UEs that have a maximum weight [41], we find in practice that a proportional division based on weight leads to better overall performances.

CQI-Fair Allocation: Here, the weight of UE is equal to its realized CQI. Hence, $w_i[t] = CQI_i[t]$, where $CQI_i[t]$ is the realized CQI of UE i at time t. This approach effectively tries to obtain a large total throughput by prioritizing these UEs that have a large CQI in the current timeslot.

Proportionally-Fair Allocation: The allocation weight for each UE is determined by the ratio of its current CQI to its average CQI, aiming to prioritize UEs with better-than-average channel conditions. The average CQI for UE i, represented as $AvgCQI_i[t]$, is computed using an exponentially weighted moving average up to time t. Therefore, the weight $w_i[t]$ is calculated as $CQI_i[t]/AvgCQI_i[t]$.

Max-weight Allocation: Here, the weight of a UE is the product of its current CQI and the backlogged bytes in the downlink queue corresponding to that UE. The max-weight policy is known to be throughput optimal [41], in that it can achieve the capacity region of the system. Thus, we have, $w_i[t] = CQI_i[t]B_i[t]$, where $B_i[t]$ is the number of backlogged bytes in the downlink queue of UE i.

Our first question, the performance comparison between downlink RB allocation algorithms as μApps on EdgeRIC versus as xApps on a cloud-based RIC focuses on the impact of latency. μApps on EdgeRIC benefit from low round-trip latencies of mere tens of microseconds for state information reception and action generation from the RAN. In contrast, xApps in the cloud suffer from significant forward and reverse network latencies, leading to round-trip times in the tens of milliseconds. To illustrate cloud latency effects, we simulate appropriate delays within EdgeRIC.

Our experiment with synthetic channel traces shows that using the CQI-Fair allocation algorithm as a μApp on EdgeRIC, compared to a cloud-based RIC with 30ms latency, results in a 50% throughput increase with stable backlog buffers (Figure 2(c)). This is further detailed in Table 3. The throughput improvement is also demonstrated in a 4-user scenario, Figure 14 and Table 7 in appendix, thus underscoring the benefits of real-time control in enhancing performance metrics.

Table 3: Load: 35Mbps, Channel: 2 UE synthetic channel

		EdgeRIC	15ms	30ms
Max CQI	Avg. Thrpt.	**32.6**	24.2	18.0
	BL[MB]	0.61	0.64	**0.57**
Prop. Fair.	Avg. Thrpt.	**30.7**	25.7	21.9
	BL[MB]	**0.65**	0.67	0.68
Max Weight	Avg. Thrpt.	**30.0**	23.3	20.9
	BL[MB]	**0.60**	0.62	0.65

To answer our second question, we demonstrate the training and evaluation of an RL algorithm. Scenario 1 is based on a 2UE environment. both connected users have uniform variation in CQI values over time, ranging from 1 to 15. In scenario 2, one user experiences good channel conditions (CQI values between 8 and 15), while the other user experiences poor channel conditions (CQI values between 1 and 7). In Scenario 3, the CQI values are randomly generated. Figure 8 shows training and evaluation on Scenario 2 and Table 8 (appendix) summarizes the performance of RL algorithms on synthetic channel traces.

We show the CDF of end-to-end latency of the entire event chain from RAN to EdgeRIC and back (including policy execution via a forward pass on the policy network), culminating in resource allocation at each TTI in Figure 9. We observe that the end-to-end latency is always less than 1 ms, i.e., the RT-E2 procedures (Section 5.1.1) successfully meet the target of event completion within each TTI.

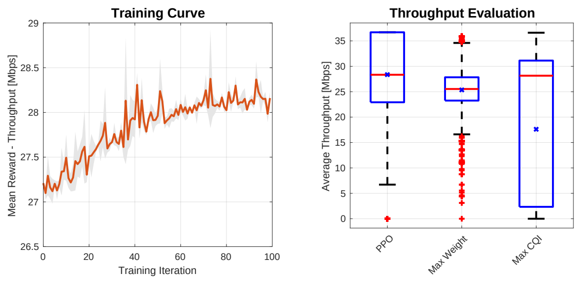


Figure 8: RL training and evaluation on emulator

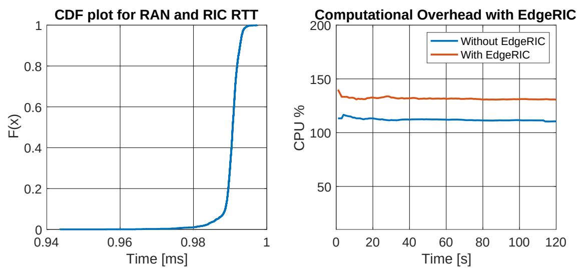


Figure 9: CDF of End-to-end RTT between RAN and EdgeRIC, showing that TTI timings are always met.

Finally, we measure the computational overhead of running EdgeRIC, as compared to running the vanilla srsRAN stack under a full traffic load. We see that the difference in CPU utilization is only about 20%, which means that EdgeRIC is fairly lightweight and does not need exececssive additional compute resources. Hence, co-locating EdgeRIC at the O-DU level seems quite feasible.

6.3 Scalability Study

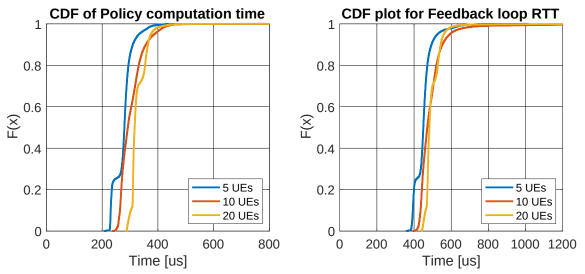
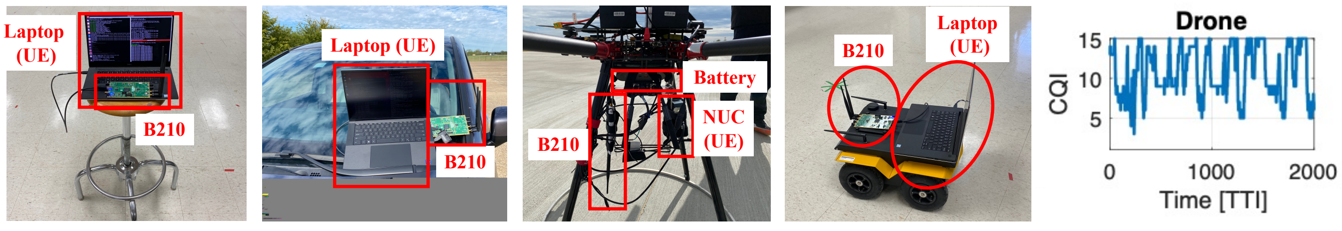


Figure 10: EdgeRIC is able to handle a large number of UEs

We show that our system is stable and is feasible to operate as we scale the number of UEs. For the scheduling μapp considered in this case study, we introduce additional, simulated UEs (which have the same states and application behavior as the real UEs) into the system in order to capture the system performance under an increased number of users. Figure 10 shows that the feedback loop RTT remains well below 1ms and the ML model inference time is under 400μs.



(a) Turntable (b) Car (c) Drone (d) Mobile Robot (e) CQI-evolution

Figure 11: Experimental setups for collecting CQI data of various mobility and evolution of their CQI traces

7 EdgeRIC Evaluation

In this section, we evaluate EdgeRIC by addressing three fundamental questions. These questions are: *(i) Does real-time RAN control through μApps on EdgeRIC outperform near-real-time control through a cloud-based RIC? (ii) Is it feasible to implement real-world AI optimization (RL training and feedback) for resource allocation over EdgeRIC* ? and *(iii) Does application state feedback to EdgeRIC provide significant improvement to the end-user Quality of Experience (QoE)*? All results presented in this section are based on real world channel traces and over the air experiments.

7.1 Experimental Setup

EdgeRIC extends the srsRAN codebase, supporting both software-defined radios and commercial UEs. Built on srsRAN version 21.10, our setup includes USRP B210 SDRs, an edge DU, and an embedded 20ms delay in the CU stack for realism. The srsRAN station features channel trace logging and operates in the 2.5 GHz EBS band under an experimental FCC license to prevent interference with local networks. Experiments utilized a 10 MHz bandwidth, involving a stationary base station and mobile users.

7.1.1 User Devices and Channel Traces

We use a static base-station with different UE types to collect channel traces as shown in Fig. 11. We considered various mobility models, both within the laboratory and in outdoor environments. We extract the channel quality indicator (CQI) values sampled at each transmission time interval (TTI) for each user at the srsRAN for runs of approximately 8 minutes each. Next, we summarize the UEs considered in this study.

TurnTable UE: Figure 11(a) shows the setup of a UE on a movable turntable, similar to a user sitting on a chair and rotating. The base station equipped with a B210 is on a static table, while a UE node with a B210 is mounted on the turntable. The UE node is moved away from or towards the base station at about 1 m/s and rotated to produce significant variations in its CQI values. The distance between the UE node and the base station ranges from 0.5 meter to 4 meter.

Car UE: Figure 11(b) shows a UE setup in a car, using a USRP B210 and laptop powered by the car. CQI data was collected while driving along three paths: 1) a 6-meter-radius circle with a maximum speed of 4.5 m/s, 2) a 30 meter x 3 meter rectangle with a maximum speed of 5.4 m/s, and 3) a 50-meter straight line with a maximum speed of 9 m/s.

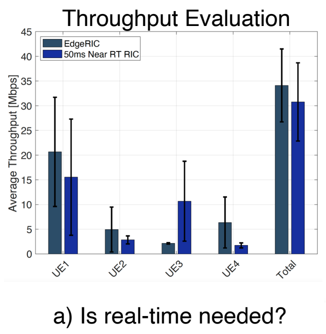
Drone UE: We used a drone experiment to demonstrate performance with faster channel variations in 3D space. Figure 11(c) shows a B210 and a NUC (a small and portable computer) mounted on a Big-Hexy hexa-copter drone, while the base station, equipped with another B210, was placed on the ground. The drone was flown over the base station along two paths: 1) a 10-meter straight line at a height of 15 meters and with a speed of 2.2 m/s, and 2) a 15-meter straight line at a height of 20 meters and with a speed of 3 m/s.

Indoor robotic UE: Mobile robots were used to for indoor mobility experiments. In Figure 11(d), a B210 and laptop were mounted on a Jackal robot, while the base station with an X310 was placed on a table. The robot moved along a 1.6 m x 1.6 m square path and rotated at each corner, causing CQI values to drop due to signal blockage. The working area was limited to 4 m from the base station, similar to robot control or industrial IoT with Private 5G.

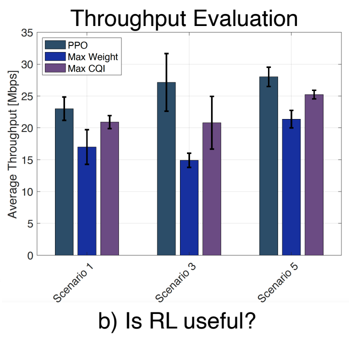
We show the drone trace in Figure 11(e), and the overall CDF was earlier shown in Figure 2(a). We see that several scenarios yield CQI changes in the sub 10 ms range, with the drone trace showing this effect for almost 90% of samples.

7.1.2 Evaluation Scenarios

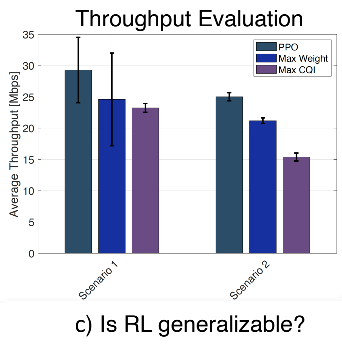
To generate realistic scenarios, we collected channel traces from various environments and utilized them in our emulation setup. This setup includes the same core, radio access network (RAN), and user equipment (UE) modules as our over-the-air experiments that generated the CQI traces. By replaying the CQI traces with a desired number of UEs, algorithmic methods can be compared while maintaining the same end-to-end applications and channel conditions. For end-to-end over-the-air experiments, we used a X310 as a base station and two B210s as UEs. One of the UEs had lower channel quality than the other by placing them at different distances from the base station. Table 4 summarizes the scenarios considered.



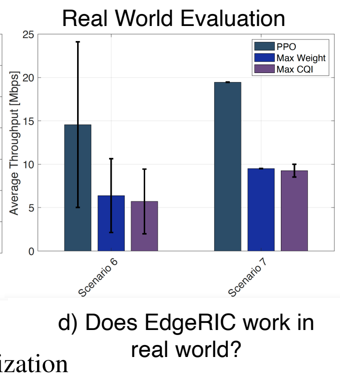
a) Is real-time needed?



b) Is RL useful?



c) Is RL generalizable?



d) Does EdgeRIC work in real world?

Figure 12: Snapshot of EdgeRIC performance for throughput maximization

Table 4: Summary of all scenarios

Scenario	Channel Description
Channel Traces from Experiments	
Scenario 1	2 Drone UEs
Scenario 2	2 Turntable UEs
Scenario 3	2 Car UEs and 2 Drone UEs
Scenario 4	2 Car UEs and 2 Indoor Robotic UEs
Scenario 5	2 Random Walk UEs and 2 Turntable UEs
Complete Over-the-Air Experiments	
Scenario 6	2 UEs on indoor mobile robots
Scenario 7	2 UEs on indoor stationary robots

7.2 Micro-benchmarks: Edge vs. Cloud

We use iPerf for measuring throughput and generate microbenchmarks. We test our scenarios on different UEs, each with varying traffic loads, and report the throughput and backlogs observed with different scheduling algorithms presented in Section 6.2. The validity of our results presented in Section 5.4 is confirmed using realistic channel traces (4 Turntable UEs), shown in Table 5 and Figure 12(a). This provides evidence in support of the hypothesis that real-time control can significantly improve the system throughput.

Table 5: Load: 35Mbps, Channel Trace: 4 Turntable UEs

		EdgeRIC	50ms	100ms
Max CQI	Avg. Thrpt.	**33.4**	21.2	29.5
	BL[MB]	1.34	**0.84**	1.12
Prop. Fair.	Avg. Thrpt.	**28.6**	26.6	23.5
	BL[MB]	1.20	1.29	**0.93**
Max Weight	Avg. Thrpt.	**33.2**	28.8	31.0
	BL[MB]	1.14	1.30	**1.12**

7.3 Impact of RL on Micro-benchmarks

We present compelling evidence for the feasibility, impact, and robustness of RL in executing real-time control policies. We begin by addressing the question of RL training in an emulation environment, which is answered by the set of curves in Figure 16 (appendix). These curves demonstrate that RL training with realistic channel traces is highly efficient, with an agent typically converging in just 20 to 40 iterations. The potential of RL is evident from a series of figures that answer important questions about its application.

Figure 12(b) shows us that RL can achieve higher throughput than traditional algorithms, bringing us to answer: is RL truly useful? This is further summarized in Table 6, which displays the total system throughput and the mean of the total backlog buffer for various scenarios, offering a snapshot of the RL performance in real-time. The values displayed in the table represent the throughput in Mbps (left) and the backlog buffer in MBytes (right).

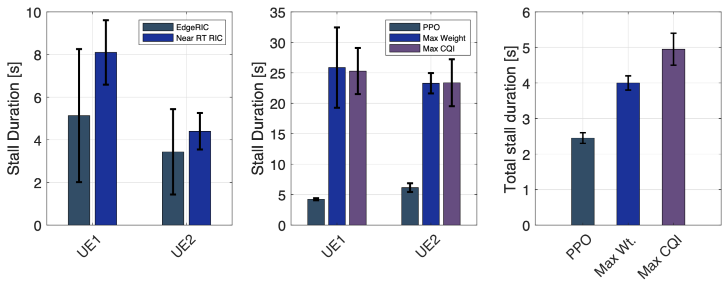
Reinforcement learning (RL) schemes can surpass traditional algorithms under specific conditions, such as varying UE application loads where the max CQI strategy falls short. Our RL algorithm, particularly PPO, adapts to mirror the max-weight algorithm yet excels in scenarios like round-robin allocations, serving one UE per TTI. This approach contrasts with max-weight's fractional resource distribution across all UEs, which, while based on a weight-control scheme, may not always yield optimal results in environments with diverse load demands.

Table 6: Throughput and Backlog Buffer Evaluation

	PPO	Max Weight	Max CQI
Realistic Channel Traces			
Scenario 1	**29.1/0.38**	26.1/0.53	14.9/0.39
Scenario 2	30.5/**0.38**	**31.9**/0.43	14.42/0.39
Scenario 3	**25.3**/1.5	22.9/1.3	18.67/**0.97**
Scenario 4	**25.9**/1.5	23.9/1.21	20.3/**1.05**
Scenario 5	**28.5/0.96**	26.3/1.46	23.3/1.01
Over the Air Experiments			
Scenario 6	**14.6/0.19**	6.4/0.45	5.7/0.44
Scenario 7	**19.33/0.05**	10.71/0.34	9.06/0.35

Table 7: RL specifications: Video Streaming

State ($s[t]$)	$B_i[t], CQI_i[t], MB_i[t]\ \forall i$
Action ($a[t]$)	$w_i[t]\ \forall i$
Reward ($\sum_i r_i[t]$)	$r_i[t] = \begin{cases} -20, & \text{if } MB_i[t] < 2\ sec \\ +2, & \text{otherwise} \end{cases} \forall i$



a) Is real-time needed? **b) Is RL useful?** **c) Does EdgeRIC work in real world?**

Figure 13: Snapshot of EdgeRIC performance for video streaming application

Figure 12(c) illustrates the transferability of RL models, showing that a model trained in one scenario can adapt to others with similar user numbers, highlighting its generalizability. Specifically, a model from a random walk scenario was applied to both a 2 Drone UEs setup (Scenario 1) and a combination of 2 Car UEs and 2 Turntable UEs (Scenario 2). The crucial question of RL's real-world efficacy in realtime wireless control is affirmatively addressed in Figure 12(d). Real-world tests with EdgeRIC revealed that traditional algorithms struggled with channel variability, impacting performance. In contrast, the RL policy maintained system stability and achieved consistent throughput, demonstrating RL's potential for performance gains in realtime scenarios.

7.4 Cross-Layer Optimization: Case Study

In this case study on enhancing video streaming performance through cross-layer optimization, we set up a heterogeneous environment with four users: two streaming high-quality video from an HTTP server and two generating background traffic with iPerf. We use the GPAC [20] library for DASH-style video segments and define a stall event when the media buffer drops below 2 seconds. The media buffer size, updated every 40 milliseconds to match a 24fps video, is communicated to EdgeRIC, along with network states at each TTI. To simulate realistic conditions, we introduce a 20ms delay representing uplink latency on srsRAN. An RL agent, trained with EdgeRIC, aims to optimize video playout and resource allocation, responding to both network and application states to minimize video stalls.

The RL framework specifications for this setup are presented in Table 7. Here, apart from downlink backlog $B_i[t]$, and channel quality $CQI_i[t]$ of UE i, we augment the state with the length of the media buffer (in seconds) of the video streaming application of UE i, denoted $MB_i[t]$. The reward now depends on the stall performance of the applications, with smooth playout receiving a positive reward, and a stall receiving a large negative reward. We next conduct real-world experiments on video streaming. Comparing the performance of UE1 and UE2 controlled by the RL policy and the standard RAN scheduling algorithms in real-time, the RL policy outperformed the standard algorithms by incorporating "application awareness", demonstrating its potential to provide a quality of experience (QoE)-optimized solution with proper training. Figure 13(b) summarizes this statement showing the metrics observed by playing a video for 120s. Figure 13(c) presents the results of our experiment, which evaluated the stall performance of the video streaming application over the air in a static environment with a one-streamer and one-loader scenario for 30 s videos. We see that the RL trained policy only has about a third of the stalls experienced by the other approaches. Further, to answer the question if a cloud based RIC can support a video streaming application under a highly fluctuating channel condition, we implemented a variant of the max weight scheduling algorithm (infused with the application state, $w_i[t] = CQI_i[t]B_i[t]/MB_i[t]$) and benchmarked the stall performance for a 30s video. The result is depicted in Figure 13(a) which clearly indicates that an application aware real-time control outperforms a near-RT control approach.

8 Limitations and Future Work

We presented a platform EdgeRIC and its full stack emulator. When combined together, we can train and deploy cross-layer AI-optimization algorithms that can provide decision and control at a TTI-timescale. Future directions of this work are:
Scale to 100s of users: Our demonstrations are with four users, primarily due to the instability with our setup of srsRAN for a larger number of UEs. However, we have shown that EdgeRIC platform can send the messages for 100 UEs, with a RAN to EdgeRIC latency of about 100μs (Figure 6). That said, we have not tested over-the-air with support for many more simultaneous UEs, which is key future work.
Mutliple base-stations: Our work has demonstrated performance improvements for a single base-station. We anticipate with multiple base-stations, we would have an EdgeRIC attached to each base-station/DU, with Near-RT RIC coordinating all the EdgeRIC instantiations. This would involve problems in federated learning that we will explore.
Ethical concerns: Does not raise any ethical issues.
Acknowledgement: This work was funded in part by NSF Grants CNS 2312978, CNS 1955696, ECCS 2030245, CNS 2312979 and ARO grant W911NF- 19-1-0367. All opinions and findings are of the authors.

References

[1] Luca Baldesi, Francesco Restuccia, and Tommaso Melodia. ChARM: NextG spectrum sharing through data-driven real-time O-RAN dynamic control. *arXiv:2201.06326*, 2022.

[2] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1,2. Athena scientific Belmont, MA, 2017.

[3] Rajarshi Bhattacharyya, Archana Bura, Desik Rengarajan, Mason Rumuly, Bainan Xia, Srinivas Shakkottai, Dileep Kalathil, Ricky KP Mok, and Amogh Dhamdhere. QFlow: A learning approach to high qoe video streaming at the wireless edge. *IEEE/ACM Transactions on Networking*, 30(1):32–46, 2021.

[4] Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. SCOPE: An open and softwarized prototyping platform for NextG systems. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 415–426, 2021.

[5] Leonardo Bonati, Salvatore D'Oro, Michele Polese, Stefano Basagni, and Tommaso Melodia. Intelligence and learning in O-RAN for data-driven NextG cellular networks. *IEEE Communications Magazine*, 59(10):21–27, 2021.

[6] CurveZMQ. http://curvezmq.org/, 2023.

[7] Salvatore D'Oro, Michele Polese, Leonardo Bonati, Hai Cheng, and Tommaso Melodia. dapps: Distributed applications for real-time inference and control in o-ran. *IEEE Communications Magazine*, 60(11):52–58, 2022.

[8] Harish Kumar Dureppagari, Ujwal Dinesha, Raini Wu, Santosh Ganji, Woo-Hyun Ko, Srinivas Shakkottai, and Dinesh Bharadia. Realtime intelligent control for NextG cellular radio access networks. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, pages 567–568, 2022.

[9] Xenofon Foukas, Navid Nikaein, Mohamed M. Kassem, Mahesh K. Marina, and Kimon Kontovasilis. FlexRAN: A flexible and programmable platform for software-defined radio access networks. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, page 427–441, New York, NY, USA, 2016. Association for Computing Machinery.

[10] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. Taking 5g ran analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2023.

[11] Yu-Pin Hsu, Navid Abedini, Natarajan Gautam, Alex Sprintson, and Srinivas Shakkottai. Opportunities for network coding: To wait or not to wait. *IEEE/ACM Transactions on Networking*, 23(6):1876–1889, 2014.

[12] Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, and Lifeng Sun. QARC: video quality aware rate control for real-time video streaming based on deep reinforcement learning. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1208–1216, 2018.

[13] Ish Kumar Jain, Raghav Subbaraman, and Dinesh Bharadia. Two beams are better than one: Towards reliable and high throughput mmwave links. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 488–502, New York, NY, USA, 2021. Association for Computing Machinery.

[14] Ish Kumar Jain, Rohith Reddy Vennam, Raghav Subbaraman, and Dinesh Bharadia. mmFlexible: flexible directional frequency multiplexing for multi-user mmWave networks. *arXiv preprint arXiv:2301.10950*, 2023.

[15] Petteri Kela, Thomas Höhne, Teemu Veijalainen, and Hussein Abdulrahman. Reinforcement learning for delay sensitive uplink outer-loop link adaptation. In *2022 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, pages 59–64, 2022.

[16] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. EdgeRIC: Delivering realtime RAN intelligence. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 1162–1164, 2023.

[17] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. SpotFi: Decimeter level localization using WiFi. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 269–282, New York, NY, USA, 2015. Association for Computing Machinery.

[18] Merima Kulin, Tarik Kazaz, Ingrid Moerman, and Eli De Poorter. End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications. *IEEE Access*, 6:18484–18501, 2018.

[19] Panqanamala Ramana Kumar and Pravin Varaiya. *Stochastic systems: Estimation, identification, and adaptive control*, volume 75. SIAM, 2015.

[20] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac. GPAC: open source multimedia framework. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, page 1009–1012, New York, NY, USA, 2007. Association for Computing Machinery.

[21] Changqing Luo, Jinlong Ji, Qianlong Wang, Xuhui Chen, and Pan Li. Channel state information prediction for 5g wireless communications: A deep learning approach. *IEEE Transactions on Network Science and Engineering*, 7(1):227–236, 2020.

[22] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210. ACM, 2017.

[23] Mehrtash Mehrabi, Mostafa Mohammadkarimi, Masoud Ardakani, and Yindi Jing. Decision directed channel estimation based on deep neural network k -step predictor for mimo communications in 5G. *IEEE Journal on Selected Areas in Communications*, 37(11):2443–2456, 2019.

[24] Tommaso Melodia, Stefano Basagni, Kaushik R. Chowdhury, Abhimanyu Gosain, Michele Polese, Pedram Johari, and Leonardo Bonati. Colosseum, the world's largest wireless network emulator. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, page 860–861, New York, NY, USA, 2021. Association for Computing Machinery.

[25] Arupa Mohapatra, Natarajan Gautam, Srinivas Shakkottai, and Alex Sprintson. Network coding decisions for wireless transmissions with delay consideration. *IEEE Transactions on Communications*, 62(8):2965–2976, 2014.

[26] Mustafa Mohsin, Jordi Mongay Batalla, Evangelos Pallis, George Mastorakis, Evangelos K. Markakis, and Constandinos X. Mavromoustakis. On analyzing beamforming implementation in O-RAN 5G. *Electronics*, 2021.

[27] Kanthi Nagaraj, Dinesh Bharadia, Hongzi Mao, Sandeep Chinchali, Mohammad Alizadeh, and Sachin Katti. Numfabric: Fast and flexible bandwidth allocation in datacenters. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 188–201, New York, NY, USA, 2016. Association for Computing Machinery.

[28] Khaled Nakhleh, Santosh Ganji, Ping-Chun Hsieh, I-Hong Hou, and Srinivas Shakkottai. NeurWIN: Neural Whittle index network for restless bandits via deep RL. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[29] Ali ParandehGheibi, Muriel Médard, Asuman Ozdaglar, and Srinivas Shakkottai. Access-network association policies for media streaming in heterogeneous environments. In *49th IEEE Conference on Decision and Control (CDC)*, pages 960–965. IEEE, 2010.

[30] Jounsup Park, Philip A. Chou, and Jenq-Neng Hwang. Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):149–162, 2019.

[31] Ali Parsa, Neda Moghim, and Pouyan Salavati. Joint power allocation and mcs selection for energy-efficient link adaptation: A deep reinforcement learning approach. *Comput. Netw.*, 218(C), dec 2022.

[32] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms. *IEEE Transactions on Mobile Computing*, pages 1–14, July 2022.

[33] Vivek Raghunathan, Vivek Borkar, Min Cao, and P Roshan Kumar. Index policies for real-time multicast scheduling for wireless broadcast systems. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, pages 1570–1578. IEEE, 2008.

[34] Javad Razavilar, KJ Ray Liu, and Steven I Marcus. Jointly optimized bit-rate/delay control policy for wireless packet networks with fading channels. *IEEE Transactions on Communications*, 50(3):484–494, 2002.

[35] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. Flexric: An SDK for next-generation sd-rans. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '21, page 411–425, New York, NY, USA, 2021. Association for Computing Machinery.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[37] Stefania Sesia, Issam Toufik, and Matthew Baker. *LTE-the UMTS long term evolution: from theory to practice*. John Wiley & Sons, 2011.

[38] S. Shakkottai and R. Srikant. Network optimization and control. *Foundations and Trends in Networking*, 2(3):271–379, 2007.

[39] srsRAN, Inc. https://www.srslte.com/, 2023.

[40] Raghav Subbaraman, Yeswanth Guntupalli, Shruti Jain, Rohit Kumar, Krishna Chintalapudi, and Dinesh Bharadia. BSMA: Scalable LoRa networks using full duplex gateways. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, MobiCom '22, page 676–689, New York, NY, USA, 2022. Association for Computing Machinery.

[41] Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control*, pages 2130–2132. IEEE, 1990.

[42] Pratheek S. Upadhyaya, Aly S. Abdalla, Vuk Marojevic, Jeffrey H. Reed, and Vijay K. Shah. Prototyping next-generation O-RAN research testbeds with SDRs, 2022.

[43] Gongwei Xiao, Muhong Wu, Qian Shi, Zhi Zhou, and Xu Chen. DeepVR: deep reinforcement learning for predictive panoramic video streaming. *IEEE Transactions on Cognitive Communications and Networking*, 5(4):1167–1177, 2019.

[44] Lin Zhang, Junjie Tan, Ying-Chang Liang, Gang Feng, and Dusit Niyato. Deep reinforcement learning-based modulation and coding scheme selection in cognitive heterogeneous networks. *IEEE Transactions on Wireless Communications*, 18(6):3281–3294, 2019.

[45] Yuanxing Zhang, Pengyu Zhao, Kaigui Bian, Yunxin Liu, Lingyang Song, and Xiaoming Li. DRL360: 360-degree video streaming with deep reinforcement learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1252–1260. IEEE, 2019.

A Appendix

In the appendix, we provide additional results for different scenarios, which are not critical to the performance but provide visibility into the system.

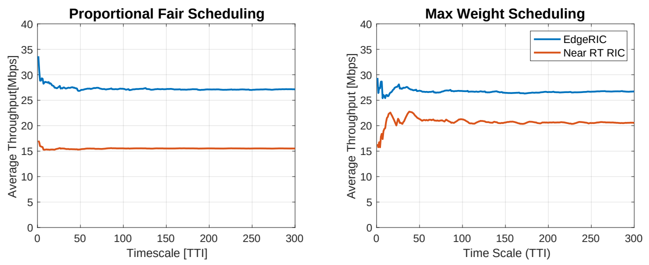


Figure 14: EdgeRIC Performance in a 4UE synthetic channel scenario.

Figure 14 shows that EdgeRIC outperforms Near Real-time RIC for implementation of microbenchmarks. This is the scenario with 4UEs on a synthetic channel trace. We implemented Proportional Fair Scheduling and Max Weight Scheduling as a μApp in EdgeRIC which exchanges state information and actions with RAN Stack over ZeroMQ-based communication. In order for performance comparison, we implemented Near RT RIC by imposing delays in message deliveries between RIC and RAN. We used synthetic CQI traces for 4 UEs and evaluated each algorithm's performance by comparing their average throughput. The graph on the left side shows average throughput of Proportional Fair Scheduling and the graph on the right side shows one of Max Weight Scheduling. A blue line in the graphs are for EdgeRIC while a red line for Near RT RIC. As Near RT RIC showed low average throughput due to the delay, EdgeRIC successfully supported those microbenchmarks to achieve their best throughput.

Table 7: Load: 30Mbps, Channel: 4UE synthetic channel

		EdgeRIC	15ms	30ms
Max CQI	Avg. Thrpt.	**26.3**	16.4	15.9
	BL[MB]	**1.22**	1.25	1.27
Prop. Fair.	Avg. Thrpt.	**24.27**	22.72	20.15
	BL[MB]	1.37	**1.08**	1.27
Max Weight	Avg. Thrpt.	**25.4**	19.8	19.1
	BL[MB]	1.33	**1.05**	0.99

Table 8: Throughput and Backlog Buffer evaluation of synthetic traces

	PPO	Max Weight	Max CQI
Synthetic Channel Traces			
Scenario 1	**27.8/0.38**	25.6/0.38	19.0/0.38
Scenario 2	**27.5/0.33**	24.7/0.51	18.9/0.38
Scenario 3	**26.8/0.38**	25.2/0.51	25.2/0.76

Figure 15 shows the real-time CQI traces we collected to characterize various CQI mobility in different environments. An x-axis is time in TTI unit and a y-axis is CQI showing real-time channel quality. The report period of CQI values was 2 TTI period which is approximately 2 mili-seconds. While a mobile robot has some drops in its CQI traces due to the radio block by the lid of the laptop mounted on the mobile robot when rotating, drone's swift motions caused radical ups and downs in its CQI traces. In car's CQI traces, its variation has a characteristic of a long period and smooth curves because of its slow acceleration and deceleration. For a turntable, we was able to generate fast angular acceleration which caused drastic and kind of periodic changes in its CQI traces by fast rotation.

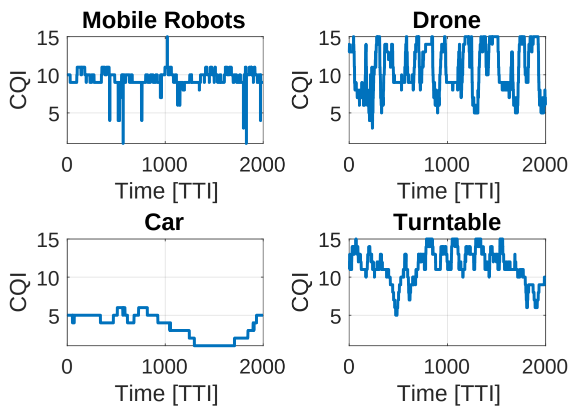


Figure 15: CQI Trace for different UEs

Figure 16 shows the training curves of RL PPO policy model and its throughput evaluations on DigitalTwin. The graphs in the first column illustrate the training curve for each scenario. We trained the policy network of RL for Scenario 1, 3 and 5 which are 2 Drone UEs, 2 Car UEs and 2 Drone UEs, and 2 Random Walk UEs and 2 Turntable UEs cases. We used specific CQI trace data to emulate channel conditions for individual scenarios. To train an RL agent using PPO, we designed a reward function to maximize throughput for general scenarios. Each iteration, we sampled 5000 data samples including previous state, previous action and current state at each iteration and updated policy network until the reward saturates. Most transient periods for training were less than 40 iterations and the training curves converged for all scenarios. The graphs in the second column describe the throughput evaluation of the trained policy model. In each graph, we compared the throughput of PPO to the ones of Max Weight and max CQI. In each algorithm, a red bar means the mean of its throughputs and a blue box their range. PPO obviously outperformed Max CQI, and even achieved almost the same throughput performance as Max Weight, which is usually considered as an optimum policy, with somehow higher ranges than Max Weight.

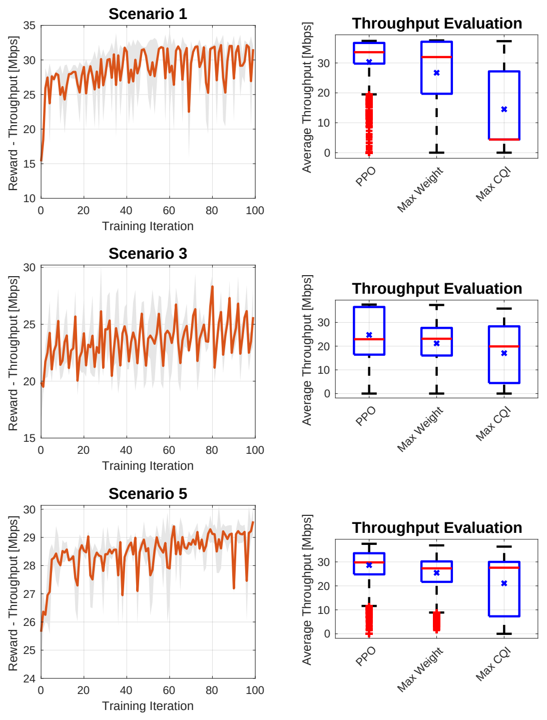


Figure 16: Can RL train and evaluate on EdgeRIC emulator?