Towards Seamless 5G Open-RAN Integration with WebAssembly

Raphael Cannatà*
EPFL

Haoxin Sun*
EPFL

Dan Mihai Dumitriu
Pavonis LLC

Haitham Hassanieh
EPFL

ABSTRACT

O-RAN (Open Radio Access Network) multivendor integration, despite its promise of a diverse 5G ecosystem, faces compatibility challenges such as vendor-specific implementations. We propose WA-RAN, a novel framework that leverages WebAssembly (Wasm) to enhance interoperability and flexibility within the O-RAN architecture. Using Wasm plugins, WA-RAN addresses the complexities of integrating multivendor equipment, enables components update on the fly, and facilitates the introduction of new features. Additionally, it offers platform and language agnosticism along with enhanced security via sandboxing. We demonstrate the design of WA-RAN through two use cases in 5G: a slice scheduler and a near-Real-Time RAN Intelligent Controller (near-RT RIC). Our implementation and evaluation show WA-RAN potential to safely overcome O-RAN integration challenges, providing a promising path to versatile 5G networks.

CCS CONCEPTS

• **Networks** → **Mobile networks**; *Programmable networks*; *Network management*.

KEYWORDS

5G, RAN Slicing, WebAssembly, O-RAN

ACM Reference Format:
Raphael Cannatà, Haoxin Sun, Dan Mihai Dumitriu, and Haitham Hassanieh. 2024. Towards Seamless 5G Open-RAN Integration with WebAssembly. In *The 23rd ACM Workshop on Hot Topics in Networks (HOTNETS '24), November 18–19, 2024, Irvine, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3696348.3696864

*Both authors contributed equally to this research.

HOTNETS '24, November 18–19, 2024, Irvine, CA, USA

ACM ISBN 979-8-4007-1272-2/24/11
https://doi.org/10.1145/3696348.3696864

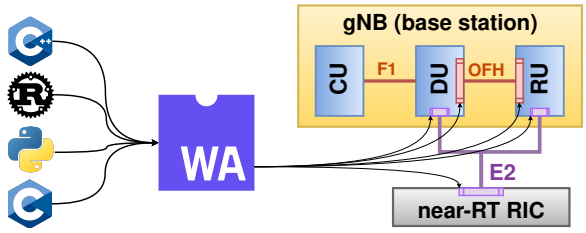


Figure 1: Framework of WA-RAN: The software, written in a high-level language, is compiled into Wasm bytecode and pushed into the RAN.

1 INTRODUCTION

Traditionally, the Radio Access Network (RAN) in cellular systems has been a closed box, typically supplied by a single vendor like Nokia, Ericsson, or Huawei [27, 72, 83]. Such a monolithic architecture has significantly limited innovation and progress in cellular networks, increased security risks [24], and led to complete capture of the RAN market by a few companies [3, 4, 62, 63, 78]. The Open-RAN Alliance [19], which was founded in 2018 and includes major cellular providers, vendors, chip manufacturers, and cloud providers, aims to promote flexibility and innovation in the 5G infrastructure by opening the radio access network [6, 22, 64]. Its goal is to define standardized interfaces, specifications, and protocols that enable the integration of equipment from multiple vendors within the 5G RAN. Such an approach allows for a diverse ecosystem where operators can select and combine the best-in-class components. This leads to potential cost savings, enhanced performance, easier transfer of innovative research, and accelerated deployment of new services [1].

However, current O-RAN implementations face several challenges, including the complexity of integrating components from different vendors, the need for extensive testing to ensure compatibility, and the difficulty of deploying new features and services [55, 56, 80]. Although there are standardized interfaces and protocols, the difficulties in interoperability mainly come from a certain degree of flexibility left in the specifications [38] (e.g., bit length for controlling radio output power [68]). Thus, vendors interpret specifications differently when developing equipment, which can lead to inconsistencies [68, 75]. This leads to a situation where the

equipment from different vendors may not work together as expected [46]. Moreover, an equipment manufacturer, in order to create a more desirable product, can extend the standard interfaces with proprietary features, rendering the integration with other vendors' products difficult.

For example, the E2 interface, shown in Fig. 1, is a crucial component that allows different vendors' near-Real-Time RAN Intelligent Controllers (RICs) to control and interact with different vendors' base stations (referred to as gNBs in 5G) [7]. However, in practice, there are compatibility issues between RIC implementations and gNB implementations that prevent seamless integration [54, 77]. Similarly, the O-RAN and 5G standards have proposed several functional splits of the 5G RAN [70]. These define how the different functions inside the gNB are split between CU (Central Unit), DU (Distributed Unit), and RU (Radio Unit), shown in Fig. 1. Through standardized interfaces, operators can mix and match CU, DU, and RU equipment from different vendors. However, even for common functional splits, such as the 7.2x split, compatibility issues arise between O-RAN RUs from one vendor and O-RAN DUs from another vendor [80], placing a huge system integration burden on commercial operators.

These integration challenges have led operators to take a wait-and-see approach to the movement of O-RAN or to source their entire O-RAN stack from a single vendor [12, 55, 56], which defeats the purpose of O-RAN. For instance, AT&T in the US relies solely on Ericsson for their O-RAN deployments [52, 54], Vodafone in the UK deploys the O-RAN network using Samsung equipment [69], and Vodafone in Italy relies fully on Nokia for their O-RAN tests [59].

A similar scenario with the illusion of interoperability also happened in data center switching. OpenFlow [50], a standardized protocol for controlling network switches, initially promised vendor-agnostic programmability. However, the need for vendor-specific extensions to address diverse use cases quickly shattered the vision of true interoperability. As a result, OpenFlow is seldom used in today's data centers [29]. To avoid the same outcome as OpenFlow, it is essential to revise Open-RAN to enable seamless integration.

In this paper, we propose WA-RAN, a WebAssembly-based 5G O-RAN architecture. Its framework is depicted in Fig. 1. WA-RAN exploits WebAssembly (Wasm) plugins [25] to run various components of the 5G RAN stack, solving conventional O-RAN integration issues. Wasm is a binary instruction format introduced in 2017, designed to run portable executables with near-native speed in web browsers [81, 85]. Its plugin-based nature has gained popularity in other domains such as edge computing and IoT [26, 44, 51, 66]. In particular, by exposing selected *host functions* to enclaved Wasm sandboxes, 5G RAN vendors can facilitate **interoperability**, allowing any third party to integrate with their RAN by developing compatible plugins. Wasm's portable bytecode can be compiled from various source languages [51], is platform-agnostic, and supports different instruction sets (ARM and x86), drivers, operating systems, and programming languages. This universality enables the same plugin to run on devices from different vendors.

In the framework of WA-RAN, programs are written in various high-level languages, which are compiled into Wasm plugin bytecode and pushed into the devices. Required functionalities such as communication between different components can run inside these plugins to facilitate interoperability. For example, traditionally, different data bit lengths in standardized interface implementations cause compatibility issues. Since devices are often closed, an extensive joint effort has to be made from all vendors, which is impractical in most cases. WA-RAN provides an easy workaround for this by integrating interfaces into Wasm plugins, allowing third parties to get around this difference in device implementations.

Two additional benefits of using WA-RAN are **flexibility** and **security**. By encapsulating a RAN component within a plugin, the modular design offers enhanced flexibility and streamlines the update process compared to conventional RAN software. Upgrading an independent plugin eliminates the need to modify and recompile the entire RAN stack. This approach also allows seamless on-the-fly integration of new components. Minimizing downtimes is essential in cellular infrastructure, particularly for networks supporting around-the-clock applications, such as industrial automation or IoT communications, where uninterrupted operation is crucial. Furthermore, the Wasm plugins operate within isolated sandboxes, providing security and memory safety to the host [14]. Wasm's sandbox ensures that even if a plugin contains unsafe code, the host remains secure and unaffected.

However, designing WA-RAN to use Wasm in 5G development is challenging. First, Wasm was not originally intended for this use, so its tools do not fully meet the needs of 5G. Second, 5G demands very tight execution time guarantees, necessitating thorough system planning to ensure that running interface communication and plugin functions can still meet these guarantees. We demonstrate how to overcome these challenges through a system design of WA-RAN for two 5G use cases: *slice scheduler for Mobile Virtual Network Operators (MVNOs)* and *near-RT RIC development and integration*. We show how to integrate Wasm plugins seamlessly into the 5G RAN and how to utilize these plugins to run 5G functions in real-time. Through this, we illustrate the framework of WA-RAN and offer a blueprint for implementation.

We also provide a concrete and real-world implementation of the former use case of MVNOs as a proof of concept, based on an open-source 5G RAN project. The implementation is evaluated across various metrics, including functionality, flexibility, memory safety, and execution time, proving the

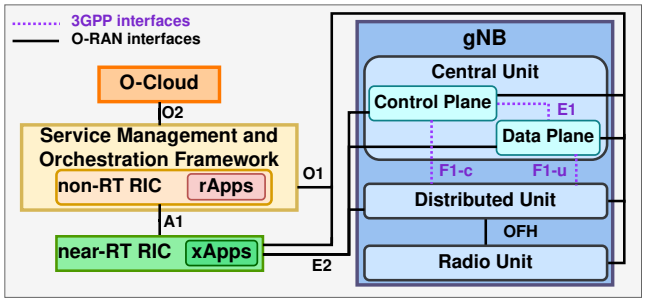


Figure 2: Architecture of a 5G O-RAN: It includes the CU, DU, RU, and the RICs, plus the relative interfaces.

feasibility of implementing WA-RAN and showcasing its advantages. Finally, we highlight open research problems that need to be solved to enable WA-RAN to seamlessly integrate any O-RAN 5G interface or function.

2 BACKGROUND AND RELATED WORK

A. Open-RAN. Open-RAN's vision is to create a more competitive and innovative ecosystem by disaggregating the traditional monolithic RAN architecture into flexible and interoperable components [82]. This disaggregation is achieved through standardized interfaces that allow components from different vendors to work together seamlessly. Fig. 2 illustrates the high-level architecture of a 5G O-RAN network built on top of the 5G cellular standards defined by 3GPP [18]. The base station (gNB) is divided into three parts: the Central Unit (CU), the Distributed Unit (DU), and the Radio Unit (RU). The CU is responsible for centralized control and management of the radio resources, the DU is responsible for distributing the baseband processing functions, and the RU is responsible for transmission and reception of the radio signal. These components communicate with each other through standardized interfaces such as the OFH interface (between DU and RU), the F1 interface (between DU and CU), and the E1 interface between the control and data plane in the CU.

The architecture also includes the RAN Intelligent Controller (RIC), responsible for controlling and managing the radio resources. There exist a non-Real-Time (non-RT RIC) and a near-Real-Time (near-RT RIC) [7].[1] The non-RT RIC, hosting applications called rApps, manages non-time-critical tasks such as network optimization and analytics. While the near-RT RIC hosts applications known as xApps, which handle time-critical tasks such as radio resource management, dynamic network optimization, as well as AI and ML workflows [64]. It communicates with and controls the gNB over the E2 interface, which is designed to be vendor-neutral. The E2 interface should theoretically allow any vendor's near-RT RIC to control any vendor's gNB. However, in practice, it faces challenges such as compatibility issues between different vendors [54]. Additionally, as a standardized protocol, the E2 interface supports a limited set of functionalities, making the implementation of advanced features difficult. These challenges have hindered its adoption, prompting researchers to develop solutions that are not standard-compliant to bypass the E2 interface [15, 16, 20, 39, 87].

[1]The 5G standards also talk about a Real-Time RIC, but it is currently not in any of the specs [82].

B. WebAssembly. Wasm is a binary instruction format designed as a portable target for the compilation of high-level languages like C, C++, and Rust [25]. Its original aim was to enable high-performance applications to run in a web browser, but its utility extends far beyond that environment, finding applications in areas such as functions-as-a-service [48], IoT [41], and edge computing [26]. Two characteristics of Wasm are [8]:

- *Portability & Performance*: Wasm modules are platform-independent and can run in any environment that supports the Wasm runtime, making it an ideal choice for heterogeneous systems. It is designed for near-native performance, taking advantage of features such as efficient binary encoding, linear memory, and direct access to hardware capabilities. It also supports Ahead-of-Time (AoT) compilation, which converts Wasm modules into native machine code before execution, further enhancing performance.
- *Security & Safety*: It enforces strict memory safety by using a linear memory model and bounds checking, preventing common vulnerabilities like buffer overflows, and ensuring safe memory access. It also maintains control flow integrity by ensuring that function calls and returns follow a well-defined and verifiable control flow graph, preventing malicious alterations to the execution flow. Finally, Wasm plugins run in isolated sandboxes. Hence, untrusted code will not affect the host system.

Given these advantages, some work [2, 17, 35, 84] proposed to integrate Wasm into 5G networks. However, these works focus on exploiting Wasm in specific 5G applications such as edge computing, not in the 5G RAN infrastructure.

C. eBPF. An alternative to WebAssembly is to use eBPF (extended Berkeley Packet Filter), which was recently leveraged to develop a programmable RIC in [20]. eBPF allows third parties to load and run codelets inline within the 5G RAN in real-time. Both eBPF and Wasm are popular lightweight code execution sandbox techniques [86], but their original objectives are different. The goal of this paper is not to dismiss eBPF as a potential solution, but rather to propose a new solution that we argue is more suitable for the following reasons. First, eBPF typically runs inside the kernel space [53, 79], making it suitable for user plane development [13, 60, 61, 71], while Wasm runs inside the user space. The difference means that eBPF has more constraints than Wasm [21, 28], thus developers will need extra efforts when implementing some

RAN functionalities with eBPF compared to Wasm [31]. For example, eBPF does not have built-in floating point arithmetic and unbounded loops, which Wasm supports and are necessary for certain RAN functionalities. Second, eBPF bytecode is mostly compiled from a restricted subset of the C language [74, 79], while Wasm is a compilation target for various source languages. Finally, eBPF assumes mainly trusted software and uses a verifier to statically verify the safety of an eBPF program [14]. However, known vulnerabilities allow malicious eBPF programs to bypass static checks, and execute unsafe behaviors such as executing arbitrary kernel code or accessing arbitrary kernel memory [45]. Conversely, Wasm enables untrusted code to run without compromising the host. This difference in security makes Wasm more suitable in some RAN use cases.

3 USE CASES AND MOTIVATION

We demonstrate two potential use cases of WA-RAN, and explain why it can outperform the conventional 5G O-RAN architecture. Nevertheless, there are more use cases that can benefit from WA-RAN, such as customized private 5G deployments, intrusion detection and prevention systems, and forward error correction (FEC) decoding. However, we will focus on the following two scenarios.

A. Slice Scheduler for MVNOs. MVNOs are wireless service providers like Google Fi, Cricket Wireless, and Metro PCS that do not own their own network infrastructure. Instead, they enter into agreements with Mobile Network Operators (MNOs) like AT&T, Verizon, and T-Mobile to access their infrastructure at wholesale rates. The MVNO market is highly competitive, with many providers offering various services to attract customers. For example, the US has 139 MVNOs operating on the networks of the four largest MNOs [33]. Similarly, Germany had 135 MVNOs operating with just three MNOs [65]. Some MVNOs focus on high-speed data, others on low-cost voice services, and others on Machine-to-Machine (M2M) and IoT services.

5G introduced the concept of network slicing, where the MNOs can slice their network resources among different services, enterprises, or MVNOs. This allows MVNOs to customize their own scheduling algorithms and differentiate their services by optimizing resource allocation based on their unique business models [10, 40]. However, in an MNO's network infrastructure, different gNBs may use various hardware, operating systems, and programming languages, making it difficult to onboard new resource schedulers. Moreover, adding or removing MVNOs would require downtime at the gNB. Similarly, updating the schedulers or their configurations would require downloading new software into the gNBs and then stopping, recompiling, and restarting the gNBs.

WA-RAN can address these issues. Since Wasm plugins are software- and hardware-agnostic, MNOs and MVNOs need only to update a single unified plugin to roll out new schedulers seamlessly. The update can be done on the fly, ensuring real-time adaptation to changing network conditions and user demands without stopping or redeploying gNBs.[2] Furthermore, MNOs may have security concerns about running MVNO software in their RAN infrastructure, since it could compromise network integrity. WA-RAN offers a secure sandbox environment that isolates the MVNO scheduler from the host system. This isolation prevents interference with the host or other plugins, enhancing RAN security. Additionally, MNOs can perform static analysis on the MVNO scheduler plugin before deployment, further ensuring safety.

B. Functional Splits and Near-Real-Time RIC. WA-RAN can bypass the complexities of a single, standardized protocol by implementing the interfaces on both DUs and RUs as independent plugins. It wraps the low-level communication, allowing operators and vendors to easily design plugins without jeopardizing overall interoperability. This is more practical than directly changing the device, which is usually closed-source. So, how can Wasm address compatibility issues in O-RAN? WA-RAN does not require vendors to implement new interfaces. Instead, each vendor implements their own interfaces and shares the specifications with the System Integrator (SI). The SI can then use a Wasm plugin to ensure compatibility between interfaces from different vendors. Take the example in the introduction regarding varying bit lengths, the plugin can convert 8-bit data to 12-bit, making the communication possible. While this approach requires additional effort from the SI, it is significantly less complex than modifying an entire vendor's codebase, which is often proprietary and inaccessible. The flexible design of the underlying communication also enables customizations in different scenarios.

Similarly, adding one abstraction layer on gNBs and near-RT RICs can be a workaround for their interoperability. This also simplifies the integration of new functionalities. Currently, RIC developers need to cope with existing standards and implementations, increasing workloads and limiting innovation. For example, fitting a new scenario requires specifying a new service model in the RIC and undergoing long standardization processes [20]. In WA-RAN, new features can be introduced by developing lightweight plugins, rather than requiring extensive modifications to the protocols. WA-RAN allows integrators and operators to implement the RIC control plane according to their specific needs, utilizing whichever wire protocols and distributed systems paradigms are most suitable for their environment.

[2]Note that while dynamically linked libraries could be an alternative, they lack isolation from the process and are intrinsically unsafe.

Moreover, near-RT RICs are vulnerable to attacks [32] such as Denial of Service (DoS), which can be triggered by untrusted xApps sending malicious packets [73]. Running the communication and message decoding in sandboxes ensures sanitization, meaning no malicious packets or codes can be injected into the host RIC and RAN. Thus, executing xApps as Wasm plugins enables the safe execution of third-party applications while providing a unified bytecode that can be executed seamlessly within different RICs, regardless of the underlying platform or programming language.

4 SYSTEM OVERVIEW

In this section, we explain the WA-RAN framework in detail by providing the system design of the MVNO slice scheduler and the near-RT RIC.

A. Slice Scheduler for MVNOs. We present a two-level scheduler that outsources the resource allocation decision-making of each slice to external plugins using WA-RAN. This approach allows third parties, such as MVNOs, to customize their own scheduling policies. The structure of this system is depicted in Fig. 3. Its functionality can be understood from the perspectives of both the gNB host and the MVNO plugin.

gNB Host. At every time slot, the gNB runs the inter-slice scheduler, which is responsible for dividing the available resources among the different slices. Each slice represents an MVNO. This allocation can be performed in various ways, such as using fixed resource percentages, prioritizing latency-sensitive information, or targeting specific bit rates. Once the inter-slice scheduler has allocated resources to a slice, it calls the intra-slice scheduler responsible for that slice, implemented as a plugin, which receives the amount of allocated resources and a list of UEs currently subscribed to that slice. The list contains valuable information for the plugin to make allocation decisions, such as channel quality, buffer status, long-term throughput, and UE identifiers. The plugin processes this information and returns a list of UEs with their identifiers, their order of priority in the allocation, and the amount of resources allocated to each UE. The inter-slice scheduler then uses this information to perform the actual resource allocation, preparing the grants for the UEs and communicating them over the control channel.

MVNO Scheduler Plugin. The MVNO plugin is responsible for the intra-slice scheduler, deciding how to distribute the resources received from the inter-slice scheduler among the UEs subscribed to its slice. This decision is entirely up to the MVNO and depends on the type of traffic the operator is targeting. For example, in our evaluation, we used three intra-slice scheduling algorithms to address different traffic types: Round Robin (RR), Proportional Fair (PF), and Maximum Throughput (MT). This flexibility allows MVNOs to tailor their scheduling strategies to their specific needs,

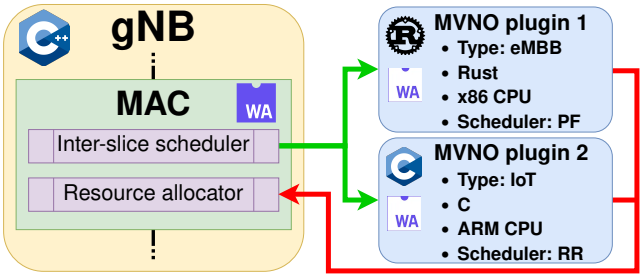


Figure 3: Slice scheduler for MVNOs within WA-RAN: The inter-slice scheduler calls the MVNO plugin, which does the intra-slice scheduling. Then the plugin returns its scheduling decision, which is used by the resource allocator to do allocation for all UEs.

optimizing service quality and standing out from the market competition.

WA-RAN's framework enables MVNOs to implement customized scheduling policies through external plugins. This modular approach enhances flexibility and allows for better and more secure optimization of network resources.

B. Near-Real-Time RIC. We demonstrate an example design of the near-RT RIC and the corresponding E2 nodes (CU and DU in the gNB) within the WA-RAN framework, as shown in Fig. 4. Plugins are located in both the near-RT RIC and the E2 nodes. This system includes two categories of Wasm plugins.

Communication Plugins. The first category of plugins is communication plugins, responsible for exchanging messages (blue in Fig. 4). Their primary objective is to wrap the wired protocol and enable bidirectional communication between the RIC and the E2 nodes, allowing the E2 nodes to send performance metrics to the RIC and receive control actions in return. In this proposed framework, the operator or vendor who wants to integrate with other devices can customize the underlying communication protocol, according to their use cases and requirements. Thus, they have flexibility over various aspects to define the communication by themselves, such as messaging protocols, data serialization formats, encryption algorithms, and timescales. For example, operators may choose to use ZeroMQ or Apache Kafka for communication, encode the payload in ASN.1, JSON, or Protocol Buffers (protobuf) [23], and encrypt the packet in AES [11] or RSA [67]. To manage the network, the gNB host exposes multiple *host functions*, which provide access to specific control processes in different layers of the gNB, such as changing the configuration of the Channel Quality Indicator (CQI) table, and modifying slice-level resource quota. The communication plugins, running on the E2 nodes, would call these exposed functions to trigger certain control actions once receiving control commands from the near-RT RIC.

xApp Plugins. The second category of plugins is xApp plugins (purple in Fig. 4), which encapsulate customized control

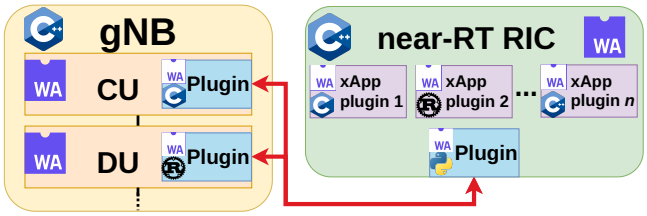


Figure 4: Near-RT RIC and gNB within WA-RAN: Instead of utilizing the E2 interface, the communication between the gNB and the RIC is wrapped by Wasm plugins on both sides. xApps in the RIC are also implemented as Wasm plugins to achieve flexibility.

logic for intelligently managing the RAN. Control applications such as traffic steering and slice SLA (Service-Level Agreement) assurance would be realized and executed within the plugins. Apart from implementing the control algorithms, these xApp plugins also export several functions, which can be called by the near-RT RIC host to run the control logic inside the plugins. For the traffic steering application, when the RIC host calls the plugin's exported function, it would invoke the internal decision-making process, and then return to the host the decision of whether some UEs need to conduct handovers. On the other side, the RIC host provides well-defined *host functions* to xApp plugins, which can achieve functionalities such as messaging between xApps. By calling these *host functions* properly in xApps plugins, WA-RAN ensures seamless integration and interaction between the near-RT RIC and xApps, independent of the implementation details of the RIC, which may vary from vendor to vendor.

In short, WA-RAN's framework encapsulates the wired protocol inside plugins, allowing operators to customize underlying communication protocols between E2 nodes and near-RT RICs. The proposed xApp plugins also mitigate the reusability and interoperability issues of xApps [9].

5 PRELIMINARY RESULTS

We demonstrate the feasibility of implementing WA-RAN in a practical 5G network by implementing and evaluating the two-level MVNO slice scheduler presented in §4.

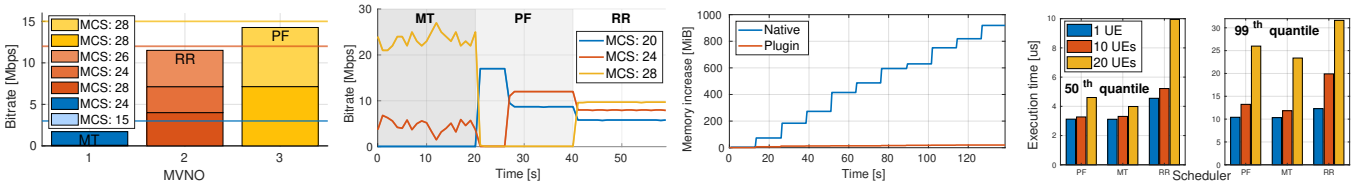
A. Implementation and Testbed. We implement WA-RAN on top of an open-source 5G RAN project called srsRAN [76] and use Open5GS [43] for the core network. We added the support for RAN slicing via a two-level scheduler. We build the Wasm host and plugin using Extism [5]. The modifications needed to support slicing and the external Wasm plugins accounted for 2000 lines of C code. Making it easy to retrofit to existing RAN solutions. We rely on Kubernetes to containerize the 5G network functions and the gNB. The core network is hosted on a Dell Precision Workstation 5820, while the gNB and the plugins run on an Intel NUC 12 with an Intel i7-1260P CPU and 64 GB of RAM. The gNB operates in FDD mode in band n3 (1.71 to 1.88 GHz), with 15 kHz subcarrier spacing and a 10 MHz bandwidth. To generate downlink (DL) traffic, all UEs are running `iperf3` [30]. We implemented the MVNOs as network slices with target rates and scheduling metrics, while admission control was managed by a centralized AMF. In a real-world deployment, additional factors such as personalized admission control, billing, and tracking would need to be taken into account.

B. Feasibility. We conduct an experiment with 3 MVNOs, each using a different scheduling strategy. MVNO 1 uses a MT scheduler with a target cumulative DL rate of 3 Mb/s. MVNO 2 uses a RR scheduler with a target cumulative DL rate of 12 Mb/s. MVNO 3 uses a PF scheduler with a target cumulative DL rate of 15 Mb/s. Fig. 5a shows that all MVNOs work correctly to achieve their target data rates and can co-exist in the same gNB.

C. Flexibility: Fig. 5b shows that WA-RAN is able to change the MVNO scheduler on the fly, without stopping the gNB or disconnecting any UE. In the evaluation, 3 UEs subscribe to the same MVNO, with the same target rate of 22 Mbit/s and different MCSs. In the first period, the MVNO is using MT, so the UE with the best channel (i.e., MCS 28) reaches its target rate, the UE with the second-best channel (i.e., MCS 24) utilizes the left resources, while the UE with the worst channel (i.e., MCS 20) is mostly not scheduled. Then the MVNO changes to PF scheduling. To stress the PF nature of the scheduler, we intentionally chose a large time constant τ, so to give a strong weight to the long-run throughput [37]. As the UE with MCS 20 has the smallest long-run throughput, it gets prioritized initially. After a while, the UE with MCS 24 also starts getting scheduled. In the last period, all the UEs equally share the resources as the RR scheduler is applied.

D. Memory Safety. We deliberately run unsafe code to verify the memory safety of WA-RAN. Using a Wasm plugin, we catch exceptions at runtime, allowing the gNB to respond appropriately. We test improper instructions such as *null pointer dereference*, *out-of-bounds memory access*, and *double free*. In all cases, the gNB host catches the exception and continues running, whereas running the improper code directly on the host causes a crash [49]. We also evaluate the gNB host's memory usage by continuously allocating memory at each scheduler execution without freeing it. Fig. 5c demonstrate that when running the unsafe code in a plugin, the system's memory usage remains stable. In contrast, running it directly on the host results in a linear increase in memory usage, indicating a memory leak.

E. Execution Time. We use Boost Accumulators [36, 57] to measure the running speed of plugins in WA-RAN. Fig. 5d shows the execution time of three plugins with varying numbers of UEs connected. The measured time includes the overhead of data serialization and de-serialization on the gNB



(a) Co-existence of MVNOs **(b)** Live swap of MVNO scheduler **(c)** Memory increases **(d)** Execution time

Figure 5: Preliminary evaluation.

host. For all plugins, 99 % of execution times are much smaller than the slot duration (1000 µs in our case), even with many UEs to schedule. Although higher numerologies may have shorter slot durations, these results were obtained without focusing on execution speed, indicating the feasibility of using Wasm plugins for 5G RAN real-time functionalities.

6 DISCUSSION AND OPEN PROBLEMS

A. Fault Tolerance. Future versions of WA-RAN aim to better handle plugin runtime faults, such as invalid or delayed outputs. To avoid disruptions, WA-RAN will implement mechanisms to detect, sanitize, and fix faults, ensuring continued gNB operation. For instance, in our scheduler system, the gNB can switch to a default scheduler or disconnect the MVNO if their plugin is not behaving as expected.

B. Resource Management. Since 5G RAN software runs at the network edge, computational and memory resources are limited. WA-RAN requires a joint resource management policy that ensures real-time functionalities in both the host and plugins meet stringent deadlines while optimizing other functionalities' performance.

C. Running Speed. Though Wasm plugins are lightweight and designed for high-performance applications, they still add additional overhead, slowing down the running speed compared to native code [34]. In the MVNO scheduler case, we showed that Wasm is capable of meeting the 5G time requirement. However, it is desirable to narrow this gap as much as possible. Literatures show that potential mitigations include AoT compilation, code optimization, and caching method lookups [42, 47].

D. Toolchain Development. Current Wasm development kits are not designed specifically for 5G RAN development, thus, they are not optimized for low latency applications. Extism, for instance, wraps around the Wasm runtime, making it easy to swap runtimes and maintaining compatibility with code written in any supported language. However, this generalization limits optimization potential, such as reducing execution time, which could be achieved using a native interface like Wasmtime. Given the unique requirements of 5G RAN plugins, developing tailored Wasm toolchains, such as compilers and sanitizers, would be advantageous.

E. Limitations of WA-RAN. We recognize that Wasm is not the sole solution for addressing the challenges within O-RAN, and additional issues, such as shared memory between components, either exist or may emerge. These concerns warrant careful consideration in the design of a comprehensive implementation. Additionally, while retrofitting WA-RAN introduces certain complexities, it is important to note that commercial-grade code is typically not open-source. Enabling the execution of sandboxed plugins is, nonetheless, a significant step forward. Furthermore, even within open-source implementations, components are often deeply integrated with the broader codebase, making them difficult to modify or upgrade externally.

F. Community. It is worth noting that, unlike the small and traditionally closed RAN community, Wasm enjoys broad support from an extensive community and major companies like Google, Mozilla, Microsoft, and Apple [66]. Currently, there are more than 5000 open-source Wasm-related projects on GitHub, compared to a couple of open-source RAN projects [58, 76]. Therefore, the future development of WA-RAN would benefit from such a large community.

7 CONCLUSION

In this paper, we argue that unless we find a solution to seamlessly integrate 5G O-RAN equipment from different vendors, history is doomed to repeat itself and O-RAN will suffer the same fate as Open-Flow. We introduced WA-RAN, a framework to address O-RAN integration challenges using WebAssembly. WA-RAN implements 5G RAN components as Wasm plugins, enhancing the interoperability, flexibility, and security of the existing 5G O-RAN architecture. It offers operators and vendors features like seamless integration, on-the-fly updates, flexible configuration, and sandboxing. We believe that WA-RAN can play a key role in the success of O-RAN, and we invite the networking research community to join us to build this ecosystem.

ACKNOWLEDGMENTS

We thank the reviewers and HotNets PC for their comments and suggestions. We also thank Arman Maghsoudnia, Aoyu Gong, and Eduard Vlad for their valuable feedback. We thank the srsRAN team and community for their support.

REFERENCES

[1] Aly S. Abdalla, Pratheek S. Upadhyaya, Vijay K. Shah, and Vuk Marojevic. 2022. Toward Next Generation Open Radio Access Networks: What O-RAN Can and Cannot Do! *IEEE Network* 36, 6 (2022), 206–213. https://doi.org/10.1109/MNET.108.2100659

[2] Saidulu Aldas and Andrew Babakian. 2023. Cloud-Native Service Mesh Readiness for 5G and Beyond. *IEEE Access* 11 (2023), 132286–132295. https://doi.org/10.1109/ACCESS.2023.3335994

[3] Jebra Aron, Olgag Ukhaneva, and Chloe Sun. 2020. *The economics of 5G deployment in the "race" to 5G: The role of Massive MIMO.* Independent report. Charles River Associates. https://media.crai.com/sites/default/files/publications/Insights-The-Economics-of-5G-Article-2-Massive-MIMO-August-2020.pdf Pag. 3.

[4] Jebra Aron, Olgag Ukhaneva, and Chloe Sun. 2020. *The economics of 5G deployment in the "race" to 5G: The role of open RAN.* Independent report. Charles River Associates. https://media.crai.com/wp-content/uploads/2020/11/20113348/Insights-The-Economics-of-5G-Article-Oct2020.pdf Pag. 2.

[5] The Extism Authors. 2022. Extism. https://extism.org/. [Online; Last accessed: 30-May-2024].

[6] Wilfrid Azariah, Fransiscus Asisi Bimo, Chih-Wei Lin, Ray-Guang Cheng, Navid Nikaein, and Rittwik Jana. 2024. A Survey on Open Radio Access Networks: Challenges, Research Directions, and Open Source Approaches. *Sensors* 24, 3 (Feb 2024), 1038. https://doi.org/10.3390/s24031038

[7] Bharath Balasubramanian, E. Scott Daniels, Matti Hiltunen, Rittwik Jana, Kaustubh Joshi, Rajarajan Sivaraj, Tuyen X. Tran, and Chengwei Wang. 2021. RIC: A RAN Intelligent Controller Platform for AI-Enabled Cellular Networks. *IEEE Internet Computing* 25, 2 (2021), 7–17. https://doi.org/10.1109/MIC.2021.3062487

[8] Rick Battagline. 2021. *The Art of WebAssembly: Build Secure, Portable, High-Performance Applications.* No Starch Press, San Francisco, CA.

[9] Chieh-Chun Chen, Mikel Irazabal, Chia-Yu Chang, Alireza Mohammadi, and Navid Nikaein. 2023. FlexApp: Flexible and Low-Latency xApp Framework for RAN Intelligent Controller. In *ICC 2023 - IEEE International Conference on Communications*. Institute of Electrical and Electronics Engineers (IEEE), Rome, Italy, 5450–5456. https://doi.org/10.1109/ICC45041.2023.10278600

[10] Yongzhou Chen, Ruihao Yao, Haitham Hassanieh, and Radhika Mittal. 2023. Channel-Aware 5G RAN Slicing with Customizable Schedulers. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 1767–1782. https://www.usenix.org/conference/nsdi23/presentation/chen-yongzhou

[11] Joan Daemen and Vincent Rijmen. 2002. *The design of Rijndael.* Vol. 2. Springer, Berlin, Germany. https://doi.org/10.1007/978-3-662-60769-5

[12] Alex Davies, Philip Hunter, Rafi Cohen, and Peter White. 2023. Open RAN Equipment Market Forecast 2023-2030. https://rethinkresearch.biz/report/open-ran-forecast/ [Online; Last accessed: 29-May-2024].

[13] Udhaya Kumar Dayalan, Ziyan Wu, Gaurav Gautam, Feng Tian, and Zhi-Li Zhang. 2023. PRAVEGA: Scaling Private 5G RAN via eBPF/XDP. In *Proceedings of the 1st Workshop on EBPF and Kernel Extensions* (New York, NY, USA) *(eBPF '23)*. Association for Computing Machinery, New York, NY, USA, 89–91. https://doi.org/10.1145/3609021.3609303

[14] Jules Dejaeghere, Bolaji Gbadamosi, Tobias Pulls, and Florentin Rochet. 2023. Comparing Security in eBPF and WebAssembly. In *Proceedings of the 1st Workshop on EBPF and Kernel Extensions* (New York, NY, USA) *(eBPF '23)*. Association for Computing Machinery, New York, NY, USA, 35–41. https://doi.org/10.1145/3609021.3609306

[15] Salvatore D'Oro, Michele Polese, Leonardo Bonati, Hai Cheng, and Tommaso Melodia. 2022. dApps: Distributed Applications for Real-Time Inference and Control in O-RAN. *IEEE Communications Magazine* 60, 11 (2022), 52–58. https://doi.org/10.1109/MCOM.002.2200079

[16] Harish Kumar Dureppagari, Ujwal Dinesha, Raini Wu, Santosh Ganji, Woo-Hyun Ko, Srinivas Shakkottai, and Dinesh Bharadia. 2022. Realtime intelligent control for NextG cellular radio access networks. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services* (Portland, Oregon) *(MobiSys '22)*. Association for Computing Machinery, New York, NY, USA, 567–568. https://doi.org/10.1145/3498361.3538787

[17] Juan José López Escobar, Rebeca P. Díaz Redondo, and Felipe J. Gil-Castiñeira. 2024. Unleashing the power of decentralized serverless IoT dataflow architecture for the Cloud-to-Edge Continuum: a performance comparison. *Annals of Telecommunications* 79 (2024), 135–148. https://api.semanticscholar.org/CorpusID:266934690

[18] ETSI. 2017. *Study on new radio access technology: Radio access architecture and interfaces.* Technical Report TR 138 912. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_tr/138900_138999/138912/14.01.00_60/tr_138912v140100p.pdf Version: 14.1.0.

[19] O-RAN Alliance e.V. 2018. O-RAN Alliance. https://www.o-ran.org/. [Online; Last accessed: 25-Jun-2024].

[20] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. 2023. Taking 5G RAN Analytics and Control to a New Level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. Association for Computing Machinery, New York, NY, USA, Article 1, 16 pages. https://doi.org/10.1145/3570361.3592493

[21] Jorge Gallego-Madrid, Irene Bru-Santa, Alvaro Ruiz-Rodenas, Ramon Sanchez-Iborra, and Antonio Skarmeta. 2024. Machine learning-powered traffic processing in commodity hardware with eBPF. *Computer Networks* 243 (2024), 110295. https://doi.org/10.1016/j.comnet.2024.110295

[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] Google. 2001. Protocol Buffers. https://protobuf.dev/. [Online; Last accessed: 30-May-2024].

[24] NIS Cooperation Group. 2019. *EU coordinated risk assessment of the cybersecurity of 5G networks.* Technical report. European Commission. https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=62132 [Online; Last accessed: 25-Jun-2024].

[25] Andreas Haas, Andreas Rossberg, Derek L Schuff, Ben L Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. 2017. Bringing the web up to speed with WebAssembly. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*. Association for Computing Machinery, New York, NY, USA, 185–200. https://doi.org/10.1145/3062341.3062363

[26] Adam Hall and Umakishore Ramachandran. 2019. An execution model for serverless functions at the edge. In *Proceedings of the International Conference on Internet of Things Design and Implementation* (Montreal, Quebec, Canada) *(IoTDI '19)*. Association for Computing Machinery, New York, NY, USA, 225–236. https://doi.org/10.1145/3302505.3310084

[27] Mutasem Q. Hamdan, Haeyoung Lee, Dionysia Triantafyllopoulou, Rúben Borralho, Abdulkadir Kose, Esmaeil Amiri, David Mulvey, Wenjuan Yu, Rafik Zitouni, Riccardo Pozza, Bernie Hunt, Hamidreza Bagheri, Chuan Heng Foh, Fabien Heliot, Gaojie Chen, Pei Xiao, Ning Wang, and Rahim Tafazolli. 2023. Recent Advances in Machine Learning for Network Automation in the O-RAN. *Sensors* 23, 21 (2023), 8792. https://doi.org/10.3390/s23218792

[28] Takanori Hara and Masahiro Sasabe. 2024. Practicality of in-kernel/user-space packet processing empowered by lightweight neural network and decision tree. *Computer Networks* 240 (2024), 110188. https://doi.org/10.1016/j.comnet.2024.110188

[29] Andreas Hegers. 2016. What Went Wrong On Our (Too) Long Journey to Open Networks? https://www.sdxcentral.com/articles/contributed/what-went-wrong-long-journey-open-networks/2016/11/. [Online; Last accessed: 05-Jun-2024].

[30] Chung-Hsing Hsu and Ulrich Kremer. 1998. IPERF: A framework for automatic construction of performance prediction models. In *Workshop on Profile and Feedback-Directed Compilation (PFDC)*. IEEE Computer Society, Paris, France.

[31] Wenjun Huang and Marcus Paradies. 2021. An Evaluation of WebAssembly and eBPF as Offloading Mechanisms in the Context of Computational Storage. arXiv:2111.01947 [cs.AR]

[32] Cheng-Feng Hung, You-Run Chen, Chi-Heng Tseng, and Shin-Ming Cheng. 2024. Security Threats to xApps Access Control and E2 Interface in O-RAN. *IEEE Open Journal of the Communications Society* 5 (2024), 1197–1203. https://doi.org/10.1109/OJCOMS.2024.3364840

[33] Mordor Intelligence. 2024. *United States Mobile Virtual Network Operator (MVNO) Market.* Market research. Mordor Intelligence. https://www.mordorintelligence.com/industry-reports/united-states-mobile-virtual-network-operator-mvno-market [Online; Last accessed: 26-Jun-2024].

[34] Abhinav Jangda, Bobby Powers, Emery D. Berger, and Arjun Guha. 2019. Not So Fast: Analyzing the Performance of WebAssembly vs. Native Code. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. USENIX Association, Renton, WA, 107–120. https://www.usenix.org/conference/atc19/presentation/jangda

[35] Wolfgang John, Ali Balador, Jalil Taghia, Andreas Johnsson, Johan Sjöberg, Ian Marsh, Jonas Gustafsson, Federico Tonini, Paolo Monti, Pontus Sköldström, and Jim Dowling. 2023. ANIARA Project - Automation of Network Edge Infrastructure and Applications with Artificial Intelligence. *Ada Lett.* 42, 2 (Apr 2023), 92–95. https://doi.org/10.1145/3591335.3591347

[36] Björn Karlsson. 2005. *Beyond the C++ Standard Library: An Introduction to Boost.* Pearson Education, Upper Saddle River, NJ, USA.

[37] Hoon Kim, Keunyoung Kim, Youngnam Han, and Sangboh Yun. 2004. A proportional fair scheduling for multicarrier transmission systems. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, Vol. 1. Institute of Electrical and Electronics Engineers (IEEE), Los Angeles, CA, USA, 409–413 Vol. 1. https://doi.org/10.1109/VETECF.2004.1400034

[38] Sean Kinney. 2023. Two big Open RAN successes, and two big problems. https://www.rcrwireless.com/20231003/fundamentals/two-big-open-ran-successes-and-two-big-problems [Online; Last accessed: 25-June-2024].

[39] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. 2023. Demo: EdgeRIC: Delivering Realtime RAN Intelligence. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 1162–1164. https://doi.org/10.1145/3603269.3610867

[40] Ravi Kokku, Rajesh Mahindra, Honghai Zhang, and Sampath Rangarajan. 2010. NVS: a virtualization substrate for WiMAX networks. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking* (Chicago, Illinois, USA) *(MobiCom '10)*. Association for Computing Machinery, New York, NY, USA, 233–244. https://doi.org/10.1145/1859995.1860023

[41] István Koren. 2021. A Standalone WebAssembly Development Environment for the Internet of Things. In *Web Engineering*, Marco Brambilla, Richard Chbeir, Flavius Frasincar, and Ioana Manolescu (Eds.). Springer International Publishing, Cham, 353–360.

[42] Georgiy Krylov, Petar Jelenković, Mark Thom, Gerhard W. Dueck, Kenneth B. Kent, Younes Manton, and Daryl Maier. 2021. Ahead-of-time compilation in eclipse OMR on example of WebAssembly. In *Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering* (Toronto, Canada) *(CASCON '21)*. IBM Corp., USA, 237–243.

[43] Sukchan Lee. 2017. Open5GS. https://open5gs.org/. [Online; Last accessed: 31-May-2024].

[44] Borui Li, Hongchang Fan, Yi Gao, and Wei Dong. 2022. Bringing webassembly to resource-constrained iot devices for seamless device-cloud integration. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services* (Portland, Oregon) *(MobiSys '22)*. Association for Computing Machinery, New York, NY, USA, 261–272. https://doi.org/10.1145/3498361.3538922

[45] Soo Yee Lim, Xueyuan Han, and Thomas Pasquier. 2023. Unleashing Unprivileged eBPF Potential with Dynamic Sandboxing. In *Proceedings of the 1st Workshop on EBPF and Kernel Extensions* (New York, NY, USA) *(eBPF '23)*. Association for Computing Machinery, New York, NY, USA, 42–48. https://doi.org/10.1145/3609021.3609301

[46] LitePoint. 2023. O-RAN: Challenges and Prospects on the Road to Maturity. https://www.litepoint.com/blog/open-ran-challenges-and-prospects-on-the-road-to-maturity/ [Online; Last accessed: 25-June-2024].

[47] Zhibo Liu, Dongwei Xiao, Zongjie Li, Shuai Wang, and Wei Meng. 2023. Exploring Missed Optimizations in WebAssembly Optimizers. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis* (Seattle, WA, USA) *(ISSTA 2023)*. Association for Computing Machinery, New York, NY, USA, 436–448. https://doi.org/10.1145/3597926.3598068

[48] Ju Long, Hung-Ying Tai, Shen-Ta Hsieh, and Michael Juntao Yuan. 2021. A Lightweight Design for Serverless Function as a Service. *IEEE Software* 38, 1 (2021), 75–80. https://doi.org/10.1109/MS.2020.3028991

[49] Hamid Mcheick, Heni Dhiab, Mohamad Dbouk, and Rakan Mcheik. 2010. Detecting type errors and secure coding in C/C++ applications. In *ACS/IEEE International Conference on Computer Systems and Applications - AICCSA 2010*. IEEE Computer Society, Hammamet, Tunisia, 1–9. https://doi.org/10.1109/AICCSA.2010.5587027

[50] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review* 38, 2 (2008), 69–74.

[51] Jämes Ménétrey, Marcelo Pasin, Pascal Felber, and Valerio Schiavoni. 2022. WebAssembly as a Common Layer for the Cloud-edge Continuum. In *Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge* (Minneapolis, MN, USA) *(FRAME '22)*. Association for Computing Machinery, New York, NY, USA, 3–8. https://doi.org/10.1145/3526059.3533618

[52] Dan Meyer. 2023. Why AT&T picked Ericsson for open RAN deployment. https://www.sdxcentral.com/articles/analysis/why-att-picked-ericsson-for-open-ran-deployment/2023/12/ [Online; Last accessed: 25-May-2024].

[53] Sebastiano Miano, Xiaoqi Chen, Ran Ben Basat, and Gianni Antichi. 2023. Fast In-kernel Traffic Sketching in eBPF. *SIGCOMM Comput. Commun. Rev.* 53, 1 (apr 2023), 3–13. https://doi.org/10.1145/3594255.3594256

[54] Iain Morris. 2024. AT&T, all in with Ericsson, seems to have shut the door to xApps. https://www.lightreading.com/open-ran/at-t-all-in-with-ericsson-seems-to-have-shut-the-door-to-xapps [Online; Last accessed: 03-June-2024].

[55] Iain Morris. 2024. Let's stop pretending open RAN is in good health. https://www.lightreading.com/open-ran/let-s-stop-

pretending-open-ran-is-in-good-health [Online; Last accessed: 18-Oct-2024].

[56] Iain Morris. 2024. Single-vendor open RAN is spreading like a virus. https://www.lightreading.com/open-ran/single-vendor-open-ran-is-spreading-like-a-virus [Online; Last accessed: 05-June-2024].

[57] Eric Niebler. 2006. *Boost accumulators*. Boost C++ Libraries, New York, NY, USA.

[58] Navid Nikaein, Mahesh K. Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. 2014. OpenAirInterface: A Flexible Platform for 5G Research. *SIGCOMM Comput. Commun. Rev.* 44, 5 (oct 2014), 33–38. https://doi.org/10.1145/2677046.2677053

[59] Nokia. 2024. Nokia and Vodafone complete successful Open RAN trial in Italy. https://www.nokia.com/about-us/news/releases/2024/04/30/nokia-and-vodafone-complete-successful-open-ran-trial-in-italy/ [Online; Last accessed: 29-May-2024].

[60] Federico Parola, Sebastiano Miano, and Fulvio Risso. 2021. A proof-of-concept 5G mobile gateway with eBPF. In *Proceedings of the SIGCOMM '20 Poster and Demo Sessions* (Virtual event) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 68–69. https://doi.org/10.1145/3405837.3411395

[61] Federico Parola, Fulvio Risso, and Sebastiano Miano. 2021. Providing Telco-oriented Network Services with eBPF: the Case for a 5G Mobile Gateway. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. Institute of Electrical and Electronics Engineers (IEEE), Tokyo, Japan, 221–225. https://doi.org/10.1109/NetSoft51509.2021.9492571

[62] Rémy Pascal. 2023. *Market Landscape: RAN Vendors 2023*. Market research. Omdia. https://omdia.tech.informa.com/om030391/market-landscape-ran-vendors-2023 [Online; Last accessed: 24-Jun-2024].

[63] Margaret H. Pinson, Mark Poletti, Julie Kub, Jeremy Glenn, Tim Thompson, Robert Kupsh, Naser Areqat, Okmar Dharmadhikari, Annie George, T. Lauriston Hardin, Cory Johnson, Spiros Kapoulas, and Sundar Sriram. 2023. *5G Challenge Preliminary Event: Evaluating Modular, Interoperable, Multi-Vendor, Open RAN Solutions*. NTIA Technical Memorandum 23-568. NTIA. https://its.ntia.gov/umbraco/surface/download/publication?reportNumber=TM-23-568.pdf [Online; Last accessed: 24-Jun-2024].

[64] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411. https://doi.org/10.1109/COMST.2023.3239220

[65] Allan Rasmussen. 2018. The state of MVNO in 2018 – More than 1300 active MVNOs in 80 countries. https://www.yozzo.com/industry-news/mvna-mvne-mvno/the-state-of-mvno-in-2018-more-than-1300-active-mvnos-in-80-countries/. [Online; Last accessed: 28-May-2024].

[66] Partha Pratim Ray. 2023. An Overview of WebAssembly for IoT: Background, Tools, State-of-the-Art, Challenges, and Future Directions. *Future Internet* 15, 8 (2023), 275. https://doi.org/10.3390/fi15080275

[67] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (feb 1978), 120–126. https://doi.org/10.1145/359340.359342

[68] Martin Rowe. 2023. Open RAN: Interoperability is not plug-and-play. https://www.5gtechnologyworld.com/open-ran-interoperability-is-not-plug-and-play/ [Online; Last accessed: 29-May-2024].

[69] Samsung. 2022. Vodafone UK and Samsung Switch on First 5G Open RAN Site in the United Kingdom. https://news.samsung.com/global/vodafone-uk-and-samsung-switch-on-first-5g-open-ran-site-in-the-united-kingdom. [Online; Last accessed: 05-Jun-2024].

[70] Egemen Sarikaya and Ertan Onur. 2021. Placement of 5G RAN Slices in Multi-tier O-RAN 5G Networks with Flexible Functional Splits. In *2021 17th International Conference on Network and Service Management (CNSM)*. International Federation for Information Processing, Laxenburg, Austria, 274–282. https://doi.org/10.23919/CNSM52442.2021.9615541

[71] Christian Scheich, Marius Corici, Hauke Buhr, and Thomas Magedanz. 2023. eXpress Data Path Extensions for High-Capacity 5G User Plane Functions. In *Proceedings of the 1st Workshop on EBPF and Kernel Extensions* (New York, NY, USA) *(eBPF '23)*. Association for Computing Machinery, New York, NY, USA, 86–88. https://doi.org/10.1145/3609021.3609298

[72] Telecoms Security and Resilience Team. 2019. *UK Telecoms Supply Chain Review Report*. Technical report. Department for Digital, Culture, Media & Sport. https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=62132 [Online; Last accessed: 25-Jun-2024].

[73] Salim S.I and Richard Lin. 2023. Opening Critical Infrastructure: The Current State of Open RAN Security. https://www.trendmicro.com/en_us/research/23/l/the-current-state-of-open-ran-security.html. [Online; Last accessed: 29-May-2024].

[74] David Soldani, Petrit Nahi, Hami Bour, Saber Jafarizadeh, Mohammed F. Soliman, Leonardo Di Giovanna, Francesco Monaco, Giuseppe Ognibene, and Fulvio Risso. 2023. eBPF: A New Approach to Cloud-Native Observability, Networking and Security for Current (5G) and Future Mobile Networks (6G and Beyond). *IEEE Access* 11 (2023), 57174–57202. https://doi.org/10.1109/ACCESS.2023.3281480

[75] Gabor Soos, Dániel Ficzere, Tamás Seres, Sándor Veress, and István Németh. 2020. Business opportunities and evaluation of non-public 5G cellular networks - a survey. *Infocommunications Journal* VOLUME XII (12 2020), 31–38. https://doi.org/10.36244/ICJ.2020.3.5

[76] Software Radio Systems. 2022. srsRAN Project. https://github.com/srsran/srsRAN_Project. [Online; Last accessed: 30-May-2024].

[77] Software Radio Systems. 2023. O-RAN NearRT-RIC and xApp. https://docs.srsran.com/projects/project/en/latest/tutorials/source/near-rt-ric/source/index.html [Online; Last accessed: 29-May-2024].

[78] Telecoms Diversification Taskforce. 2021. *Findings and report*. Independent report. UK Government. https://assets.publishing.service.gov.uk/media/60644773e90e074e57968b72/April_2021_Telecoms_Diversification_Taskforce_Findings_and_Report_v2.pdf Pag. 16.

[79] Marcos A. M. Vieira, Matheus S. Castanho, Racyus D. G. Pacífico, Elerson R. S. Santos, Eduardo P. M. Câmara Júnior, and Luiz F. M. Vieira. 2020. Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications. *ACM Comput. Surv.* 53, 1, Article 16 (feb 2020), 36 pages. https://doi.org/10.1145/3371038

[80] Vodafone and DOCOMO. 2023. *O-RU/O-DU Integration Challenges in an Open RAN Environment*. White paper. Vodafone & DOCOMO. https://assets.ctfassets.net/q7ob9vms4z5k/3Y6OYGRe7bJvUDLs5TUwff/01048637b951bd9b71059212e9541f7c/docomo-whitepaper.pdf [Online; Last accessed: 29-May-2024].

[81] Elliott Wen and Gerald Weber. 2020. Wasmachine: Bring IoT up to Speed with A WebAssembly OS. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. Institute of Electrical and Electronics Engineers (IEEE), Austin, TX, USA, 1–4. https://doi.org/10.1109/PerComWorkshops48775.2020.9156135

[82] O-RAN WG1. 2024. *O-RAN Architecture Description (O-RAN.WG1.OAD-R003-v08.00)*. Technical Specification TS 103 982. 3GPP, Sophia Antipolis, France. https://www.etsi.org/deliver/etsi_ts/103900_103999/103982/08.00.00_60/ts_103982v080000p.pdf Version: 8.0.0.

[83] Tom Wheeler and David Simpson. 2022. *5G is Smart, Now Let's Make it Secure*. Brookings, NW, Washington, DC.

[84] Vinay Yadhav., Andrew Williams., Ondrej Smid., Jimmy Kjällman., Raihan Islam., Joacim Halén., and Wolfgang John. 2024. Benefits of Dynamic Computational Offloading for Mobile Devices. In *Proceedings of the 14th International Conference on Cloud Computing and Services Science - CLOSER*. INSTICC, SciTePress, Angers, France, 265–276. https://doi.org/10.5220/0012719800003711

[85] Yutian Yan, Tengfei Tu, Lijian Zhao, Yuchen Zhou, and Weihang Wang. 2021. Understanding the performance of webassembly applications. In *Proceedings of the 21st ACM Internet Measurement Conference* (Virtual Event) *(IMC '21)*. Association for Computing Machinery, New York, NY, USA, 533–549. https://doi.org/10.1145/3487552.3487827

[86] Michael Yuan. 2021. eBPF and WebAssembly: whose VM reigns supreme? https://medium.com/codex/ebpf-and-webassembly-whose-vm-reigns-supreme-c2861ce08f89 [Online; Last accessed: 13-June-2024].

[87] Frederik Jonathan Zumegen, Ish Kumar Jain, and Dinesh Bharadia. 2024. BeamArmor: Seamless Anti-Jamming in 5G Cellular Networks with MIMO Null-steering. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications* (San Diego, CA, USA) *(HOTMOBILE '24)*. Association for Computing Machinery, New York, NY, USA, 121–126. https://doi.org/10.1145/3638550.3641138