CloudRIC: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing

Leonardo Lo Schiavo[*§], Gines Garcia-Aviles[†], Andres Garcia-Saavedra[‡]
Marco Gramaglia[§], Marco Fiore[*], Albert Banchs[*§], Xavier Costa-Perez[†‡¶]
[*]IMDEA Networks Institute, [†]i2CAT, [‡]NEC Laboratories Europe, [§]Universidad Carlos III de Madrid, [¶]ICREA

ABSTRACT

Open and virtualized Radio Access Networks (vRANs) are breeding a new market with unprecedented opportunities. However, carrier-grade vRANs today are expensive and energy-hungry, as they rely on hardware accelerators (HAs) that are dedicated to individual distributed units (DUs). In this paper, we argue that sharing pools of heterogeneous processors among DUs leads to more cost- and energy-efficient vRANs. We then design CloudRIC, a system that, powered by lightweight data-driven models, meets specific reliability targets while (*i*) coordinating access between DUs and heterogeneous computing infrastructure; and (*ii*) assisting DUs with compute-aware radio scheduling procedures. Experiments on a GPU-accelerated O-Cloud show that CloudRIC can achieve, respectively, 3x and 15x mean gains in energy- and cost-efficiency under real RAN workloads while ensuring 99.999% reliability even in dense scenarios.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Network reliability**.

KEYWORDS

vRAN, O-RAN, O-Cloud, Distributed Unit, HW Accelerators

ACM Reference Format:
Leonardo Lo Schiavo[*§], Gines Garcia-Aviles[†], Andres Garcia-Saavedra[‡], Marco Gramaglia[§], Marco Fiore[*], Albert Banchs[*§], Xavier Costa-Perez[†‡¶]. 2024. CloudRIC: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing. In *The 30th International Conference on Mobile Computing and Networking (ACM MobiCom '24), Sept. 30-Oct. 4, 2024, Washington D.C., USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3636534.3649381

ACM MobiCom '24, September 30-October 4, 2024, Washington D.C., USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *The 30th International Conference on Mobile Computing and Networking (ACM MobiCom '24), Sept. 30-Oct. 4, 2024, Washington D.C., USA*, https://doi.org/10.1145/3636534.3649381.

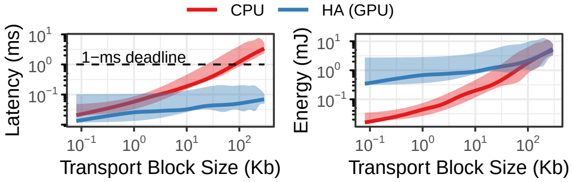

Figure 1: Mean (line) and max-min range (shaded area) latency and energy consumption to decode an LDPC-encoded transport block. Intel FlexRAN LDPC library [30] on an Intel Xeon Gold 6240R CPU core @ 2.40GHz; and commercial driver on an NVIDIA V100 GPU. Details in §3.

	CPU core	GPU	FPGA	ASIC
Programmable	Software	Software	Yes	No
TTM(1) (months) [45]	<2	<2	2	30
NRE(2) cost ($) [45]	0	0	0	350K-1000K
Unit cost ($)(3)	110	8000	4000	3000

(1) Time-To-Market (TTM). (2) Non-Recurrent-Engineering (NRE). (3) Intel Xeon 6240R CPU, NVIDIA V100 GPU, Intel PAC N3000 FPGA, and Intel ACC100 ASIC, as observed in Dec. 2022.

Table 1: Comparison of processors for 5G LDCP workload.

1 INTRODUCTION

Virtualized Radio Access Networks (vRANs) enable baseband processing on commercial off-the-shelf servers. This approach has many advantages over traditional hardwired RANs, such as mitigating vendor lock-in, streamlining upgrades, and enabling resource multiplexing [47]. Led by the O-RAN Alliance [22], practically all the industry is building vRANs [27, 59], breeding a new market with unprecedented business opportunities in an ossified RAN ecosystem [11]. Analysts project that open vRANs may outgrow the traditional RAN market by 2028, with $29B in revenue [4].

As later detailed in §2, a key component of O-RAN 5G base stations is the distributed unit (DU) that performs physical layer (PHY) tasks including forward error correction (FEC) [22]. DU functions must fulfill strict compute time guarantees since the signal processing pipeline they contribute to has hard deadlines in the 1-3 ms range [3, 19, 21]; also, these deadlines must be met with 99.999% (*5-nines*) probability to provide *reliability* [19], and avoid that users lose synchronization and drop connectivity (see [21]).

These settings make conventional virtualization approaches, which rely on software running in general-purpose CPUs, insufficient for industry-grade DUs: for instance, Fig. 1 (left) shows that a state-of-the-art FEC LDPC decoding library in a CPU can take over 1-3 ms to process a large transport block (TB) compromising the latency budget.

The industry today: DU-dedicated HAs. To address this, vRANs on the market today resort to offloading compute-intensive FEC tasks to dedicated hardware accelerators (HAs) *that are co-located with every DU* [20, 39]. HAs are ASICs [62], FPGAs [31], or GPUs [63] that, using in-line or look-aside models (see §2.2), can provide >10× latency gains over CPUs when processing large TBs, as shown in Fig. 1 (left).

However, HAs are expensive, as exemplified in Table 1, and are energy-hungry, as shown in Fig. 1 (right) for a GPU-based HA. More broadly, an Intel ACC100 ASIC and an NVIDIA V100 GPU consume up to 52W and 250W respectively [48, 62], i.e., 20-82% of the overall consumption of a commodity server [33]. In fact, the economic and energy costs of DU-dedicated HAs are so high that they have cast doubts in the industry about this approach, as implied by top figures of, e.g., Nokia [60], Ericsson [61] or Mavenir [38].

Our proposal: HA and CPU pooling. We present a more efficient solution, which combines the next two approaches. ① *Opportunistic HA offloading*. CPUs may handle some 5G PHY workloads without the assistance of HAs by exploiting SIMD programming and other optimizations [12, 24]. However, CPUs alone cannot ensure 5-nines reliability for *all* workloads, as we illustrate in §3 and, consequently, they are usually shunned for this job in industry-grade RANs [20]. Instead, we prove in §3 that they can be a valuable complement to HAs in those tasks, and propose to *balancing DU workloads between CPUs and HAs* as a way to substantially improve the cost- and energy-efficiency of vRANs.

The rationale is that minimizing processing latency brings no benefit as long as deadlines are met, hence CPUs may be occasionally exploited to alleviate the HAs' energy toll. As an example, Fig. 1 (left) shows that a CPU core can decode within 1 ms TBs below 100 Kb (which correspond to a large portion of today's real-world TBs, see §3) consuming ~5.7× less energy than a GPU-based HA (right plot).
② *Processor pooling*. HAs co-located with (and thus exclusively used by) individual DUs suffer from low usage under real workloads, as we show in §3. We seize this opportunity to *share HAs among multiple DUs*, so as to amortize the cost of these expensive resources, and provide the needed acceleration at an affordable cost per DU.

The concept of RAN pooling is not new. Indeed, 71% of US operators intend to realize RAN pooling solutions by 2025 [28], and some already implement it [13], but the traditional RAN centralization approaches *only* exploit long-term traffic variations, such as day-night ones [9, 46], which is insufficient for cost-efficient RAN virtualization.

Contributions. Designing the above solution is technically challenging as pooling heterogeneous computing resources efficiently requires: (*i*) harnessing real-time multiplexing opportunities at sub-millisecond timescales where both PHY processing latencies and user loads fluctuate [19]; and (*ii*) anticipatory operation that effectively copes with fluctuations in the future user demand. Note that, because resources are no longer over-dimensioned, rare peak loads risk violating deadlines, which compromises reliability.

Moreover, the solution to these challenges goes well beyond the current capabilities of O-RAN. Although O-RAN provides convenient abstractions for heterogeneous processors, it falls short to support (*i*) real-time coordination among DUs and (*ii*) radio scheduling policies that are *compute-aware*, two requirements that, as shown in §3, are essential to attain multiplexing gains reliably. In this paper, we realize the above solution, making three main contributions in the process.

- In §3, we study CPU- and GPU-based 5G PHY processors using real-world mobile traffic data over an implementation of the standard-defined O-RAN Acceleration Abstraction Layer (AAL). Our dataset is publicly available. We opt for GPUs as they represent a new and still little understood 5G HA, with features like AI-on-5G and programmability [36] that are attractive for operators [35]. To our knowledge, this is the first in-depth analysis of this emerging resource for 5G. We find that:
  - Dedicating HAs to individual DUs results in dramatic under-utilization of expensive resources with real-world workloads, which supports our strategy of HA pooling;
  - SIMD-capable CPUs can handle 5G FEC workloads reliably in a wide range of contexts, which motivates our opportunistic HA offloading model; and
  - Central coordination of shared CPU/HA resources and their joint control with radio scheduling are key to attain multiplexing gains while ensuring reliability.
- Motivated by our empirical analysis, in §4 we present CloudRIC, a brokering system that jointly controls access to shared resources of heterogeneous processors and optimizes radio scheduling policies across multiple DUs in real-time. CloudRIC seamlessly integrates into O-RAN, and comprises two key elements:
  - An AAL Broker that maximizes efficiency by exploiting CPUs and performing HA offloading opportunistically.
  - A Real-Time RAN Intelligent Controller (RT-RIC) that provides compute-aware radio policies to ensure reliability with our AAL Broker solution in shared platforms.
- In §5, we present a prototype of CloudRIC based on DPDK's Environment Abstraction Layer (EAL) and ONNX, which we then comprehensively evaluate in §6 with realistic mobile traffic demands. Results demonstrate that:
  - CloudRIC incurs low overhead, <10 μs to process scheduling grants and <50 μs to route PHY processing tasks;
  - CloudRIC meets the targeted 99.999% processing reliability even in severely congested scenarios; and
  - CloudRIC achieves average gains equal to 3x and 15x in energy and cost efficiency, respectively, over the industry standard approach with real-world RAN workloads.

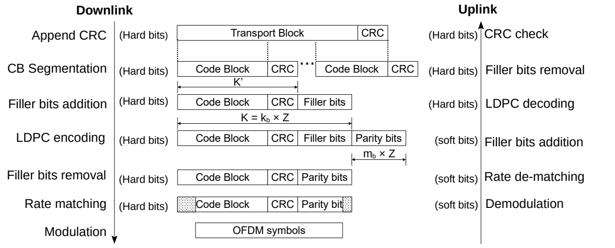

Figure 2: O-RAN DU data processing pipeline.

2 BACKGROUND

Our work builds on 5G New Radio and its data processing pipeline, the O-RAN architecture and its AAL, and current hardware acceleration strategies, which are introduced next.

2.1 5G New Radio

5G base stations comprise a radio unit (RU), which performs basic radio operations such as signal sampling; a central unit (CU), which processes the highest layers; and a DU, which processes the PHY, MAC, and radio link control layers [49].

New Radio (NR) is 5G's PHY/MAC interface. Our focus is on sub-6GHz bands, which allow up to 100 MHz per carrier and have flexible *numerology* $\mu = \{0,1,2\}$ [10]. The basic spectrum unit is the resource block (RB), which encompasses 12 subcarriers with $15 \cdot 2^{\mu}$-KHz spacing. Time is divided into 1-ms subframes, each carrying 2^{μ} slots with, usually, 14 OFDM symbols lasting $66.7 \cdot 2^{-\mu}$ μs. Every Transmission Time Interval (TTI), often one slot, the DU's MAC schedules one TB for/from every active User Equipment (UE), which are signalled to UEs by *grants*. The TB size depends on the numerology, the amount of buffered data, the DU's RB scheduling policy, and the modulation and coding scheme (MCS), selected based on the signal-to-noise ratio (SNR). Uplink TBs must be sent within $K2$ slots of receiving the grant [1].

Fig. 2 shows the pipeline of DU operations required to process a TB. At the transmitter side, TBs are divided into code blocks (CBs) with individual CRC fields. Filler bits adapt the CB size to the requirements of the LDPC encoder used for FEC, which produces a codeword with parity bits. Finally, the codeword is aligned to the capacity of the allocated RBs (which depends on their MCS) via rate matching, by applying puncturing or repetition. At the receiver side, a soft-output detector computes the reliability of the data as log-likelihood ratios (LLR) called *soft bits*. Then, an LDPC decoder maps soft bits into hard bits through an iterative belief propagation algorithm. The algorithm terminates after a maximum number of iterations (usually 10), or earlier if CRC validates the codeword. The TB is reconstructed once all of its CBs are successfully decoded. More details can be found in [8].

To adhere to 3GPP and O-RAN requirements [3, 49], processing the heavier LDCP tasks has a deadline $D = \{1, \ldots, 3\}$ ms, depending on the base station, which must be met with 99.999% probability to reach the industry's *5-nines* reliability target [19]. This is achieved today with DU-dedicated HAs.

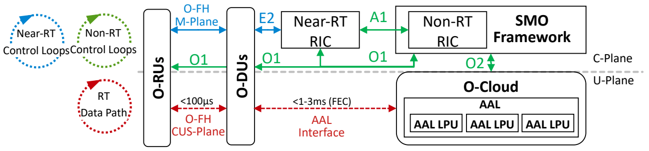

Figure 3: O-RAN architecture.

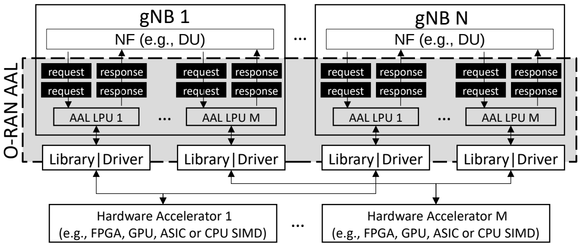

Figure 4: O-Cloud high-level architecture.

2.2 Hardware Acceleration

There are two hardware acceleration models, typically implemented with ASICs, FPGAs or GPUs [20]: *in-line*, which processes the pipeline of Fig. 2 as wireless symbols arrive, without software intervention; and *look-aside*, which operates on data managed by a software controller to perform selected tasks like LDPC decoding.

Traditionally, in-line HAs offer lower latency than look-aside HAs because the former does not require software mediation. However, in-line HAs tie the complete pipeline of Fig. 2 to the choice of HA, thus limiting the advantages of virtualization. Moreover, the performance gap between the two models is quickly closing [32], suggesting that look-aside HAs may become predominant.

2.3 O-RAN

The O-RAN Alliance is a major carrier-led effort to define an open RAN architecture [22], depicted in Fig. 3. The control plane includes two components, the Non-Real-Time RAN Intelligent Controller (Non-RT RIC) and the Near-RT RIC, that use A1 and E2 interfaces to manage network functions (NFs) such as DUs at, respectively, >1 s and >100 ms timescales.

The data plane has the O-Cloud, which provides computing resources, including CPUs and HAs, to NFs through an Acceleration Abstraction Layer (AAL) [51]. The AAL abstracts O-Cloud resources as *Logical Processing Units* (LPUs). As shown in Fig. 4, each LPU is dedicated to one NF via individual FIFO queues. Consequently, though a physical processor (CPU or HA) can be shared among NFs, the state of each LPU (e.g., its queue occupancy) is *not* shared.

The O-Cloud is governed by the Service & Management Orchestrator (SMO) through the O2 interface, but operates on several-second timescales. Moreover, the Near-RT RIC lacks O-Cloud visibility, hindering real-time compute-aware radio policies and DU coordination — both crucial for achieving efficiency gains reliably as shown in §3. While CloudRIC is O-RAN compliant to facilitate its adoption, it provides key extensions, presented in §4, to address these limitations.

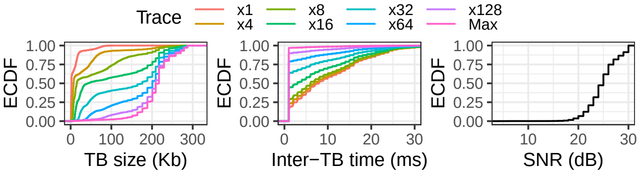

Figure 5: UE demand profiles used for evaluation.

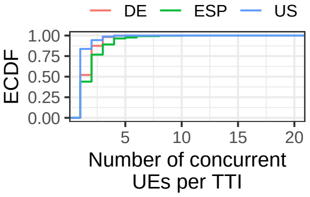

Figure 6: UEs concurrently active at TTI timescales.

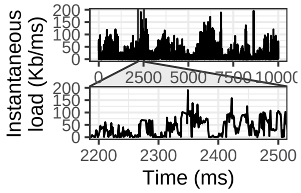

Figure 7: Instantaneous load fluctuations in a cell.

3 ANALYSIS

Using Falcon [18] and 5GSniffer [42], we tracked the workload dynamics experienced by several (sub)urban cells in Madrid, Spain (Vodafone, April 2021), in Frankfurt, Germany (DT, Dec. 2022), and in Boston, US (T-Mobile, May 2023). Our goal is to use this data to emulate the behavior of real UEs.

Consistent with previous studies [19, 55], we observe that the individual loads and the number of concurrently active UEs are low at the TTI scale. The "x1" curves in Fig. 5, which characterize each UE's buffer (left), inter-arrival time (middle), and SNR (right) at the TTI level, show a median and 99th percentile UE buffer of 2 Kb and 78 Kb, respectively. Moreover, Fig. 6 indicates that the median and 99th percentile of active UEs in one TTI is only 1 and 6, respectively.

To emulate higher network demands, we generated additional profiles by amplifying our traces by factors of 4, 8, and so forth, denoted as "x4", "x8", etc. in Fig. 5. With the mobile traffic CAGR pegged between 25-30% [16], these multipliers enable us to project expected workloads up to 2030, all the while preserving the genuine dynamics of real users. The additional "Max" profile, where the UE is consistently backlogged, let us analyze worst-case scenarios too.

These traces also show remarkable burstiness at TTI level. For instance, Fig. 7 shows a 10-second cell traffic snapshot that reveals wide fluctuations within milliseconds. This observation, which is also in line with the literature [19, 55], suggests that, *to harness pooling opportunities in real-world cells while meeting the 5-nines reliability target, it is crucial to develop effective real-time control schemes that can react appropriately upon quick yet infrequent load peaks.*

3.1 Reliability of legacy CPUs and HAs

The industry today favors in-line or look-aside HAs (ASIC, FPGA, or GPUs) for 5G FEC workloads due to concerns about CPU reliability [39]. Indeed, there exist fundamental limitations to parallelizing individual decoding request across CPU cores [17], hence state-of-the-art LDPC libraries such as FlexRAN [30] and others [23, 26] just exploit data parallelization using SIMD programming and task parallelization over concurrent decoding tasks [12, 24]. Yet, those strategies cannot bound the latency of *individual* tasks, which prevents CPUs from guaranteeing reliability for *all* workloads alone.

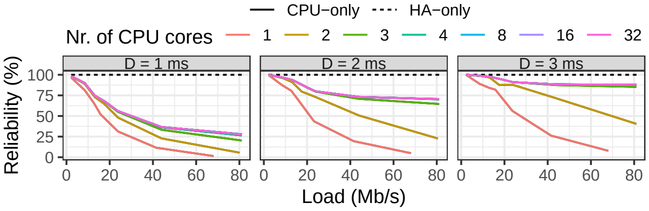

Figure 8: Reliability performance as a function of the DU load for a CPU-only processor and a HA-only processor.

To precisely quantify the reliability that CPU- and HA-based processors may achieve, we measure in Fig. 8 the ratio of TBs processed within a deadline D (reliability) for two solutions. The first one uses FlexRAN, a de-facto standard library [12, 19, 24], with the above optimizations on a pool of Intel Xeon Gold 6240R CPU cores. The second one uses a commercial NVIDIA V100 GPU decoder, which represents a new and still little understood HA with compelling features like CUDA, CPU-like time-to-market, and AI-on-5G [36].

We evaluate both processors for different loads by varying the profile of the active UEs as explained before ("x1", "x2", and so on). Every TTI, a number of UEs become active following the distribution shown in Fig. 6, each receiving a fair amount of the radio resources from a 100-MHz DU. Then, these UEs generate TBs following the corresponding profile and radio allocation by executing the downlink pipeline of Fig. 2. To emulate the wireless channel, we add AWGN noise with zero mean and the appropriate variance to obtain the target SNR across all OFDM symbols received by the DU.

To ground our tests on realistic settings, we implemented O-RAN's AAL using DPDK's Wireless Baseband Device Library (BBDev) [15], which provides abstractions for wireless processing tasks that can be used to implement the LPUs for both processors [50] (~1K lines of C code). The signals demodulated by the DU are encoded as soft bits, as shown in the uplink pipeline of Fig. 2, stored in a memory pool (*mempool*), and then allocated to the corresponding LPU.

As shown in Fig. 8, the GPU (dotted black line), just like any look-aside or inline HA, can sustain 100% reliability regardless the load. Conversely, the CPU-only approach can only meet the 99.999% reliability target for small loads. This is naturally worse for smaller CPU pools as queuing effects cause additional delays but reliability drops regardless the pool size because higher loads carry larger TBs—which can exceed deadlines on CPUs—with higher probability.

These results illustrate the underlying reasons why CPUs are ignored by the industry for 5G FEC. *In contrast, we show next that CPUs can be a valuable complement to HAs as long as radio and computing resources are jointly controlled.*

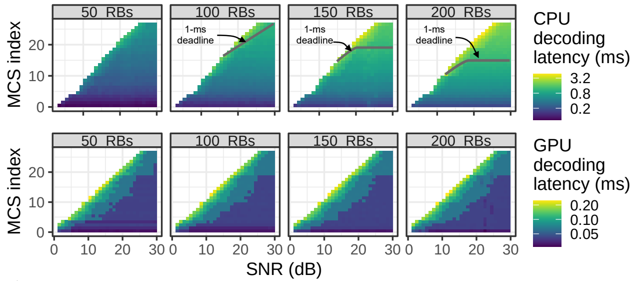

Figure 9: Median latency in decoding TBs of different sizes (i.e., combinations of RBs and MCS) under varied SNR conditions with CPU and HA processors. Log-scale z-axis.

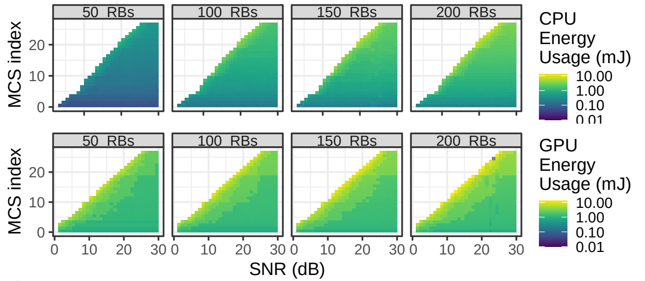

Figure 10: Median energy to decode TBs of different size (i.e., combinations of RBs and MCS) under varied SNR conditions with CPU and HA processors. Log-scale z-axis.

3.2 Complementarity of CPUs and HAs

We next experimentally investigate the performance of both categories of 5G FEC processors, unveiling their complementarities. For the CPU solution, we set the pool to 16 cores which our earlier results proved to be sufficient (see Fig. 8). For each test, we select an MCS $m \in \mathcal{M} := \{0, 1, \dots, 27\}$ (see [2, Table 5.1.3.1-2]), an SNR $s \in \mathcal{S} := \{1, 2, \dots, 30\}$ dB, and a bandwidth $b \in \mathcal{B} := \{1, 2, \dots, 250\}$ RBs to generate 10 random TBs as indicated before, adding up to ∼2M samples[1].

3.2.1 Latency. Using a colored gradient, Fig. 9 shows the median latency of the CPU running FlexRAN (top) and of the GPU-based HA (bottom) for all combinations from $\mathcal{S}$ and $\mathcal{M}$ and for subset of $b \in \{50, 100, 150, 200\} \subset \mathcal{B}$. The blank cells indicate contexts that cannot be decoded within 10 iterations.

The HA provides roughly an order of magnitude improvement in latency, as expected. More interestingly, Fig. 9 unveils for the first time key properties of the latency achieved by a GPU-based 5G HA, such as its invariance to the TB size or complex relationship with MCS and SNR combinations. Yet, *our CPU solution can decode TBs within common 1-3 ms deadlines [19, 21] in a wide range of contexts*, those below the grey line in the figure. In fact, 100% of the TBs observed in the "x1" UE profile fall within this range.

The plot also explains the high variance depicted in Fig. 1, as different combinations of MCS, SNR and TB size require a different number of decoding iterations. Our dataset includes experiments with alternative libraries [23, 26] showing that different decoding algorithms and implementations also have a remarkable impact on latency. Therefore, *there exists a complex relationship between TB context (MCS, SNR, number of RBs), the processor, the decoder implementation, and performance, which suggests data-driven models for proper control.*

3.2.2 Energy consumption. Fig. 10 shows the energy consumed by the CPU and the HA processors. For the less demanding bandwidth setting, the HA consumes more than 28× the energy required by the CPU. Yet, for the most exacting bandwidth configuration, the HA consumes *only* around 71% more than the CPU. The latency provided by CPUs precludes their use in specific contexts; however, *the remarkable advantages in terms of capital (see Table 1) and operating (i.e., energy) costs still make CPUs appealing to process less-stringent TBs*, which are rather frequent as discussed before.

3.3 HA multiplexing opportunities

Let us now explore HA sharing opportunities as a means to improve cost-efficiency. Fig. 11 depicts the relative busy time of the GPU-based HA (y-axis) to process the workload of a varying number of concurrently active UEs (x-axis) with different profiles (colors) in a single 100-MHz 5G DU. For "x1"-generated workloads, the HA utilization is below 12% even with 10 concurrently active UEs. In fact, 10 "Max" UEs concurrently active every TTI—or 85 "x4" UEs, though not shown in the figure—would be required to saturate the HA. In conclusion, *modern HAs are largely underutilized by individual DUs when handling real-world workloads, which creates clear opportunities for sharing, even in a relatively far future.*

A legitimate ensuing question is whether HA sharing among multiple DUs can be sustained by the fronthaul network (FH) [64] that connects them to their corresponding RUs as shown in Fig. 3. O-RAN uses a 7-2x DU-RU split with an open FH interface based on eCPRI that has a latency tolerance of 100 μs [52]. Assuming a 5-μs/Km propagation delay and a 2D Manhattan tessellation model, the area of RUs that can be connected to a single location is up to 400 Km^2 [49]. Moreover, assuming PCIe v3.0+ bus and 100-GbE FH interfaces, we could aggregate up to 3.8 GHz of radio spectrum per server [54]. Hence, *it is technically feasible to aggregate the workload of multiple DUs and multiplex the resources of high-performing HAs in centralized edge clouds.*

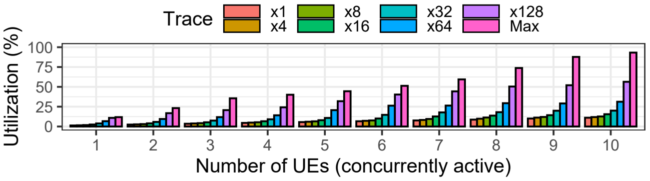

Figure 11: GPU HA mean usage for different 5G workloads.

[1] The dataset is publicly available at https://doi.org/10.5281/zenodo.10691661

3.4 O-RAN AAL policies

The insights above promote a real-time sharing of both high-performance HAs and low-consuming CPUs across DUs. To inform our solution design, we next explore the space of possible policies that can balance DU workloads among shared heterogeneous processors at sub-ms timescales.

Our experiments involve a varying number of 100-MHz 5G DUs, with 5 active "x4" UE each, that aggregate workloads between 85 Mb/s and 5 Gb/s. We also set $D = 3$ ms (for the moment), and discard overdue TBs. We consider the same two processors used before. In this case, each DU locally subscribes to two LPUs, each handling one shared processor. As explained in §2.3, each DU maintains exclusive local access (with its own queues, as depicted in Fig. 4). Then, for every TB, the corresponding DU selects an LPU following a *policy*.

With this setup, we study conceptual policies of increasing complexity, several of which are not viable in practice as they assume perfect knowledge of the future or functions not supported by O-RAN today. Yet, our goal now is studying potential gains; we will present a practical solution in §4.

3.4.1 Coordination. The first policy, "O-Greedy", follows a simple heuristic: given a TB, the DU selects the faster HA, as long as its local LPU queue is not full; otherwise, it falls back to CPU. Fig. 12 (left) shows the system reliability (i.e., the fraction of TBs processed within D) as a function of the aggregated network demand. Reliability quickly drops when the demand exceeds 200 Mb/s as the local LPU queues, specific to each DU, grow too much with respect to the capability of the HA to handle the demand of all DUs, which results in TBs frequently missing the deadline D. When all the LPU queues saturate, all TBs miss D and reliability drops to 0%.

Inspired by [40], in "O-MWT" (O-RAN Minimum Waiting Time) DUs compute the completion time at each LPU *by looking at the occupancy of its LPU queues* for every TB, and then select the fastest one. We model the performance of both LPUs using the dataset in §3.2, and let DUs have perfect knowledge of the exact processing latency of each new TB *a priori* to make optimal LPU choices. Though not realistic in practice, this policy illustrates the potential gains from exploiting *queuing information*, which in Fig. 12 amount to 18% higher reliability and 10% lower energy-per-bit cost on average. Yet, given sufficiently high workloads, "O-MWT" also drops to 0% reliability at loads of 2 Gb/s or higher.

Under "C-MWT" (Coordinated MWT), the DUs consider each other's LPU queues to calculate the actual completion time of new TBs based on the system-wide load. This approach further increases reliability by 15% and reduces the energy cost by 19% on average over O-MWT; however, it requires shared knowledge of LPU states across DUs, which is currently not supported by O-RAN, as explained in §2.3. Moreover, it still cannot avoid reliability loss.

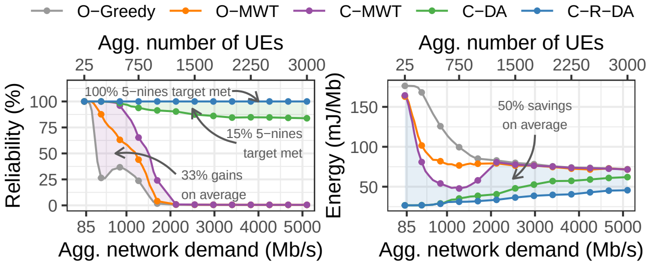

Figure 12: Performance of a heterogeneous O-Cloud with a varying number of DUs hosting "x4" UEs. Deadline D = 3 ms.

3.4.2 Deadline awareness. Minimizing processing latency, as done by C-MWT, tends to overuse the HA, which incurs an energy toll, and neglect CPUs; moreover, when TBs are eventually allocated to the CPUs, it is done ignoring whether they can process those TBs within their deadline, which causes unreliability. Hence, we test a policy "C-DA" (Coordinated Deadline-Aware) that aims at exploiting CPUs as much as possible. Under C-DA, given a TB, the DU calculates whether the TB can be processed within D by the CPU pool, and only offloads the TB to the HA otherwise. Thus, in addition to coordination, the policy requires *deadline awareness.* Fig. 12 shows that C-DA boosts reliability and enables *opportunistic offloading* to high-performance HAs that save up to 80% of the energy cost for low-loaded scenarios.

3.4.3 Joint radio and computing control. All previous policies fall short of achieving the 5-nines reliability target, even under mild average workloads. This is due to the burstiness inherent to real workloads, discussed in §3, which occasionally introduces peaks that exceed the computing capacity.

The problem cannot be solved via LPU queue dimensioning: as the decoding latency of a single TB does not depend solely on its size but also on the associated SNR and MCS (see §3.2), bounding the amount of bits (or TBs) allowed in a queue cannot deterministically bound the processing time and, hence, does not yield guarantees to meet deadlines. Instead, we argue that a better approach is to adapt the workload to the processing capacity *proactively*, through compute-aware radio policies that guarantee reliability.

We thus experiment with "C-R-DA" (Coordinated Radio-controlling Deadline-Aware), a policy that extends C-DA by throttling down radio grants when required to ensure that every TB can be processed in time afterwards. Note that, to explore the maximum gain of such a *compute-aware radio scheduling* strategy, C-R-DA has access to perfect knowledge of the future requests, which is clearly not possible in practice. However, C-R-DA shows how to efficiently trade off radio resources—*wasted* by the other policies anyway—for reliability when the O-Cloud gets congested and how it addresses the CPU reliability concern of §3.1. Indeed, Fig. 12 shows that C-R-DA consistently meets the 5-nines reliability target with a 18% lower energy cost than C-DA on average.

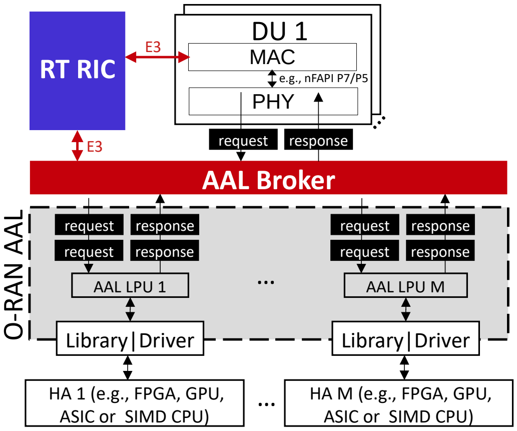

Figure 13: CloudRIC-powered O-Cloud high-level architecture. CloudRIC extensions are highlighted in red and blue.

4 CLOUDRIC SYSTEM DESIGN

The analysis in §3 sets a roadmap for more cost- and energy-efficient vRANs, which builds on (*i*) centralized allocation of processing tasks to shared HAs and CPUs, and (*ii*) a policing of radio schedulers that is sensitive to congestion in a shared O-Cloud, which, though no longer over-dimensioned, it must effectively accommodate sporadic peak demands.

We propose CloudRIC, an O-RAN compliant model that implements the complete design above with a three-fold goal, ordered by priority: (*i*) processing the DUs workload within a predefined deadline D with 99.999% probability; (*ii*) maximizing network throughput; and (*iii*) minimizing energy consumption. A high-level view of a CloudRIC-powered O-Cloud is depicted in Fig. 13. Our solution integrates seamlessly into the standard O-RAN architecture, to which it adds two key enhancements, highlighted in red in Fig. 13:

- A "Real-Time RIC (RT-RIC)", which audits radio grants issued by DUs so as to guarantee that all the scheduled TBs can be processed by the O-Cloud on time. Different from the existing O-RAN RICs, our RT-RIC assists DUs with compute-aware radio policies in real time.
- An "AAL Broker (AAL-B)", which presents DUs with a single abstraction of the O-Cloud, and enables a centralized coordination of its processors, via two sub-components:
  - "AAL-B User Plane" (AAL-B-UP) acts as a proxy between O-RAN NFs (DUs, in our case) and O-RAN AAL. From the NFs' viewpoint, the AAL-B-UP behaves as a virtual LPU that abstracts all the resources in the O-Cloud. From the perspective of the AAL, the AAL-B-UP appears as a virtual DU that is associated with the actual LPUs. Its job is routing arriving TBs from UEs to an LPU that is assigned by the AAL-B-CP.
  - "AAL-B Control Plane" (AAL-B-CP) schedules granted TBs to LPUs in a way to guarantee that PHY processing deadlines are met at minimum energy cost.

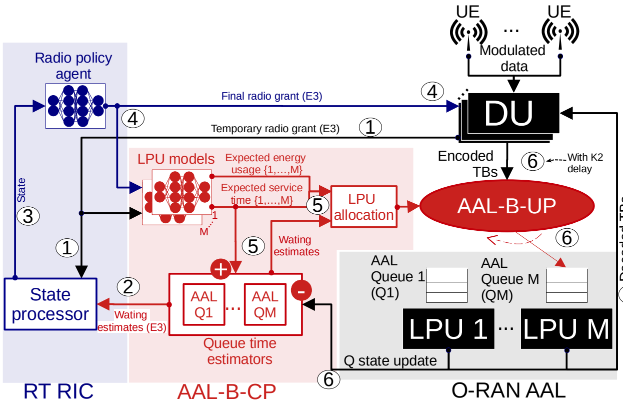

Figure 14: Detailed architecture of CloudRIC.

Component	Operations	**Time budget (μs)**	**Validation**
RT RIC	① - ④	TTI ∈ {250, 500, 1000}	§5.2
AAL-B-CP	⑤	~200	§5.2
AAL-B-UP	⑥	D (1000 - 3000, see §2.1)	§5.1, §6

Table 2: Time budget for CloudRIC operations, with pointers to the sections where we demonstrate budget compliance.

Given the real-time nature of the system, we expect DUs, RT-RIC and AAL-B to be deployed in the same physical infrastructure (e.g., an edge data center), and to be connected via a *new* E3 real-time interface that may be implemented using shared memory or low-latency tools such as Zenoh [41].

Our approach achieves three key results:

(1) By centralizing the allocation of computing resources with AAL-B, we can efficiently police the load from multiple DUs across heterogeneous processors, as per §3.4.1;
(2) By controlling the grants of DUs, the RT-RIC can throttle radio resources when LPU Queues are congested and ensure that deadlines are met, as per §3.4.2 and §3.4.3.
(3) By decoupling AAL-B-CP and AAL-B-UP, we can minimize data-plane overhead as we show in §4.1 and §5.1.

4.1 Detailed design & Workflow

Fig. 14 details the design of CloudRIC for uplink operation, which is the most challenging in a vRAN. The workflow follows steps ①–⑥, whose time budget is shown in Table 2.

① **Temporary grants.** Each radio grant $\bar{g}^{(i)}, i \in \{1, 2, \dots\}$ generated by the DUs becomes a *temporary grant*. Prior to being allocated, it is sent to the RT-RIC, as a tuple containing the bandwidth $\bar{b}^{(i)}$ (number of RBs), selected MCS $m^{(i)}$, UE's SNR $s^{(i)}$, and corresponding TB size (bits) $\bar{t}^{(i)}$, i.e.,

$$\bar{g}^{(i)} := [s^{(i)}, m^{(i)}, \bar{b}^{(i)}, \bar{t}^{(i)}].$$

We use the top bar $\bar{\cdot}$ to indicate that element $\cdot$ is *temporary.*

② **Latency estimation.** The RT-RIC gets from AAL-B-CP estimates of the waiting time $\hat{w}^{(i)} := [\hat{w}_1^{(i)}, \dots, \hat{w}_M^{(i)}]$ at each LPU queue $Q_n^{(i)}$, for all LPUs $n \in \mathcal{L} := \{L_1, \dots, L_M\}$ exposed by O-RAN AAL. The top hat $\hat{\cdot}$ indicates that $\cdot$ is *a prediction.*

Queue time estimators in AAL-B-CP produce $\hat{w}_n^{(i)}$ by keeping track of the time foreseen to serve all grants already

Algorithm 1 Greedy LPU allocation

```
Require: $D, t^{(i)}, \hat{w}_n^{(i)}, \hat{d}_n^{(i)}, \hat{e}_n^{(i)}, \forall n \in \mathcal{L}$
  $\mathcal{L}_1 = \mathcal{L}_2 = \emptyset$
  for $n \in \mathcal{L}$ do
      if Available Space in $Q_n > t^{(i)}$ then $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup n$
  for $n \in \mathcal{L}_1$ do
      if $\hat{w}_n^{(i)} + \hat{d}_n^{(i)} < D - \delta$ then $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup n$
  if $\mathcal{L}_2 \neq \emptyset$ then $L \leftarrow \arg\min_{k \in \mathcal{L}_2} \hat{e}_k^{(i)}$
  else $L \leftarrow \arg\min_{k \in \mathcal{L}_2} \hat{w}_k^{(i)} + \hat{d}_k^{(i)}$
  return $L$
```

queued in each $Q_n^{(i)}$, $\forall n \in \mathcal{L}$. These estimators perform simple calculations that rely on the (expected) LPU processing time $\hat{d}_n^{(k)}$ of each grant $g^{(k)}$ in $Q_n^{(i)}$, and on (real) time elapsed since the head-of-line grant started to be served. Step ⑤ explains how we derive $\hat{d}_n^{(i)}, \forall i, n$.

③ **Radio policy.** The RT-RIC *state processor* consolidates all the above information into a single feature vector $x^{(i)}$,

$$x^{(i)} := [s^{(i)}, m^{(i)}, \bar{t}^{(i)}, \hat{w}^{(i)}].$$

Given $x^{(i)}$, a *radio policy agent* ρ computes a radio allocation policy $r^{(i)} := \rho(x^{(i)}) \in \mathbb{R}_{0 \le r^{(i)} \le 1}$, where $b^{(i)} := r^{(i)} \cdot \bar{b}^{(i)}$ is the bandwidth (RB count) allowed for the final grant ($b^{(i)} \le \bar{b}^{(i)}$). The design of ρ is in §4.2.

④ **Final grant.** Policy $r^{(i)}$ determines a final grant

$$g^{(i)} := [s^{(i)}, m^{(i)}, b^{(i)}, t^{(i)}]$$

for the UE, where its TB size $t^{(i)}$ is computed as per 3GPP specs [2]. Importantly, the grant must be compiled within the granularity of a TTI, as shown in §2.1 and in Table 2 for the time budget of operations ①–④. Then, as shown in Fig. 14, $g^{(i)}$ is communicated to both the corresponding DU and the AAL-B-CP through interface E3. The DU can now notify $g^{(i)}$ to the UE as specified by 3GPP.

⑤ **LPU allocation.** The AAL-B-CP also receives $g^{(i)}$, which is used to allocate the computing resources necessary to process the grant. To this end, *LPU models* $\{\nu_n, \mu_n\}$ of each LPU $n \in \mathcal{L}$ are used to estimate the latency $\nu_n(g^{(i)}) := \hat{d}_n^{(i)}$, without queuing, and energy $\mu_n(g^{(i)}) := \hat{e}_n^{(i)}$ required by LPU n to process the TB associated with grant $g^{(i)}$. As they are stateless models, $\{\nu_n, \mu_n\}$ can be built with neural networks trained offline for each LPU. We present their design in §4.3.

Informed by the output of all LPU models ($\hat{d}_n^{(i)}$, $\hat{e}_n^{(i)}$) and queue time estimators ($\hat{w}_n^{(i)}$ from ②), the *LPU allocation function* uses the greedy Algorithm 1 to pre-assign one LPU to TB $g^{(i)}$. Algorithm 1 first prunes those LPUs that do not have queuing room, which yields $\mathcal{L}_1 \subseteq \mathcal{L}$. Then, using the waiting $\hat{w}_n^{(i)}$ and processing time estimates $\hat{d}_n^{(i)}$, it removes from $\mathcal{L}_1$ all LPUs for which $\hat{w}_n^{(i)} + \hat{d}_n^{(i)}$ falls outside a guard period δ of the deadline D, which yields $\mathcal{L}_2 \subseteq \mathcal{L}_1$. Finally, the LPU $k \in \mathcal{L}_2$ that can process the TB with the least amount of expected energy $\hat{e}_k^{(i)}$ is selected. The result is communicated to the AAL-B-UP (to route the grant) and to the relevant queue time estimator (to update its estimate).

As explained in §2.1, UEs must send the granted TBs within $K2$ slots, which sets a worst-case deadline of ~200 μs for this step (numerology $\mu = 2$ and $K2 = 0$). While simple, Algorithm 1 proves both efficient and extremely fast.

⑥ **LPU processing.** After $K2$ slots, the UE transmits the granted TB $g^{(i)}$. Once received, the DU forwards the TB to the AAL-B-UP's dispatcher, which routes the TB to the pre-assigned LPU queue for processing. Once decoded, the TB data is sent to the DU and the corresponding queue time estimator in the AAL-B-CP is updated. As explained in §2.1, this step must be completed within D.

4.2 Radio policy agent

CloudRIC's radio policy agent shall produce grants that are conscious of the current pressure on the LPUs and of the added load brought by the new requests. In turn, these properties depend on the specific hardware and software settings of the network environment (e.g., the number, type, and detailed specifications of the LPUs, which substantially affect the performance as we saw in §3.2), and are stateful (i.e., are contingent on the utilization of LPUs at the moment of the radio scheduling decision). Hence, they are hard to model in advance, and require a solution that automatically adapts itself to the target O-RAN system upon deployment.

In light of this consideration, we resort to a data-driven model that can learn the tangled relationships above at runtime. We implement the radio policy agent as a *soft actor-critic* deep reinforcement learning (RL) algorithm, under a twofold rationale: (*i*) the RL paradigm allows training the model online from system observations directly; and (*ii*) inference is achieved with a simple neural network (the actor) that involves trivial arithmetic operations to minimize inference latency, as later proven in §5. Among many actor-critic instances, we opted for a soft version that maximizes the reward while acting as randomly as possible to explore the solution space during training; this model achieves state-of-the-art performance in tasks similar to the one we tackle [25].

The instantaneous reward function used by the critic is

$$R^{(i)} = \begin{cases} t^{(i)}/\bar{t}^{(i)} & \text{if } w_n^{(i)} + d_n^{(i)} < D - \delta \\ -\beta \cdot & \text{otherwise.} \end{cases} \tag{1}$$

Two cases are possible. If TB $g^{(i)}$ is processed within a guard period δ to the deadline D, the decision is assigned a positive reward proportional to the granted fraction of the requested TB ($t^{(i)}/\bar{t}^{(i)}$). Otherwise, penalty β is assigned. For additional details on soft actor-critic models, we refer the reader to [25].

4.3 Logical Processing Units (LPU) models

The purpose of the LPU models $\{\nu_n, \mu_n\}$ is to estimate the processing latency, without queuing delays, and the energy

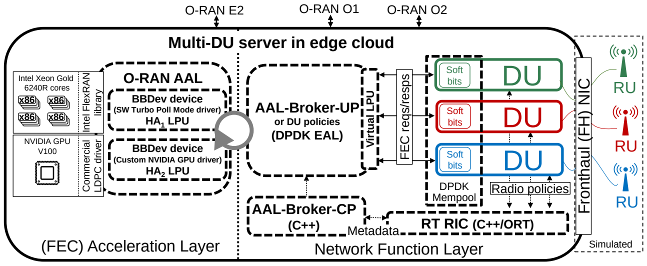

Figure 15: A CloudRIC-powered O-Cloud prototype.

consumption of each processor in the pool, for a given grant $g^{(i)}$. This is a stateless task, hence the LPU models can be easily built offline with data collected in a pre-calibration phase. Indeed, we take this approach in our implementation of CloudRIC, employing the measurement data presented in §3.2 to train LPU models for the specific LDPC drivers and GPU/CPU LPUs considered there.

We realize the LPU models as simple feed-forward neural networks, which are accurate and efficient universal function approximators. An important remark concerns the loss function used to train the models. For the energy estimator μ_n, we use a legacy mean absolute error (MAE) loss to minimize the absolute error. However, the loss for the processing latency estimator ν_n has a dedicated design that stems from the following consideration. Underestimating $\hat{d}_n^{(i)}$ risks violating deadlines, which incurs significant throughput loss and poor spectrum usage; conversely, overestimating $\hat{d}_n^{(i)}$ only implies far less disruptive radio resource under-allocations. Therefore, it is critical that the LPU model never underestimates latency, while still trying to minimize overestimation. Hence, we train ν_n with an asymmetric loss function that we call worst-case estimation time (WCET) loss. Let the latency prediction error for grant $g^{(i)}$ be $\epsilon_n^{(i)} := \hat{d}_n^{(i)} - d_n^{(i)}$, $\forall n \in \mathcal{L}$; the WCET loss is then expressed as

$$L(\epsilon_n^{(i)}) = \begin{cases} -\lambda \cdot \epsilon_n^{(i)} & \text{if } \epsilon_n^{(i)} \leq 0 \\ \epsilon_n^{(i)} & \text{if } \epsilon_n^{(i)} > 0, \end{cases} \tag{2}$$

where $\lambda > 1$ allows avoiding optimistic predictions.

5 IMPLEMENTATION

Fig. 15 illustrates our CloudRIC implementation in a multi-DU server, on top of the O-RAN AAL introduced in §3.4 with heterogeneous GPU and SIMD-capable 16-core CPU processors. As mentioned in §3.3, a server with a conventional PCIe v3.0+ bus and a 100-GbE fronthaul interface can aggregate up to 3.8 GHz of spectrum, supporting dense small-cell networks [54]. RUs and UEs are simulated following the bursty demand profiles presented in §3, which generate realistic workload fluctuations with varied levels of stress on our O-Cloud platform. To this end, we encode, modulate, add noise and demodulate signals locally.

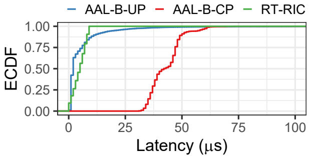

Figure 16: Overheads of our RT-RIC and AAL-B models.

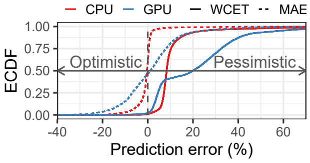

Figure 17: Prediction error of LPU models in AAL-B.

5.1 User Plane: AAL Broker

Our implementation of CloudRIC's AAL-B-UP (∼2K lines of C++) follows Fig. 14, and builds on top of the O-RAN AAL we developed in §3.4. We use the DPDK Environment Abstraction Layer (EAL) to abstract the complexity of thread and memory management of four main threads.

The 1st thread runs the AAL-B-UP, as well as its interfaces with the AAL-B-CP and the DUs using shared memory to reduce latency overhead. Upon each TB arrival, the AAL-B-UP routes its data to an LPU queue pre-assigned by the AAL-B-CP, as shown in Fig. 14. The 2nd and 3rd threads execute the CPU and GPU LPUs, respectively: they (*i*) gather bursts of TBs from the associated LPU queue; (*ii*) enqueue bursts in the LPU; (*iii*) execute the corresponding driver or library; and (*iv*) enqueue the decoded bits into an output queue. The last thread handles performance metrics.

Validation. To validate our implementation, we use all the profiles presented in §3.3 with a random LPU allocation, and measure the latency overhead introduced by our AAL-B-UP operations. Fig. 16 (blue line) shows the distribution of the overheads, with a median latency of 2 μs and a 99th percentile of 46 μs, which is negligible compared to the 1-3 ms time budget available for the LPU processing of each TB according to the summary in Table 2.

5.2 Control Plane: AAL-B-CP and RT-RIC

The AAL-B-CP and the RT-RIC are implemented in C++ (∼1.5K lines) using DPDK for efficient inter-process communication, and ONNX Runtime (ORT) for neural network acceleration [65] The AAL-B-CP uses four threads implementing, respectively, the main pipeline of §4.1, the management of queue state updates received from the AAL, the queue time estimator logic, and the LPU models. The RT-RIC uses a single thread to implement the radio policy agent.

The policy selects an action $r^{(i)} \in \{0, 0.1, \ldots, 0.9, 1.0\}$ for every TB i. Both actor and critic are neural networks with (5x140, 140x140, 140x11) and (16x140, 140x140, 140x11) layers, respectively, that are jointly trained using eq. (1) as reward, parameterized with $\delta = 10\%$ of the deadline D and $\beta = 10$, which proved to work well in all experimental scenarios considered in the study. For the target soft Q-function of the critic [25], we set a discount factor equal to 0.99.

The LPU models' neural networks are composed of three dense layers (3x128, 128x128, 128x1) and WCET loss in eq. (2)

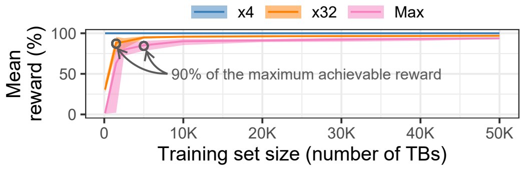

Figure 18: CloudRIC's mean (line) and max-min range (shaded area) reward for different training sets.

is parametrized with $\lambda = 10$. To illustrate the advantage of training the latency model v_n with our WCET approach, we used the dataset presented in §3.2 to show the distribution of the model's relative errors $\epsilon_n^{(i)}$ with solid lines in Fig. 17. As a benchmark, we also plot with dashed lines the relative error of an identical neural network trained to minimize the standard mean absolute error (MAE). What is relevant to CloudRIC operations is the fraction of overly optimistic estimates ($\epsilon_n^{(i)} < 0$) that lead to overrating the O-Cloud capacity and to queuing TBs that will later violate their deadline. Only 1.5% of the errors are optimistic for our method in contrast to the 53% of a MAE loss. Note that percent errors are higher for GPU than CPU, because the GPU latencies $d_n^{(i)}$ are considerably smaller, hence harder to estimate: the absolute error in μs is however comparable for the two processors.

Validation. As explained in §4, the overhead of the RT-RIC, which runs the the radio policy agent, and the AAL-B-CP, which runs the LPU models, have latency constraints indicated in Table 2. To assess that timing requirements are fulfilled by our implementation, we run CloudRIC for all the UE demand profiles presented in §3 and measure the overhead introduced by both modules. The distribution of the resulting latencies is depicted in Fig. 16. The RT-RIC's radio agent (green line) has a median and a 99th percentile latency of 5 μs and 9 μs, respectively. The LPU model neural networks incur instead into a median and a 99th percentile latency of 42 μs and 62 μs, respectively (red line). Both meet their budgets and validate CloudRIC's real-time operation.

6 EVALUATION

We next perform a thorough experimental campaign. First, we quantify the training overhead of CloudRIC (§6.1). Next, we validate that CloudRIC meets the design goals set in §4 in terms of reliability, throughput, and energy (§6.2). Lastly, we juxtapose CloudRIC against several benchmarks, scrutinizing aspects of reliability, energy- and cost-efficiency (§6.3).

6.1 Training time

CloudRIC hinges on data-driven approaches to realize the LPU models and radio policy agent. As explained in §4.1, the LPU models are stateless and can be pre-trained, e.g., using the dataset introduced in §3.2 in our case. Instead, the radio agent is trained online upon deployment in the target system, and the training time it requires to learn effective radio policing decisions is an important metric.

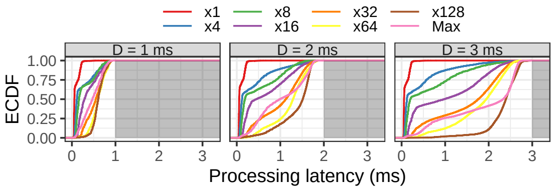

Figure 19: Processing latency distribution for different deadlines D = $\{1, 2, 3\}$ ms (shaded area).

Deadline (D)	x1	x4	x8	x16	x32	x64	x128	Max
3 ms	0.79	2.84	2.99	2.81	2.82	2.84	2.99	2.79
2 ms	0.80	1.85	1.88	1.92	1.90	1.95	1.99	1.97
1 ms	0.73	0.99	0.96	0.99	0.96	0.97	0.97	0.98

Table 3: 99.999th latency percentile (ms) achieved by CloudRIC for different deadlines D and demand profiles.

To analyze its training cost, we initialize RT-RIC's radio agent with random weights and study the evolution of its performance as it learns to schedule the traffic of 150 UEs distributed among thirty 100-MHz DUs. Fig. 18 presents the mean normalized reward achieved by CloudRIC after it has observed an increasing number of TBs across three processing deadlines $D = \{1, 2, 3\}$ ms commonly used in the literature [19, 21]. We present results for "x4", "x32", and "Max" UEs, as introduced in §3, to assess low, medium, and high workloads. CloudRIC reaches maximum reward almost instantaneously for small workloads ("x4"): the radio agent readily learns that it does not need to constrain the allocation of radio resources to guarantee deadlines in this case. With more demanding "x32" and "Max" users, CloudRIC takes up to 50K TBs to converge, and achieves 90% of that reward with 10 times less TBs. It shall be noted that 50K TBs correspond to less than 4 s in real-time with "x32" users, which demonstrates that the radio agent can learn good policies with minimal service degradation upon deployment.

6.2 CloudRIC validation

As introduced in §4, CloudRIC has three goals, prioritized as: (*i*) achieving 5-nines reliability, (*ii*) maximizing throughput, and (*iii*) minimizing energy use, which we evaluate next.

6.2.1 *Reliability.*

Compliance with the processing deadline during the inference phase directly controls the reliability of CloudRIC. We now better substantiate the dependability of our solution for real-time operation after the (very quick, as shown above) online training phase.

Fig. 19 shows the empirical distribution of the processing latency for all scenarios in §6.1. The shaded areas highlight deadline violations: by bounding all distributions to the left of those regions, CloudRIC meets the reliability target in all cases. More precisely, Table 3 depicts the latency's 99.999th percentiles, showing that CloudRIC consistently grants 5-nines reliability, which validates the first design goal. To achieve this, with growing loads and (to a minor extent) with more stringent deadlines, CloudRIC increases the average use of both CPU and HA as shown Fig. 20 (top).

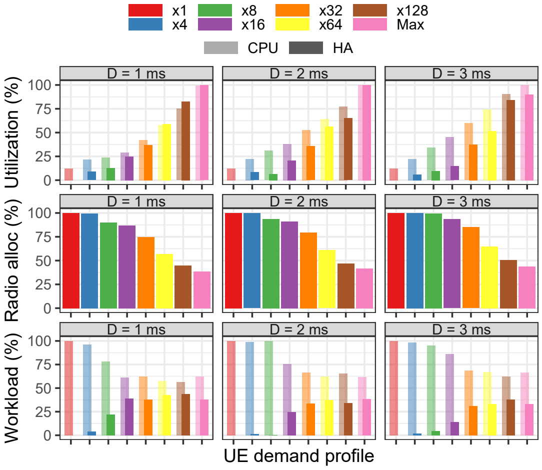

Figure 20: Mean resource utilization (top), radio allocation (middle), and workload allocation (bottom) for different deadlines D = $\{1, 2, 3\}$ ms and demand profiles.

However, when LPUs are under pressure, CloudRIC must bound the allocation of radio resources as depicted in Fig. 20 (middle). That is, in response to workloads that increasingly exceed the O-Cloud's capacity, the radio policy agent throttles down a progressively higher fraction of radio resources that otherwise would be wasted. Despite this, CloudRIC attains on average 98.7% of the achievable throughput (i.e., that of an offline exhaustive search with perfect knowledge).

It is worth noting that the results in Fig. 20 above are averages, while the UE traffic at the ms timescale is inherently bursty (as exemplified in Fig. 7). Therefore, CloudRIC must manage sudden upswings of requests (which saturate the LPUs and force radio grant limiting) alternating to low-demand periods (where all grants can be accommodated but resource utilization remains low). This explains why LPU usage inflation is fairly linear in the face of exponentially surging demands of "x1" through "Max" UEs, and why 100% radio allocations are not feasible even when the LPUs are not fully used, with "x8" through "x128" UEs.

6.2.2 Throughput and energy. Having assessed reliability, we now focus on the last design goals, i.e., exploiting computing heterogeneity to maximize throughput and energy efficiency. To that end, CloudRIC's LPU allocation function, informed by the LPU models, prioritizes energy-prudent processors as long as they meet deadlines but resorts to more powerful HAs when strictly required to maximize throughput. This yields a flexible load balancing, as shown in Fig. 20 (bottom): for instance, 100% of the workload is assigned to the CPU pool to save energy in the case of "x1" UEs, but around 40% is allotted to the HA to protect reliability in higher-loaded scenarios with "x32" through "Max" UEs.

To study this, we deploy CloudRIC on different AAL platforms: a CPU-only, a HA-only, and a heterogeneous platform combining CPUs and HAs. To provide a fair comparison across platforms, we sized each platform (number of CPU cores, HAs, etc.) such that additional computing resources of any type could not improve performance any further. We then emulate five DUs on each platform and vary the load as we did in §3.1 for $D = \{1, 2, 3\}$ ms.

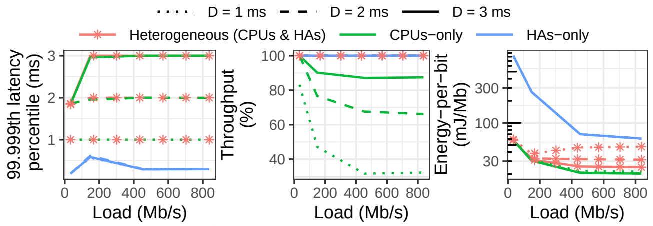

Figure 21: 99.999$^{\text{th}}$ latency percentile, throughput and energy consumption of CloudRIC on three different platforms.

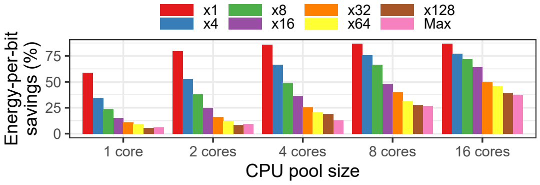

Figure 22: Energy savings of CloudRIC when using a CPU pool opportunistically relative to only using the HA.

In contrast to §3.1's results, Fig. 21 proves that CloudRIC meets the 99.999% reliability target in all cases. More importantly, the throughput (relative to the load) and energy performance figures illustrate the advantages of a heterogeneous platform: there, CloudRIC matches the throughput of an HA-only platform at a similar energy cost to that of a CPU-only system. In contrast, CloudRIC must sacrifice 15-75% of throughput in a CPU-only platform to ensure reliability, and consumes 3-14× more energy per bit in an HA-only system.

Indeed, the ability of CloudRIC to exploit CPUs opportunistically provides substantial gains in energy consumption. We analyze this in more detail in Fig. 22. The figure shows the energy cost savings achieved by CloudRIC on a heterogeneous platform with respect to the cost of exclusively relying upon a HA. The results refer to CPU pools of varied size, under all UE profiles. Remarkably, CloudRIC provides over 50% energy cost savings with just a single CPU core in presence of low-load "x1" UEs. Increasing the CPU pool size provides more savings, but with diminishing returns as many cores remain unused. For higher workloads, the energy cost savings are understandably lower as the LPU allocation function prioritizes network throughput and reliability over energy consumption. For instance, a single-core CPU pool provides practically no gain for the case of "Max" users but a 16-core pool attains 37% savings for the same scenario.

6.3 Comparison with other approaches

CloudRIC improves reliability and sustainability of vRANs over both best practices that the operators currently adopt in their deployments, as well as advanced benchmarks that only partially implement our design guidelines.

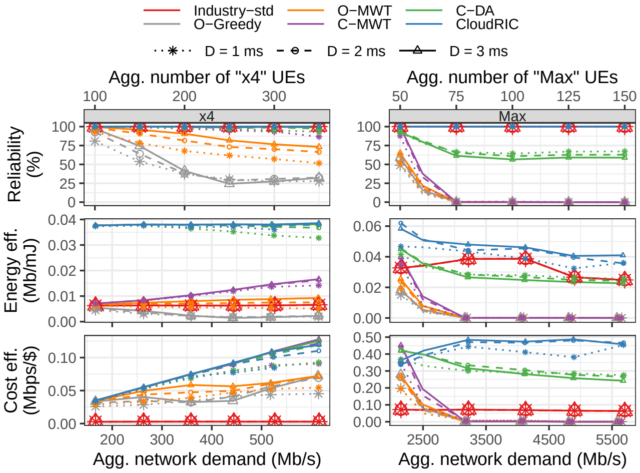

Figure 23: Reliability (top), energy-efficiency (middle), and cost-efficiency (bottom) for different policies on different scenarios with with "x4" (left) and "Max" (right) users.

6.3.1 Different benchmarks. We first demonstrate this by juxtaposing CloudRIC to the following baselines.

"Industry-std" uses a dedicated *in-line* HA co-located with every DU, which is *the conventional approach in the industry today.* To test this approach fairly, we simulate as many in-line HAs as DUs, each with the same performance of our NVIDIA V100 GPU, using the dataset presented in §3.2 and ignoring the look-aside overhead shown in Fig. 16 (blue line).

The other benchmarks are the policies introduced in §3.4: "O-Greedy", "O-MWT", "C-MWT", and "C-DA", which are not simulated anymore but implemented in our shared and heterogeneous AAL platform, like CloudRIC. In §3.4, several of these approaches relied on future knowledge, which is not feasible for real-time operation. To enable real-time operation, we use LPU models trained with a MAE loss (as explained in §4.3) to predict the information they require.

We run comparative experiments with a varying number of 100-MHz DUs aggregating 50 to 350 UEs in a single server. Fig. 23 depicts reliability (top), expressed as the percentage of TBs processed within D, energy efficiency (middle), calculated as the amount of data bits processed successfully per unit of energy), and cost efficiency (bottom), measured as the mean network throughput achieved per unit of capital investment[2]. Results are for $D = \{1, 2, 3\}$ ms and a variable number of "x4" (left) and "Max" (right) UEs, hence DUs.

The first key remark is that only CloudRIC and "Industry-std" provide high reliability across all scenarios. Precise figures on this critical aspect for industry-grade vRANs are presented in Table 3: the 99.999th percentile of the processing latency confirms that "Industry-std" is the only benchmark to achieve the same 5-nine reliability of CloudRIC.

UEs profile	Deadline (D)	CloudRIC	Industry-std	O-Greedy	O-MWT	C-MWT	C-DA
"x4"	3 ms	2.94	0.20	28.5	10.9	6.73	6.59
	2 ms	1.95	0.20	23.4	9.83	7.51	4.72
	1 ms	0.99	0.29	13.9	9.26	9.06	2.92
"Max"	3 ms	2.95	0.29	35.4	35.5	35.4	6.15
	2 ms	1.99	0.29	24.1	24.1	24.1	4.22
	1 ms	0.99	0.29	12.9	12.9	13.0	2.40

Table 4: Comparison of the 99.999th latency percentile (ms) of different approaches for the scenarios presented in Fig. 23.

However, "Industry-std" matches the dependability of CloudRIC by largely over-dimensioning the computing capacity, which yields the worst cost-efficiency performance. Fig. 23 demonstrates how CloudRIC makes the best use of a much smaller, shared, and heterogeneous O-Cloud to achieve 15x higher cost efficiency and 3x higher energy efficiency, on average, than the legacy industry practice.

The capital cost of all the other benchmarks is the same as CloudRIC's so its cost-efficiency gains are solely due to network throughput gains. Because "C-DA" uses CloudRIC's AAL-B, it achieves similar performance to CloudRIC for small workloads; however, the absence of the RT-RIC causes "C-DA" to waste radio and computing resources on TBs that miss their deadlines during heavier workloads, which results in 34% and 31% lower energy efficiency and throughput, respectively. "O-Greedy", "O-MWT", and "C-MWT" all confirm the poor reliability observed in §3.4 on a real-world platform, with throughput dropping to 0% for "Max" UEs.

6.3.2 Different Hardware Accelerators. Finally, we analyze the impact of different HAs on the performance gains of CloudRIC over "Industry-std", i.e., the only two approaches that meet the reliability target of 99.999% at all times.

To this end, we scale the latency and the power consumption measurements of the NVIDIA V100 GPU by factors $0.2 \leq \alpha_{\text{lat}} \leq 1$ and $0.2 \leq \alpha_{\text{pwr}} \leq 1$, respectively. Thus, different combinations of α_{lat} and α_{pwr} simulate diverse and future HAs. For instance, based on publicly available information, we roughly estimate that the look-aside Intel ACC100 HA may be modeled with $0.5 \lesssim \alpha_{\text{pwr}} \lesssim 0.7$, and the look-aside Intel ACC200 (integrated into the new Sapphire processors) or the in-line Qualcomm X100 HA with $0.6 \lesssim \alpha_{\text{lat}} \lesssim 0.8$.

To study different scenarios, we generate various workloads by emulating different numbers of DUs, each with five "x4" or "Max" UEs. Like before, one heterogeneous platform is shared among all the DUs in the case of CloudRIC, while we simulate a HA-only server per DU for "Industry-std".

On the one hand, Fig. 24 depicts the energy efficiency of CloudRIC relative to that of "Industry-std". For small-medium workloads (aggregating <500 "x4" UEs), the heterogeneous platform enables CloudRIC to achieve substantial energy efficiency gains—up to 6× the efficiency of "Industry-std"—when the HA satisfies $\alpha_{\text{pwr}} + \alpha_{\text{lat}} \geq 1$. As mentioned above, cutting-edge HAs today are well within this range. When $\alpha_{\text{pwr}} + \alpha_{\text{lat}} < 1$, perhaps achievable by future HA technology, the HA's latency and energy consumption is so low

[2] To calculate energy- and cost-efficiency, we ignore the baseline energy and financial cost of the servers, and use the representative market prices of Table 1, i.e., $110 per CPU core, and $3,000 per HA, making each HA-only "Industry-std" server 37% cheaper than the shared heterogeneous server.

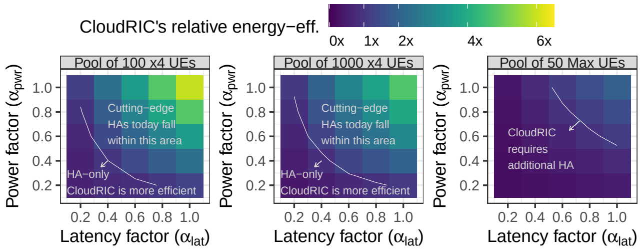

Figure 24: CloudRIC’s energy-efficiency with respect to Industry-std’s for different HAs characterized by parameters α_{lat} and α_{pwr}, which scale the power consumption and latency performance, respectively, of an NVIDIA GPU V100.

compared to today’s CPUs that using the latter becomes inefficient. This occurs in a larger (α_{pwr}, α_{lat}) area for very large workloads (>750 “x4” or >10 “Max” UEs), where CloudRIC must throttle down radio resources to preserve reliability. Note that, in such extreme cases, CloudRIC may share more HAs (instead of CPUs) to boost energy efficiency.

On the other hand, Fig. 25 presents the cost efficiency of CloudRIC relative to that of “Industry-std”. Since CloudRIC aggregates all the workload into a single server, its cost-efficiency gain over “Industry-std” grows linearly with the number of DUs. This is depicted in Fig. 25 for different workloads, generated in the same way as before, and for different α_{lat} values (note that cost-efficiency is independent of α_{pwr} as it does not impact on throughput). The growth rate degrades when the computing resources available in CloudRIC’s server are maxed out and CloudRIC has to sacrifice throughput to preserve reliability (e.g., >310 “x4” or >50 “Max” UEs). This is more severe for higher α_{lat} because reducing α_{lat} essentially increases the HA’s computing capacity. Moreover, since cost-efficiency also depends on the number of DUs per server, high-loaded DUs with “Max” UEs incur lower gains as large workloads are generated with fewer DUs.

7 RELATED WORK

Most task scheduling frameworks [37, 57, 58] operate at coarse time granularity, which is not suitable for vRAN workloads. Shinjuku [34], Shenango [53], and Snap [43] can provide μs-level tail latencies; however they are generic approaches alien to the specifics of radio scheduling operations and hence cannot be used for sharing heterogeneous HAs while providing many-9s reliability in dense vRAN platforms.

Agora [12] is a 5G data channel processor built on top of FlexRAN’s publicly available libraries for many-core general-purpose CPU platforms, and does not require other HAs such as ASICs, FPGAs or GPUs. However, Agora is not suitable for shared computing platforms as it relies upon dedicated CPU cores to provide deterministic computing latency. More recently, Concordia [19] proposed a CPU scheduler to jointly execute 5G processing tasks and other latency-elastic applications. However, Concordia (*i*) does not police radio resources, and hence cannot ensure reliability in platforms shared by multiple DUs, and (*ii*) cannot manage heterogeneous pools of HAs and CPUs. Nuberu [21] is a DU that trades-off network latency for reliability in shared computing platforms. While the use of reliable DUs is important, Nuberu neither coordinates multiple concurrent DUs nor optimizes the use of computing resources. Additional related works are GPF [29], an extremely fast (~100μs) GPU-accelerated radio resource scheduler, and [44], an approach to share RU front-ends. Neither addresses the specific problem we pose in this paper of joint control of computing and radio resources.

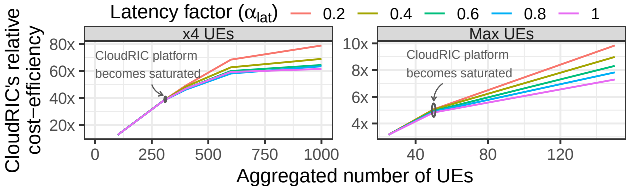

Figure 25: CloudRIC’s cost-efficiency with respect to Industry-std’s for different HAs characterized by α_{lat}.

Orthogonal yet related to our work is the work on O-RAN orchestration, e.g., [5–7] use machine learning to optimize the system in long timescales. ColO-RAN [56] is a large-scale testbed with software-defined radios-in-the-loop to evaluate O-RAN RIC algorithms. Finally, OrchestRAN [14] orchestrates data-driven models to provide intent-based control operations for mobile operators. However, as shown in §3 real-world workloads are bursty at small timescales. Therefore, orchestration solutions, which operate at second- or minute-timescales, cannot attain CloudRIC’s multiplexing gains while concurrently meeting hard reliability constraints.

8 CONCLUSIONS

In this paper, we introduce CloudRIC, a system that addresses the limitations of current vRANs that rely on expensive, energy-intensive Hardware Accelerators (HAs) co-located with each Distributed Unit (DU) to meet stringent reliability requirements. Extending O-RAN’s cloud architecture, CloudRIC enables the sharing of HAs across multiple DUs to reduce costs and utilizes low-cost CPUs to reduce energy consumption, without compromising reliability. Our implementation, based on DPDK’s BBDev framework and light data-driven models, shows that CloudRIC incurs low overhead and delivers, on average, 3x and 15x gains in energy- and cost-efficiency, respectively, compared to the industry standard approach in real-world and worst-case scenarios.

ACKNOWLEDGMENTS

Work supported by the EC via grants no. 101017109 (DAEMON) and 101139270 (ORIGAMI), by NextGeneration EU through UNICO I+D grants no. TSI-063000-2021 (OPEN6G) and 022/0005395 (CLARION), and by CERCA Programme.

REFERENCES

[1] 3GPP. 2020. 5G;NR; Physical layer procedures for control. 3GPP TS 38.213 version 16.2.0 Release 16.

[2] 3rd Generation Partnership Project (3GPP). 2021. 3GPP TS 38.214; Technical Specification Group Radio Access Network; NR; Physical layer procedures for data (Release 16). Technical Specification.

[3] 3rd Generation Partnership Project (3GPP). 2022. 3GPP TR 38.913; Technical Specification Group Radio Access Network; Study on Scenarios and Requirements for Next Generation Access Technologies; (Release 17). Technical Report.

[4] Analysys Mason. 2023. Open RAN: translating the hype into revenue. Webinar.

[5] Jose A. Ayala-Romero et al. 2020. vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs. *IEEE Transactions on Mobile Computing* (2020), 1–1. https://doi.org/10.1109/TMC.2020.3043100

[6] Jose A. Ayala-Romero et al. 2021. Bayesian Online Learning for Energy-Aware Resource Orchestration in Virtualized RANs. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM42981.2021.9488845

[7] Jose A. Ayala-Romero et al. 2021. *EdgeBOL: Automating Energy-Savings for Mobile Edge AI*. Association for Computing Machinery, New York, NY, USA, 397–410. https://doi.org/10.1145/3485983.3494849

[8] Yufei Blankenship, Dennis Hui, and Mattias Andersson. 2021. *Channel Coding in NR*. Springer International Publishing, Cham, 303–332. https://doi.org/10.1007/978-3-030-58197-8_10

[9] Aleksandra Checko, Henrik L Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S Berger, and Lars Dittmann. 2014. Cloud RAN for mobile networks—A technology overview. *IEEE Communications surveys & tutorials* 17, 1 (2014), 405–426.

[10] Erik Dahlman, Stefan Parkvall, and Johan Skold. 2020. *5G NR: The next generation wireless access technology*. Academic Press.

[11] Deloitte. 2021. The Open Future of Radio Access Networks. *Editorial Report* (2021).

[12] Jian Ding, Rahman Doost-Mohammady, Anuj Kalia, and Lin Zhong. 2020. Agora: Real-time massive MIMO baseband processing in software. In *Proceedings of ACM CoNEXT '20*. ACM.

[13] NTT Docomo. 2016. Base-station Equipment with the Aim of Introducing 3.5-GHz band TD-LTE. *NTT Docomo Technical Journal* (2016).

[14] Salvatore D'Oro, Leonardo Bonati, Michele Polese, and Tommaso Melodia. 2022. OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 270–279. https://doi.org/10.1109/INFOCOM48880.2022.9796744

[15] DPDK. 2023. *Wireless Baseband Device Library*. https://doc.dpdk.org/guides/prog_guide/bbdev.html

[16] Ericsson. 2022. Mobility Report. *White Paper* (June 2022).

[17] Gabriel Falcao, Leonel Sousa, and Vitor Silva. 2010. Massively LDPC decoding on multicore architectures. *IEEE Transactions on Parallel and Distributed Systems* 22, 2 (2010), 309–322.

[18] Robert Falkenberg and Christian Wietfeld. 2019. FALCON: An Accurate Real-time Monitor for Client-based Mobile Network Data Analytics. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Waikoloa, Hawaii, USA. https://doi.org/10.1109/GLOBECOM38437.2019.9014096 arXiv:1907.10110

[19] Xenofon Foukas and Bozidar Radunovic. 2021. Concordia: Teaching the 5G VRAN to Share Compute. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 580–596. https://doi.org/10.1145/3452296.3472894

[20] Fujitsu. 2023. What is the difference between inline and lookaside accelerators in virtualized distributed units? *White Paper* (2023).

[21] Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Pablo Serrano, and Albert Banchs. 2021. Nuberu: Reliable RAN Virtualization in Shared Platforms. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking* (New Orleans, Louisiana) *(MobiCom '21)*. Association for Computing Machinery, New York, NY, USA, 749–761. https://doi.org/10.1145/3447993.3483266

[22] Andres Garcia-Saavedra and Xavier Costa-Pérez. 2021. O-RAN: Disrupting the Virtualized RAN Ecosystem. *IEEE Communications Standards Magazine* 5, 4 (2021), 96–103. https://doi.org/10.1109/MCOMSTD.101.2000014

[23] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. 2016. srsLTE: an open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. 25–32.

[24] Junzhi Gong, Anuj Kalia, and Minlan Yu. 2023. Scalable Distributed Massive MIMO Baseband Processing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 405–417.

[25] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1861–1870. https://proceedings.mlr.press/v80/haarnoja18b.html

[26] Wang Tsu Han and Raymond Knopp. 2018. OpenAirInterface: A pipeline structure for 5G. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. IEEE, 1–4.

[27] Heavy Reading. 2021. Heavy Reading's Accelerating Open RAN Platforms Operator Survey. *White Paper* (June 2021).

[28] Heavy Reading. 2022. 5G Transport: A 2021 Heavy Reading Survey. *White Paper* (Feb. 2022).

[29] Yan Huang, Shaoran Li, Y. Thomas Hou, and Wenjing Lou. 2018. GPF: A GPU-Based Design to Achieve 100 μs Scheduling for 5G NR. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 207–222. https://doi.org/10.1145/3241539.3241552

[30] Intel. 2019. *FlexRAN LTE and 5G NR FEC Software Development Kit Modules*. https://software.intel.com/content/www/us/en/develop/articles/flexran-lte-and-5g-nr-fec-software-development-kit-modules.html

[31] Intel. 2020. *Enabling 5G Wireless Acceleration in FlexRAN: for the Intel® FPGA Programmable Acceleration Card N3000*. https://www.intel.com/content/www/us/en/programmable/documentation/ocl1575542673666.html

[32] Intel. 2023. *4th Gen Intel Xeon Scalable Processors with Intel vRAN Boost. Enabling high performance, energy-efficient vRAN with fully integrated acceleration.* https://download.intel.com/newsroom/2023/5g-communications/2023MWC-vRAN-Fact-Sheet.pdf

[33] Chaoqiang Jin, Xuelian Bai, Chao Yang, Wangxin Mao, and Xin Xu. 2020. A review of power consumption models of servers in data centers. *Applied Energy* 265 (2020), 114806. https://doi.org/10.1016/j.apenergy.2020.114806

[34] Kostis Kaffes, Timothy Chong, Jack Tigar Humphries, Adam Belay, David Mazières, and Christos Kozyrakis. 2019. Shinjuku: Preemptive Scheduling for {μsecond-scale} Tail Latency. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 345–360.

[35] Keith Dyer. 2023. *AI everywhere all the time*. https://the-mobile-network.com/2023/03/ai-everywhere-all-the-time/

[36] Anupa Kelkar and Chris Dick. 2021. NVIDIA aerial GPU hosted AI-on-5G. In *2021 IEEE 4th 5G World Forum (5GWF)*. IEEE, 64–69.

[37] Jing Li, David Ferry, Shaurya Ahuja, Kunal Agrawal, Christopher Gill, and Chenyang Lu. 2017. Mixed-criticality federated scheduling for parallel real-time tasks. *Real-time systems* 53, 5 (2017), 760–811.

[38] Light Reading. 2022. *Mavenir unhappy about chip prices for smaller open RAN players.* https://www.lightreading.com/open-ran/mavenir-unhappy-about-chip-prices-for-smaller-open-ran-players/d/d-id/781327

[39] Light Reading. 2023. *Chip choices kickstart open RAN war between lookaside and inline.* https://www.lightreading.com/semiconductors/chip-choices-kickstart-open-ran-war-between-lookaside-and-inline

[40] Yeon-sup Lim, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. 2017. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies* (Incheon, Republic of Korea) *(CoNEXT '17)*. Association for Computing Machinery, New York, NY, USA, 147–159. https://doi.org/10.1145/3143361.3143376

[41] Luca Cominardi. 2021. *Zenoh performance: a stroll in Rust async wonderland.* https://zenoh.io/blog/2021-07-13-zenoh-performance-async/

[42] N. Ludant, P. Robyns, and G. Noubir. 2023. From 5G Sniffing to Harvesting Leakages of Privacy-Preserving Messengers. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 3146–3161. https://doi.org/10.1109/SP46215.2023.00110

[43] Michael Marty, Marc de Kruijf, Jacob Adriaens, Christopher Alfeld, Sean Bauer, Carlo Contavalli, Michael Dalton, Nandita Dukkipati, William C Evans, Steve Gribble, et al. 2019. Snap: A microkernel approach to host networking. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 399–413.

[44] Jose Mendes, XianJun Jiao, Andres Garcia-Saavedra, Felipe Huici, and Ingrid Moerman. 2019. Cellular access multi-tenancy through small-cell virtualization and common RF front-end sharing. *Computer Communications* 133 (2019), 59–66. https://doi.org/10.1016/j.comcom.2018.10.010

[45] Diksha Moolchandani, Anshul Kumar, and Smruti R. Sarangi. 2021. Accelerating CNN Inference on ASICs: A Survey. *Journal of Systems Architecture* 113 (2021), 101887. https://doi.org/10.1016/j.sysarc.2020.101887

[46] Fahri Wisnu Murti, Jose A Ayala-Romero, Andres Garcia-Saavedra, Xavier Costa-Pérez, and George Iosifidis. 2020. An optimal deployment framework for multi-cloud virtualized radio access networks. *IEEE Transactions on Wireless Communications* 20, 4 (2020), 2251–2265.

[47] NTT Docomo. 2021. 5G Open RAN Ecosystem. *White Paper* (June 2021).

[48] NVIDIA. 2022. *NVIDIA V100 Tensor Core GPU Specifications.* https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf

[49] O-RAN Alliance. 2022. Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN (O-RAN.WG6.CADS-v04.00). Technical Report.

[50] O-RAN Alliance. 2022. *O-DU Low Project Introduction.* https://docs.o-ran-sc.org/projects/o-ran-sc-o-du-phy/en/latest/overview1.html

[51] O-RAN Alliance. 2022. O-RAN Acceleration Abstraction Layer – General Aspects and Principles (O-RAN.WG6.AAL-GAnP-v04.00). Technical Specification.

[52] O-RAN Alliance. 2022. O-RAN Working Group 4 (Open Fronthaul Interfaces WG). Control, User and Synchronization Plane Specification (O-RAN.WG4.CUS.0-v10.00). Technical Report.

[53] Amy Ousterhout, Joshua Fried, Jonathan Behrens, Adam Belay, and Hari Balakrishnan. 2019. Shenango: Achieving high {CPU} efficiency for latency-sensitive datacenter workloads. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 361–378.

[54] Gabriel Otero Pérez, David Larrabeiti Lopez, and José Alberto Hernández. 2019. 5G new radio fronthaul network design for eCPRI-IEEE 802.1 CM and extreme latency percentiles. *IEEE Access* 7 (2019), 82218–82230.

[55] Pablo Fernández Pérez, Claudio Fiandrino, and Joerg Widmer. 2023. Characterizing and Modeling Mobile Networks User Traffic at Millisecond Level. In *Proceedings of the 17th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization* (Madrid, Spain) *(WiNTECH '23)*. Association for Computing Machinery, New York, NY, USA, 64–71. https://doi.org/10.1145/3615453.3616509

[56] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. 2022. ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms. *IEEE Transactions on Mobile Computing* (2022), 1–14. https://doi.org/10.1109/TMC.2022.3188013

[57] George Prekas, Marios Kogias, and Edouard Bugnion. 2017. Zygos: Achieving low tail latency for microsecond-scale networked tasks. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 325–341.

[58] Henry Qin, Qian Li, Jacqueline Speiser, Peter Kraft, and John Ousterhout. 2018. Arachne:{Core-Aware} Thread Management. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 145–160.

[59] RCR Wireless News. 2021. From greenfield to brownfield: Open RAN in 2022 (With large scale carrier commitments in place, what's next for the Open RAN ecosystem?). *Editorial Report* (October 2021).

[60] Rethink Technology Research. 2019. *Nokia says its FPGA strategy hit 5G margins, but other factors are at work too.* https://rethinkresearch.biz/articles/nokia-says-its-fpga-strategy-hit-5g-margins-but-other-factors-are-at-work-too/

[61] Rethink Technology Research. 2020. *Is general purpose silicon too slow and expensive for the vRAN?* https://rethinkresearch.biz/articles/is-general-purpose-silicon-too-slow-and-expensive-for-the-vran/

[62] Silicom. 2022. *Silicom's eASIC ACC100 FEC Accelerator.* https://www.silicom-usa.com/wp-content/uploads/2022/10/Lisbon-ACC100-FEC-Accelerator-Extended-temp-Server-Adapter.pdf

[63] Soma Velayutham. 2019. *NVIDIA CEO Introduces Aerial — Software to Accelerate 5G on NVIDIA GPUs.* https://blogs.nvidia.com/blog/2019/10/21/aerial-application-framework-5g-networks/

[64] Aleksejs Udalcovs, Marco Levantesi, Patryk Urban, Darli AA Mello, Roberto Gaudino, Oskars Ozolins, and Paolo Monti. 2020. Total cost of ownership of digital vs. analog radio-over-fiber architectures for 5G fronthauling. *IEEE Access* 8 (2020), 223562–223573.

[65] Pedro Vicente, Pedro M Santos, Barikisu Asulba, Nuno Martins, Joana Sousa, and Luis Almeida. 2023. Comparing Performance of Machine Learning Tools Across Computing Platforms. In *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 1185–1189.