Distributed AI Platform for the 6G RAN

Ganesh Ananthanarayanan, Xenofon Foukas, Božidar Radunović, Yongguang Zhang
Microsoft Corp.
{ga, xefouk, bozidar, ygz}@microsoft.com

***Abstract*—Cellular Radio Access Networks (RANs) are rapidly evolving towards 6G, driven by the need to reduce costs and introduce new revenue streams for operators and enterprises. In this context, AI emerges as a key enabler in solving complex RAN problems spanning both the management and application domains. Unfortunately, and despite the undeniable promise of AI, several practical challenges still remain, hindering the widespread adoption of AI applications in the RAN space. This article attempts to shed light to these challenges and argues that existing approaches in addressing them are inadequate for realizing the vision of a truly AI-native 6G network. Motivated by this lack of solutions, it proposes a generic distributed AI platform architecture, tailored to the needs of an AI-native RAN and discusses its alignment with ongoing standardization efforts.**

***Index Terms*—6G RAN, Distributed AI, Radio Access Network, vRAN, AI RAN.**

I. Introduction

Cellular Radio Access Networks (RANs) are undergoing a paradigm shift. The main driver behind this evolution is the reduction of the high CapEx and OpEx that telco operators are faced with and the introduction of new revenue streams. This transformation started with the launch of 5G, which introduced several key changes in the way that the network is being deployed and managed. As illustrated in Fig. 1, it turned the monolithic base stations into disaggregated and virtualized components – Central Unit (CU), Distributed Unit (DU) and Radio Unit (RU) – that can be deployed on top of commodity hardware across several locations (far edge, near edge, cloud), simplifying their lifecycle management and release of new features. Furthermore, it introduced open and programmable interfaces, allowing the deployment of applications on top of RAN Intelligent Controllers (RICs), to further accelerate innovation. Finally, it introduced new radio access technologies, like Massive MIMO and millimeter waves, and pushed the densification of the network to new limits, with the goal of expanding the network capacity and enabling new types of applications (e.g., IoT).

While the shift to 5G has so far served as a transitional phase, the emerging 5G Advanced and 6G networks are expected to truly unlock the networks' potential in two ways. First, by developing sophisticated control and management solutions that can tackle a plethora of long-withstanding network problems (e.g., in the context of radio resource management, mobility, energy savings, etc.), which have been further amplified by the added complexity of 5G. Second, by introducing new transformative applications (e.g., joint communication-sensing, security, slicing), that can add value and provide differentiation for operators and verticals.

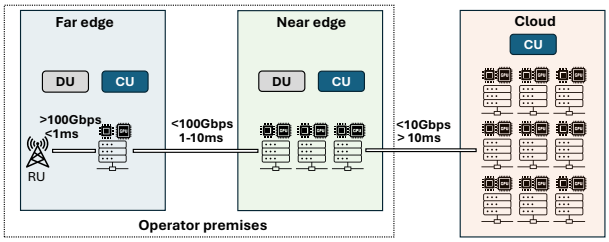

Fig. 1. **High-level overview of the 5G RAN. The virtualized RAN infrastructure capabilities can vary depending on the location.**

In this context, AI is expected to play a pivotal role. The recent advances in the domain of generative AI have acted as a catalyst in drawing the attention of the telecommunications industry to the benefits of AI in solving the aforementioned problems. AI is an ideal fit for many long standing RAN problems, which involve pattern recognition of signals for classification, predicting traffic and devising solutions for computationally intractable problems, such as scheduling. In fact, the influence of AI is becoming so big in the telco space, that the general consensus is that next generation mobile networks should be AI-native, with both the industry and academia rallying behind initiatives like the AI-RAN Alliance [1].

Unfortunately, while the promise of AI in the RAN is undoubtedly high, its potential still remains untapped, due to several practical roadblocks. First, the high accuracy of AI models is contingent to the existence of rich data sources. Given that RAN base stations are distributed across tens of thousands of locations, one would have to deal with several problems related to the collection and transmission of vast amounts of data. Second, RAN applications leveraging AI models present heterogeneous characteristics in terms of their compute requirements, response latency and privacy constraints. When operating over a distributed hierarchy of edges and the cloud, the varied compute availabilities and network bandwidths pose a significant challenge for operational deployments and handling AI RAN application constraints is far from straightforward.

Motivated by these observations, in this article, we attempt to shed light to the challenges of designing and deploying AI models for the RAN. We argue that existing approaches, which rely on the static data collection and orchestration of AI models are not suitable to meet the demands of AI-native 6G networks. Building on these observations, we present our vision of a distributed AI architecture that is tailored to the needs and requirements of an AI-native RAN, and we outline our thoughts of how our proposed architecture can be aligned with the ongoing standardization efforts.

II. The Need for AI in RAN

Here, we discuss why the need for AI in the RAN space is becoming increasingly important. The AI-RAN Alliance has succinctly grouped AI use cases in three key domains [1]:

1) AI-for-RAN – This refers to utilizing AI to optimize and enhance the RAN performance, motivated by the need to use the limited spectrum efficiently. The demand for more traffic has led to the introduction of new, higher frequency bands. These bands offer more capacity but lower coverage, leading to an increase in the overall tower density. Furthermore, new radio resource allocation technologies, like Massive MIMO, see the use of large antenna arrays as a way to further increase the cell capacity. Assigning users to different bands and managing interference across cells by adjusting scheduling, power control and antenna assignments leads to a search space explosion, which is computationally intractable to manage with conventional optimization methods, making the use of AI a compelling alternative [2].

Similar to radio resource optimization, there exist a whole set of infrastructure optimization problems, caused by the increased 5G complexity, which AI promises to solve. Telco operators often manage close to 100k RAN sites. Predictive maintenance for an infrastructure at that scale (e.g., dealing with hardware issues, software bugs, etc.) is a challenge on its own. The virtualization of the RAN adds an extra layer of complexity, making the identification and root cause analysis of performance issues far from obvious [3]. Finally, in the context of energy efficiency, a lot of effort has been on how to reduce the power consumption of various elements, while maintaining the network coverage and quality, with several AI-based research works showing promising results [4].

2) AI-and-RAN – This domain focuses on the compute sharing (e.g., CPU, GPU) between RAN network functions and AI applications, for efficient utilization of the infrastructure. Typically, virtualized RAN functions utilize the infrastructure lightly (typically $< 50\%$ utilization) [5], for reasons inherent to the characteristics of the RAN workload. vRAN servers have to be provisioned for peak capacity, but they typically serve significantly less traffic (e.g., due to the diurnal traffic patterns). This inefficiency persists even at times of peak usage due to imbalances in the computational demand when processing uplink and downlink traffic [6].

In contrast to cloud computing, where the sharing of compute resources is commonplace, sharing resources with RAN workloads introduces significant challenges. The real-time nature of the RAN means that sharing can lead to deadline violations and adversely affect network performance. Several recent proposals have demonstrated how AI can be leveraged to share the infrastructure in both CPUs [6] and GPUs [7].

3) AI-on-RAN – This domain encompasses use cases that leverage the RAN infrastructure to support AI applications. A key enabler of such use cases are the open interfaces exposed by the RAN functions, which can allow third-parties to tap into RAN data and/or affect RAN traffic-related decisions, in order to enhance the application capabilities. One example that falls under this category is localization via sensing. By leveraging channel state information exposed by the physical layer of the RAN, one can perform user localization, which can be important for several use cases, like asset tracking and self-driving cars [8]. AI-based techniques leveraging physical layer information can also be employed to provide context-aware security [9], in consideration of battery-constrained devices. Finally, more traditional AI-based applications, like video analytics could also leverage the interplay with the RAN, to accommodate latency critical use-cases (e.g., [10]).

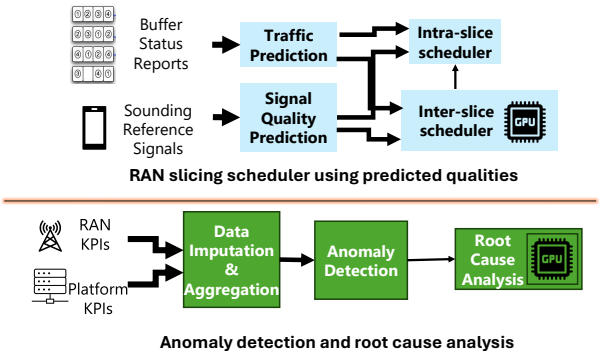

Fig. 2. **Graphs of AI/ML models for RAN applications**

III. Background

A. Canonical application examples

To help with highlighting the need for a distributed AI RAN platform, we define two AI application examples (Fig. 2), which we use as a reference for the remainder of this article.

RAN slicing scheduler – A scheduler allocates radio resources across network slices (inter-slice scheduling) and across users of the same slice (intra-slice scheduling). Both schedulers use the traffic demand information provided by the phones through buffer status reports, to try and predict the traffic load for the upcoming period [11]. They also use physical layer sounding reference signals to predict the users' signal quality. The inter-slice scheduler makes scheduling decisions at coarse granularities (e.g., seconds) and feeds its decisions to the real-time intra-slice scheduler. This example illustrates both AI-on-RAN (predicting load using buffer status reports) as well as AI-for-RAN (scheduling decisions).

Anomaly detection and root cause analysis – A service management and orchestration framework tries to detect RAN-related anomalies and identify their root cause for potential mitigations towards AI-for-RAN. Since the anomalies can originate both from the RAN and the platform, the application collects data from both sources [3]. It also applies imputation and aggregation techniques, to deal with noisy/missing data and to reduce their volume. It then processes the collected statistics, to detect if an anomaly is present, and if so, it attempts to localize it and detect its root cause.

From these examples, we see that a typical AI RAN application, rather than being a monolith, may consist of a sequence of blocks. Each block can do its own data pre-processing and can include an ML model in itself, with substantial research and development driving its release. Those blocks are chained together into a graph (e.g., by solution providers), where the

output of one model (e.g., mobility predictor) becomes the input of the next (e.g., resource block scheduler). Typically the traffic volume between components reduces from left to right, as more data gets processed and aggregated (signified with narrower arrows in the examples).

B. Distributed edge infrastructure

The RAN infrastructure is typically deployed across a distribution of far and near edges, extending to the cloud (Fig.1). As we move from the far edge to the cloud, more compute resources become available for the AI RAN applications. However, choosing the cloud for deployment (e.g., due to the presence of GPUs) is not always the right choice. The abundance of resources comes at the expense of reduced bandwidth and higher latency. This can be crucial factors for applications such as the example anomaly detector, which requires large volumes of input data, or the RAN slicing scheduler, which is latency-sensitive in its allocation decisions.

IV. Challenges to build AI RAN applications

We now discuss the challenges of deploying AI RAN applications, that make the case for a distributed AI RAN platform.

A. Data collection

Different applications require different feature sets, that might have heterogeneous characteristics in terms of type, time granularity, etc. For example, the radio resource scheduling application might require real-time data from the RAN network functions, while the anomaly detection might need to combine aggregate data from both the RAN and the platform in time windows of seconds. Similarly, the anomaly detection application might rely on the inter-arrival time between IQ samples, while the scheduler might require the actual raw IQ samples carrying the sounding reference signals.

The heterogeneity in the input features of AI applications is what makes the data collection process challenging. Exposing raw data from all possible data sources is not a viable option, as it would lead to a huge volume of data that one would have to process, store and transmit. For example, capturing all the raw IQ samples from the physical layer of the RAN translates into several gigabits of traffic per second, even for a single base station with four antennas. Similarly, capturing all the CPU scheduling events of a server, in order to detect if a platform anomaly due to CPU interference is present, would result in the collection of millions of events per second.

The current standard practice to bypass this roadblock is to expose a set of coarse-grained data sources, that are applicable to a wide range of use cases. These data sources are exposed through static APIs specified by standardization bodies like 3GPP and O-RAN. For instance, O-RAN defines the E2 interface for the collection of pre-defined RAN KPIs in the form of so-called static "service models", and the O2 interface for the collection of monitoring data from the platform.Any change to a data source (e.g. reporting the 90th percentile of packet latency instead of max), requires the standardization body consensus.This can be a long process that is typically met with skepticism, due to performance concerns that might arise when modifying mission critical RAN and platform code.

In conclusion, data collection challenges force AI RAN applications to be developed and deployed as an afterthought, subject to the available data sources, rather than in a truly AI-native way, where the data collection is application-driven.

B. RAN AI application orchestration

The disaggregated nature of the RAN means that its network functions might be deployed across the edges and the cloud. This raises a fundamental question as to what should be the location in which the blocks of AI applications should reside. Answering this question is far from straightforward. It involves matching the applications' requirements to the capabilities of the infrastructure, which can vary in terms of compute resources, network bandwidth, latency, etc, depending on the location (Fig. 1). The placement decision could also be affected by other factors, such as privacy concerns, since operators might have a preference or legal obligation to process certain data streams in their own premises.

The many constraints and trade-offs make the deployment of AI RAN applications a major challenge. First, a developer needs to understand the underlying constraints which may differ from one deployment to another. Second, they need to carefully choose what data to collect and where to place the various application blocks, to maintain high accuracy, while respecting the constraints. Third, deployed applications will naturally have competing requests for the same resources, meaning that it falls on the developer to design their application with placement flexibility in mind. Finally, as alluded to in the case of AI-and-RAN, AI RAN applications have to coexist with other AI applications (non-RAN) that also execute on the edge such as video analytics and self-driving cars [10]. In the absence of a structured framework, AI RAN applications have to be developed with ad hoc programming interfaces and manually distributed by developers. As a result, they are rarely deployed at scale in production settings and have led to their potential remaining largely untapped.

V. Distributed AI platform for RAN

We present our vision for distributed AI-native RAN platform, with the architecture illustrated in Fig. 3. It builds on top of three components; i) programmable probes for the flexible collection of data, ii) AI processor runtimes for the deployment of AI applications across the distributed compute fabric, and iii) an orchestrator for the coordination of the platform.

Programmable probes for dynamic data collection – To allow developers to define optimal feature sets for their AI applications, we propose the use of dynamic probes for the platform (i.e., OS kernel) and the userspace (i.e., RAN network functions) to programmatically collect data. Through the probes, a developer could write small pieces of code to access raw events and data structures and summarize them in a custom way. This would expose the right features for training and inference, while minimizing the volume of generated data. For example, a developer could leverage a probe to access the raw IQ samples of the base station to feed them directly to the

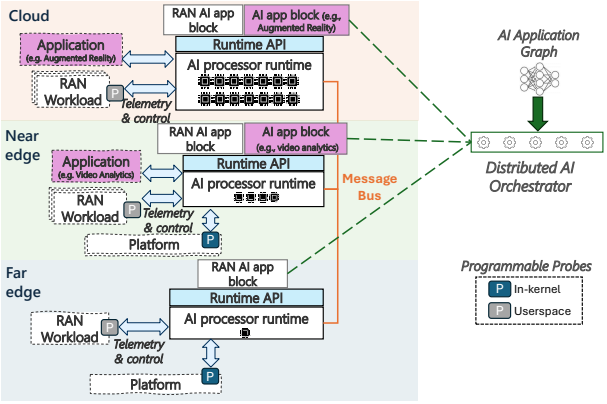

Fig. 3. **Distributed AI platform architecture.**

application, like in the example of the RAN slicing scheduler. Alternatively, they could pre-process them and export a derived KPI, like in the case of the anomaly detection application. The same approach could also be used to capture platform data (e.g., capture the incoming TCP packets and calculate the average inter-packet delay).

While this approach provides flexibility in tailoring the data collection process to the AI application's requirements, it also introduces safety concerns (e.g., illegal memory accesses). For this, we propose to use probes based on the eBPF technology, which has recently caught the attention of the telco industry [12], [13]. eBPF allows the injection of code to instrumentation points, subject to a static verification process that guarantees the safety of the injected code. While eBPF originates in the Linux kernel, recent advances [12] have expanded its use to userspace RAN applications.

AI processor runtime – Considering the constraints and trade-offs that characterize the various locations of the infrastructure, an AI-native platform should allow the seamless deployment of AI applications to the most suitable location, without the developer having to worry about providing location-specific flavors (i.e., implementations). In the proposed architecture, this is achieved through the *AI processor runtime*. The runtime can be deployed at any location where AI applications are expected to run and introduces an API for its interactions with the applications. The runtime should provide the following functionalities:

– *Data ingestion and control:* It should enable the ingestion of data captured through the local programmable probes and their exposure to applications. It should also enable the issuing of API calls towards the RAN functions and the platform for closed loop control.

– *Data exchange:* It should allow the exchange of data streams amongst the blocks of AI applications deployed in different locations through a common message bus. Similarly, it should enable the exposure and issuing of RPCs for applying control decisions.

– *Execution environment:* It should implement the process of running inference or training tasks, by abstracting the underlying compute resources (e.g., CPUs and GPUs) and exposing them to the AI applications, catering to both RAN and non-RAN AI applications.

– *Life-cycle management:* It should provide a standard interface to deploy, update, and remove AI applications, as well as to monitor their performance and resource utilization.

The distributed AI RAN platform does not prescribe an implementation for the AI processor runtime, as long as it provides the functionalities described above. We eschew making a prescription because we believe that the framework should be extensible to include existing and future runtime environments and messaging technologies, thus not restraining AI developers. As an example, one could consider using Docker containers combined with a WebAssembly (WASM) runtime as a highly portable and sandboxed execution environment, while the message bus implementation could be based on REST or gRPC calls.

Orchestrator – The distributed AI framework is overseen by an orchestrator (Fig 3) that is responsible for the placement of the AI applications across the AI processor runtimes. The orchestrator allows for dynamic addition or removal of applications, and also handles dynamic changes to the available resources (compute and network). It also allows to plug in diverse policies that trade off between compute at the different edges and the network load between the edges.

Applications are exposed to the orchestrator in the form of blocks in a graph.The blocks are characterized by requirements in terms of compute, memory, network, latency, as well as other constraints (e.g., the locations where the block is not allowed to run due to privacy concerns). The orchestrator takes into account both the developer requirements and the capabilities of the infrastructure and places the blocks to the AI processor runtimes that maximize the overall utility of the platform. Unique to AI workloads are *inference parameters* that influence resource demand [10], and hence is a resource allocation knob for the orchestrator. Examples of inference parameters would be data sampling rates, or AI models for anomaly detection. The sampling rate directly impacts the compute and network demand, while the choice of model dicates the compute demand, including the necessity for GPUs. We propose the AI RAN applications to expose their inference parameters to the orchestrator, and design the orchestrator to dynamically change the inference parameters.

We use the example applications of Fig. 2 to explain the operation of the orchestrator. We consider a far-edge location without a GPU and a limited amount of CPUs, a near-edge location with a single GPU, and a cloud location with many GPUs. Under this setup, we consider the three cases of Fig. 4. In the first case, we install the anomaly detection application. Given that the root cause analysis model requires a GPU, the orchestrator places it at the near edge and the rest of the applications' blocks at the far edge, minimizing the amount of outgoing traffic. In the second case, we install the RAN slicing scheduler application in addition to the anomaly detector. The latency-sensitive inter-slice scheduler requires a GPU, however the one available at the near edge is already reserved. Therefore, the orchestrator migrates the root cause analysis model to the cloud, since it is not latency sensitive, and deploys the inter-slice scheduler to the near edge. The

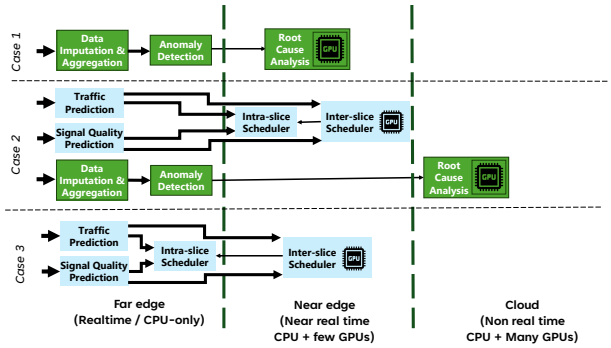

Fig. 4. **Distributed deployment scenarios for orchestrated AI applications.**

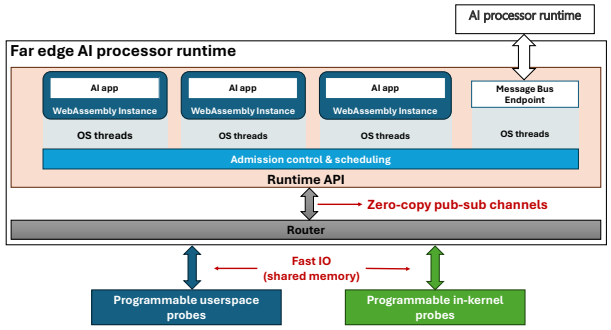

Fig. 5. **Architecture of far edge AI processor runtime.**

latency-tolerant intra-slice scheduler is also deployed at the near edge, since there is not enough CPU capacity at the (preferable) far edge location. Finally, in the third case, we illustrate how the orchestrator handles the completion of the anomaly detector and reschedules the radio resource scheduler. With the anomaly detector removed, CPU resources are freed up at the far edge, so the orchestrator migrates the intra-slice scheduler there, reducing the traffic sent to the near edge.

VI. An Efficient AI Processor Runtime for the Far Edge

We advocate that a highly optimized far edge runtime is required, as it is expected to host real-time AI applications (e.g. the RAN slicing scheduler), as envisaged with a concept of dApps [14]. As such, it needs to have sub-millisecond reaction times. Furthermore, the far edge is resource constrained, with only a small fraction of CPUs available for AI applications, and with a small or no GPU present. Our proposed design is illustrated in Fig. 5 and has the following characteristics:

Tight integration with the probes – The AI processor runtime communicates with the probes through fast IO channels using shared memory and a zero-copy mechanism. This allows the passing of data between the probes and the AI applications with very low compute and latency overhead.

Efficient, flexible and secure execution environment – We have experimented with the use of WASM as an execution environment and observed its benefits. It enables the sandboxing of applications with a very small overhead, while maintaining near-native runtime performance. The interaction with the rest of the system for the acceleration of inference or the data collection can take place through API calls exposed by the WASM Interface (WASI). For example, wasi-nn could be used for the exposure of the CPUs and the GPUs (when available).

Admission control & real-time resource allocation – All AI applications deployed on the processor runtime need to request a fraction of the CPU time or the GPU area. An admission control process ensures that this request can be met, while also serving other applications. Once deployed, an appropriate schedule is imposed by the runtime (for example, for the CPU we leverage the Linux deadline scheduler).

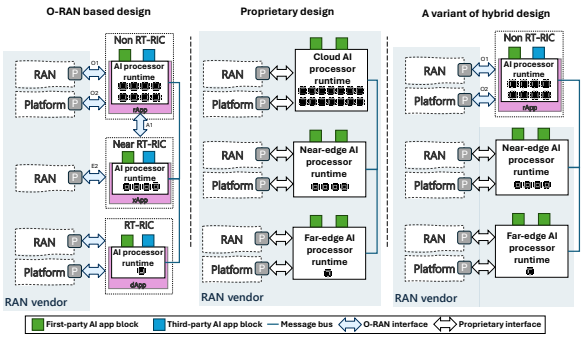

Fig. 6. **Deployment options for integration of distributed AI platform with the RAN.**

VII. Open vs Closed Architecture and Interfaces for Integration with RAN

Despite the progress within the O-RAN community, many fundamental tensions remain around control interfaces. For example, several major vendors do not support the idea of a near real-time RIC [15], expressing concerns regarding the network stability and security. If RAN vendors allow developers to exert fine-grained control on the RAN behavior, these may clash with the proprietary algorithms vendors have implemented. Also, some vendors argue that exposing sensitive RAN data may affect the security of the network and the data privacy. Vendors are also reluctant to open up some of their interfaces, as this may reduce their competitive advantage. All these concerns are important, and different operators take different stances on them. Our view is that the distributed AI platform proposed in this article should be a building block that can be customized appropriately for different use cases. Next, we briefly discuss some examples of deployment options (also illustrated in Fig. 4).

O-RAN based design – One extreme is an O-RAN based design (Fig. 6, left), allowing any first or third-party vendor to deploy AI applications across the RAN infrastructure. One could leverage the O-RAN RICs and integrate the AI processor runtimes in them as apps (i.e., rApps, xApps, dApps). For the local data collection and control operations, the AI processor runtimes could leverage the open RIC interfaces (e.g., E2, O1), augmented with programmable probes and exposed to the AI runtimes through appropriate adapters. The communication between the AI processor runtimes could be facilitated by a message bus, which could be standardized and realized as an overlay network on top of the RIC fabric. For the far-edge, and considering that there is currently no real-time RIC

specification, one could leverage our far-edge AI processor runtime design as a blueprint.

Proprietary design – Another extreme is a proprietary design, fully controlled by the RAN vendor (Fig. 6, middle). In this case, the group that is in charge of the RAN product development could also be in charge of developing and managing the AI platform and of defining proprietary interfaces for data collection and control. A different group of the RAN vendor could deploy and manage their own AI RAN applications, without having to constantly go through the RAN product development group, to ask them to introduce new features or expose more data. By decoupling the responsibilities, the vendor can accelerate its innovation, while still maintaining a full control over the RAN behavior.

Hybrid design – It is also possible to create a hybrid solution (Fig. 6, right), that would be a mix of proprietary AI processor runtimes and runtimes running on top of standard O-RAN RICs (e.g., only non RT-RIC). AI applications could be composed of a mixture of first-party and third-party AI blocks, where the proprietary data and control interfaces of the RAN functions can only be accessed by the first-party AI blocks, whereas thrid-party blocks can only access APIs exposed by the standard interfaces. This would allow the RAN vendor to maintain control over the RAN behavior, while still allowing third-party developers to innovate.

VIII. Conclusions

In this paper we focused on the topic of AI as a key enabler behind realizing the 6G vision. We argued about the benefits of introducing AI across several dimensions of the RAN, from the management and infrastructure up to the application layer. Based on these observations, we outlined the challenges for deploying AI solutions, stemming from the requirements of the applications and the characteristics of the RAN infrastructure. Motivated by these challenges, we proposed a distributed AI platform architecture. Its goal is to alleviate the common painpoints of deploying AI applications in the RAN without being prescriptive about the implementation details. This allows the platform to be tailored to the needs of the infrastructure provider or a standardization body. We believe that by identifying the future requirements and by initiating a discussion on the architecture of a 6G AI platform we can help both the standards and the vendors to create new opportunities for introducing AI solutions in this space.

Ganesh Ananthanarayanan is a Senior Principal Researcher at Microsoft. His research focuses on systems and networking, with current work on AI inference systems. He received his PhD from UC Berkeley.

Xenofon Foukas is a Principal Researcher at Microsoft. His research interests are on networks and distributed systems with his current focus on edge computing. He received his PhD from the University of Edinburgh.

Božidar Radunović is a Senior Principal Researcher at Microsoft Research. His research interests are in design and building next generation cloud and edge infrastructure. He received his PhD from EPFL.

Yongguang Zhang is currently a Partner Principal Researcher at Microsoft Research. He is currently leading the cloud RAN platform R&D. He received his Ph.D. in Computer Science from Purdue University.

References

[1] A.-R. Alliance, "Integrating AI/ML in Open-RAN: Overcoming Challenges and Seizing Opportunities," AI-RAN Alliance, Tech. Rep., Aug 2024.

[2] I. A. Bartsiokas *et al.*, "ML-based radio resource management in 5G and beyond networks: A survey," *IEEE Access*, vol. 10, pp. 83 507–83 528, 2022.

[3] C. Sun *et al.*, "SpotLight: Accurate, Explainable and Efficient Anomaly Detection for Open RAN," in *ACM MobiCom (To appear)*, 2024.

[4] L. Kundu *et al.*, "Towards Energy Efficient RAN: From Industry Standards to Trending Practice," *arXiv preprint arXiv:2402.11993*, 2024.

[5] "Building Software-Defined, High-Performance, and Efficient vRAN Requires Programmable Inline Acceleration," Nvidia Developer (online), 2023, https://developer.nvidia.com/blog/building-software-defined-high-performance-and-efficient-vran-requires-programmable-inline-acceleration/.

[6] X. Foukas and B. Radunovic, "Concordia: Teaching the 5g vran to share compute," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 580–596.

[7] L. L. Schiavo *et al.*, "Yinyangran: Resource multiplexing in gpu-accelerated virtualized rans," in *IEEE INFOCOM*, 2024, pp. 1–10.

[8] S. E. Trevlakis *et al.*, "Localization as a key enabler of 6g wireless systems: A comprehensive survey and an outlook," *IEEE Open Journal of the Communications Society*, 2023.

[9] A. Chorti *et al.*, "Context-aware security for 6g wireless: The role of physical layer security," *IEEE Communications Standards Magazine*, vol. 6, no. 1, pp. 102–108, 2022.

[10] Y. Zhang *et al.*, "Vulcan: Automatic query planning for live ML analytics," in *NSDI 24*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 1385–1402.

[11] A. Balasingam *et al.*, "Application-Level service assurance with 5g RAN slicing," in *NSDI 24*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 841–857.

[12] X. Foukas *et al.*, "Taking 5g ran analytics and control to a new level," in *ACM MobiCom*, 2023, pp. 1–16.

[13] D. Soldani *et al.*, "ebpf: A new approach to cloud-native observability, networking and security for current (5g) and future mobile networks (6g and beyond)," *IEEE Access*, vol. 11, pp. 57 174–57 202, 2023.

[14] S. D'Oro *et al.*, "dApps: Distributed applications for real-time inference and control in O-RAN," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 52–58, 2022.

[15] "AT&T, all in with Ericsson, seems to have shut the door to xApps," LightReading (online), 2024, https://www.lightreading.com/open-ran/at-t-all-in-with-ericsson-seems-to-have-shut-the-door-to-xapps.