Accelerating Inter-App Communications for Time-Critical O-RAN Control Loops

Lorenzo Rosa, *Member, IEEE*, Andrea Garbugli, *Member, IEEE*, Domenico Scotece, *Member, IEEE*, and Luca Foschini, *Senior Member, IEEE*

***Abstract*— The Open Radio Access Network (O-RAN) is reshaping cellular architectures through disaggregation, openness, and programmability, enabling intelligent control across modular RAN components. A growing ecosystem of user-defined control-plane applications, namely rApps, xApps, and the emerging dApps, operates at different timescales and unlocks advanced control loop capabilities, but introduces diverse and stringent Quality of Service (QoS) requirements for communication. Current implementations typically rely on cloud-based messaging systems, which privilege transparency and ease of use for developers and limit support for time-sensitive workloads. In this letter, we propose that O-RAN control loops base their internal communication on INSANE, a cloud-native, data-centric middleware that supports multiple networking stacks, including kernel-bypass option such as eBPF XDP, DPDK, and RDMA. Our preliminary evaluation shows that INSANE achieves nearly 2× higher throughput than widely used alternatives, while reducing 99.9th-percentile latency of over an order of magnitude for small messages. At the same time, INSANE preserves a uniform and easy-to-use programming interface. These results highlight INSANE as a promising foundation for faster, more predictable control loops, a significant step toward the ultimate goal of enabling AI-driven RAN optimizations in O-RAN systems.**

***Index Terms*— O-RAN, dApps, rApps, xApps, kernel-bypass.**

I. Introduction

THE Open Radio Access Network (O-RAN) paradigm is transforming cellular network architectures, promoting openness, flexibility, and intelligence within the Radio Access Network (RAN). This shift involves disaggregating traditional, vertically integrated RAN components into modular, standardized elements such as the O-RAN Central Unit (O-CU), O-RAN Distributed Unit (O-DU), and O-RAN Radio Unit (O-RU). These elements are orchestrated via open interfaces and controlled by software-defined RAN Intelligent Controllers (RICs), fostering interoperability and enabling data-driven optimization. This evolution is seen as a foundational step toward future 6G networks, with standardization bodies such as 3GPP already engaging in exploratory efforts [1].

Received 2 February 2026; accepted 27 February 2026. Date of publication 4 March 2026; date of current version 1 April 2026. This work was supported in part by European Union under Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001-Program "RESTART") under Grant CUP: J33C22002880001. The associate editor coordinating the review of this article and approving it for publication was C. Skianis. *(Corresponding author: Domenico Scotece.)*

The authors are with the Department of Computer Science and Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: lorenzo.rosa@unibo.it; andrea.garbugli@unibo.it; domenico.scotece@unibo.it; luca.foschini@unibo.it).

Digital Object Identifier 10.1109/LNET.2026.3670470

At the core of the O-RAN architecture are user-defined control-plane applications, namely rApps, xApps, and the recently proposed dApps [2], operating on distinct timescales. rApps run in the Non-Real Time RIC (Non-RT RIC), performing policy-level operations at multi-second intervals. xApps, deployed in the Near-Real Time RIC (Near-RT RIC), handle latency-sensitive control in the order of milliseconds. dApps, hosted directly in the RAN, enable sub-ms control over low-level functions by processing user-plane data and enabling AI-driven control loops near the radio edge [3]. These applications are interconnected by a set of standardized interfaces, but also rely on *Internal Messaging Infrastructures* (IMIs) that support registration, discovery, routing, and message dissemination within and across layers composing the control plane.

Despite this structured layering, the current implementation landscape for intra-O-RAN communication remains highly fragmented, with each layer adopting distinct communication platforms and transport protocols tailored to its specific performance needs. In practice, many implementations rely on standard cloud-based messaging systems such as Nanomsg (used internally by the RIC Message Router, RMR [4]), ZeroMQ, and NATS, which are easy to use and generally sufficient to meet nominal latency requirements. However, these solutions struggle to support the most latency-critical control loops. This limitation stems from their reliance on the traditional OS networking stack and their lack of support for modern kernel-bypass techniques, such as DPDK [5] and eBPF XDP [6], which are increasingly required in private edge deployments to achieve lower latency and higher throughput. While effective, these techniques remain difficult to use and are poorly integrated with standard cloud-native tools [7], [8]. As a result, this fragmentation hinders unified system-level QoS enforcement and significantly increases the development and integration complexity of O-RAN applications.

This letter proposes the adoption of INSANE, a unified, cloud-native communication framework [9], for internal communication among O-RAN applications, with the goal of enabling faster and more predictable control loops. We envision INSANE as an alternative communication backend for the internal IMIs of the Non-RT RIC, Near-RT RIC, and RAN nodes, allowing its integration within existing O-RAN software components and workflows. Our approach preserves the ease of use and developer experience of existing messaging systems, while transparently improving communication performance through the option to adopt kernel-bypass techniques. A preliminary performance evaluation shows that INSANE can outperform currently adopted messaging systems while

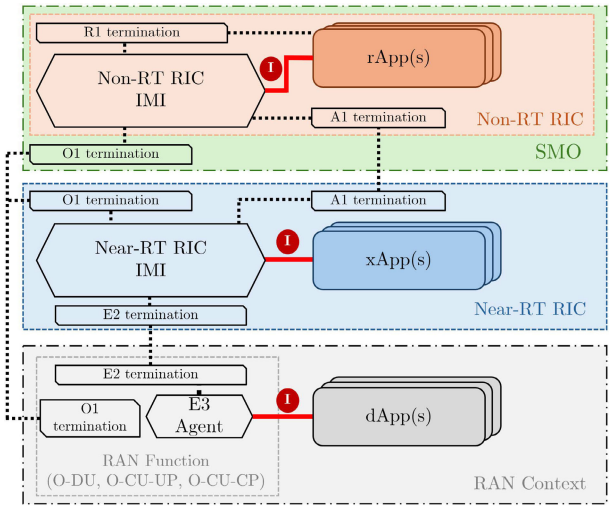

Fig. 1. INSANE positioning in the O-RAN architecture encircled *I*. The solid red lines represent intra-layer paths where INSANE QoS differentiation can be used.

TABLE I
COMPARISON OF O-RAN APPLICATION TYPES

App Type	Timescale	Positioning	Messaging Infra
rApps	> 1 s	Non-RT RIC	Non-RT RIC IMI
xApps	10ms – 1s	Near-RT RIC	Near-RT RIC IMI
dApps	< 10 ms	RAN Context	E3 Agent

supporting the diverse requirements of O-RAN applications within a single, portable abstraction.

II. BACKGROUND AND RELATED WORK

This Section provides background on two foundational aspects of our work: the O-RAN architecture and its communication infrastructure (Sec.II-A), and kernel-bypass networking techniques (Sec. II-B).

A. O-RAN Architecture, Applications, and Communication

O-RAN introduces a modular, interoperable, and software-defined architecture (Fig. 1) that disaggregates traditional RAN functions into a microservice-based ecosystem of independently developed applications. Central to this design are the Non-RT RIC and the Near-RT RIC, operating at different timescales and hosting user-defined rApps and xApps that enable data-driven, closed-loop RAN control [1] (Table I). More recently, dApps have been introduced to support ultra-low-latency control by running directly on O-CUs and O-DUs and accessing raw user-plane data [2]. Together, rApps, xApps, and dApps span a wide range of control objectives and timing constraints, from long-term optimization to fine-grained, real-time adaptation, requiring frequent and efficient inter-application communication [3].

To support this interaction, O-RAN relies on IMIs to distribute data among applications within each control layer. However, IMIs are only loosely specified, and their realization is highly heterogeneous in practice. The O-RAN Software Community (OSC) implements the near-RT RIC IMI through the RAN Message Router (RMR), which is based on communication backends such as NNG [4]. At the RAN layer, no standardized IMI exists; recent proposals introduce components such as the E3 Agent to mediate communication between dApps and higher-level control functions, typically implemented using low-level POSIX sockets or middleware like NATS and ZeroMQ [3]. Hence, current O-RAN deployments employ multiple messaging systems with similar performance features but limited ability to differentiate communication behavior according to application-specific QoS requirements.

Overall, the implementation landscape of internal O-RAN communication remains fragmented, with different layers adopting distinct mechanisms to satisfy specific QoS requirements. While a unified approach would improve consistency and maintainability, none of the existing messaging solutions provides a clear advantage in handling heterogeneous QoS demands, as they all ultimately rely on the same OS-based networking stack. This common reliance fundamentally limits their ability to differentiate network support and achieve the performance required by time-sensitive O-RAN control loops.

B. Kernel-Bypass Networking

As network traffic volumes and link speeds increase, traditional OS-based networking stacks, designed for generality rather than performance, struggle to keep up, particularly in latency-sensitive environments such as the edge [8]. To address this issue, various kernel-bypass technologies have emerged, including the Data Plane Development Kit (DPDK) [5] and eBPF eXpress Data Path (XDP) [6], which allow applications to interact directly with NIC driver, bypassing the kernel to achieve lower latency and higher throughput. These techniques are especially relevant in private edge environments, where operators have direct control over hardware and can exploit acceleration options unavailable in the public cloud [10]. However, despite their performance benefits, these solutions remain difficult to integrate into modern cloud-native stacks due to their complexity, limited compatibility with container orchestration, and lack of standard interfaces [7].

To simplify access to these technologies, several recent systems aim to abstract kernel-bypass mechanisms through higher-level interfaces. Demikernel [11] and Pegasus [12] offer improved usability but either focus on lower-level APIs, or require intrusive architectural changes. In contrast, INSANE [9] stands out as a messaging system that enables fine-grained QoS differentiation without sacrificing cloud compatibility or developer productivity, making it uniquely suitable for heterogeneous O-RAN workloads (see Section III).

III. QOS-AWARE COMMUNICATION SUPPORT FOR O-RAN CONTROL LOOPS

Current O-RAN implementations adopt heterogeneous communication platforms and protocols for internal messaging within each architectural layer, in order to meet the diverse performance requirements of RAN applications. This fragmentation complicates development, as developers must navigate multiple APIs, software stacks, and transport mechanisms even within a single layer. Critically, none of the commonly adopted

internal communication frameworks support integration with modern network acceleration techniques. As a result, despite advanced orchestration and scheduling strategies, enforcing communication QoS for latency-sensitive control loops remains difficult, particularly in cloud-based deployments.

To address this gap, we propose the adoption of INSANE within the IMI of each O-RAN layer, supporting the internal inter-app communication (red lines in Fig. 1). This implementation enables uniform and QoS-aware communication management across the O-RAN architecture, with a twofold goal. First, to simplify the development of microservice-based O-RAN applications, by providing a unified communication interface that abstracts over the time scale of operations and decouples rApps, xApps, and dApps from specific RAN implementations. Second, to enable system-level QoS differentiation by supporting multiple networking stacks, including kernel-bypass techniques, to meet the diverse latency requirements of O-RAN control loops.

INSANE is a cloud-native messaging system that enables system-level control of the communication QoS. Designed for high performance networking at microsecond scale, INSANE runs as a userspace service and integrates with platforms such as Kubernetes through a Container Network Interface (CNI). Applications interact with INSANE via a multi-language, data-centric API that abstracts transport details, allowing developers to focus on data semantics (e.g., topic names) rather than networking logic (e.g., IP addresses). To this end, INSANE translates high-level *QoS policies* into dynamic backend selection: a modular plugin system enables support for acceleration technologies such as eBPF XDP, DPDK, and shared-memory IPC, allowing per-flow backend selection based on QoS needs. In turn, each plugin supports multiple transport protocols, including UDP, TCP, and SCTP. The internal *zero-copy* data path internalizes advanced networking functions while hiding complexity to the end users. This design enables INSANE to achieve low latency, high throughput, and resource constrained communication within the same system while simplifying the development process.

We envision the adoption of INSANE as the backend messaging system for the internal messaging interfaces (IMIs) within O-RAN components, replacing the currently adopted solutions. This approach simplifies the development and maintenance of rApps, xApps, and dApps, while enabling finer-grained QoS control through INSANE's selectable communication plugins. At the same time, the simplicity and ease of use of the INSANE API preserve application portability, allowing existing implementations to be adapted with minimal programming effort.

The key advantage of this approach is the ability to explicitly identify distinct traffic flows and to support them with differentiated QoS levels, directly mapped to appropriate system-level networking backends. In this design, communication among non-critical applications can be routed through the INSANE *slow path*, which relies on the standard OS-based network stack to minimize resource usage. Conversely, time-sensitive control loops, such as those typical of dApps operating at sub-millisecond timescales, as well as selected xApps, can leverage INSANE's high-performance path based on DPDK. This selective use of kernel-bypass networking has the potential to significantly reduce communication latency and increase throughput, thereby enabling more responsive real-time control loops in the RAN.

The main challenge to fully integrate INSANE in the O-RAN context is that it must satisfy the standard specifications of the IMIs. To this end, we extended INSANE to provide RESTful APIs that enable external management by the SMO via the O1 interface. This interface enables exposure of runtime communication metrics across RICs and the RAN, and allows dynamic adjustment of QoS levels, such as upgrading service quality for a specific user slice when resources permit or downgrading it under load. Additionally, this API could support other essential functions required by the near-RT RIC specification, including subscription control, conflict resolution, security enforcement, and interaction with the shared data layer [1]. Finally, INSANE does not replace standardized O-RAN interfaces; rather, it complements them by accelerating the underlying communication mechanisms and execution paths, thus improving responsiveness and scalability without altering the functional semantics defined by the O-RAN.

In the following section, we evaluate INSANE in an O-RAN setting with diverse communication patterns. Our results show that INSANE matches existing messaging systems on its low-overhead *slow* path, while substantially outperforming them when using its high-performance *fast* path, thus representing a promising, flexible, and easy-to-use solution for effectively supporting diverse communication QoS requirements.

IV. Performance Evaluation

This evaluation assesses whether INSANE can replace currently adopted messaging systems for internal O-RAN communication, while enabling differentiated QoS through a unified and easy-to-use API. First, we consider three representative systems commonly used in practice: ZeroMQ, NATS, and NNG (formerly the backend for RMR), and we highlight that their performance is largely comparable due to reliance on the standard OS networking stack. Then, we compare them against INSANE in both its socket-based profile, for low-overhead non-critical traffic, and its DPDK configuration, which enables kernel-bypass without specialized hardware. All systems are tested using a pair of Python applications mimicking O-RAN control-plane communication, with varying payload sizes to reflect different workloads; all experiments use TCP, as required by the baselines. We also measure resource consumption to capture the cost of higher performance.

Two benchmarks were conducted: an echo test to measure round-trip time (RTT) and expose communication overhead, and a stress test to evaluate achievable throughput by measuring sustained goodput. Experiments were performed on a private cloud deployed as a Kubernetes cluster, with INSANE running as a network CNI. Each application executed on a separate worker node on distinct hosts to capture worst-case protocol overhead. Each node was equipped with 8 vCPUs, 16 GB of RAM, and connected via a dedicated physical slice (SR-IOV) of a Mellanox DX-6 100 GB NIC.

Figure 2 reports latency results as Cumulative Distribution Functions (CDFs) for different payload sizes. ZeroMQ,

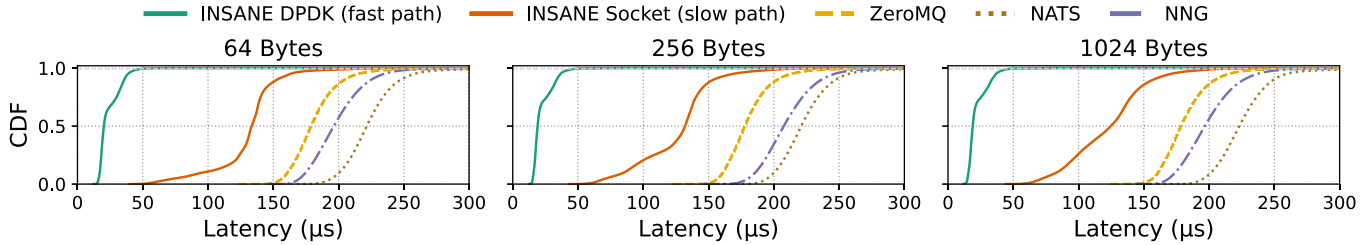

Fig. 2. Latency CDF comparison between INSANE (slow and fast path) and the most commonly adopted messaging systems for O-RAN inter-app communication.

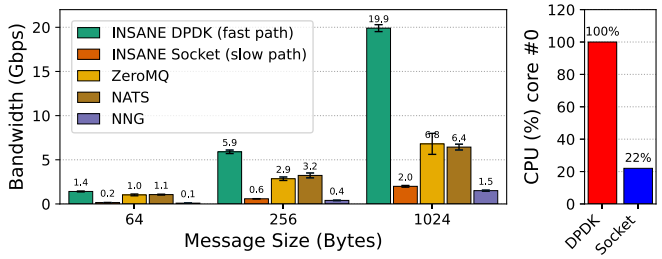

Fig. 3. Bandwidth comparison and average CPU usage.

NATS, and NNG exhibit closely clustered distributions, indicating similar latency behavior and establishing a baseline for socket-based messaging systems. INSANE's DPDK-based fast path shows consistently lower latency across the entire distribution, with a markedly steeper CDF. The socket-based INSANE configuration lies between these extremes, achieving lower latency than the baseline systems.

Throughput results in Fig. 3 (left) follow a similar trend. ZeroMQ, NATS, and NNG achieve comparable throughput, reflecting the limits of OS-based networking, while INSANE's fast path attains the highest throughput by leveraging kernel-bypass mechanisms. Figure 2 (right) reports CPU utilization for INSANE configurations, showing that the higher performance of the DPDK fast path is associated with dedicating a full CPU core to busy polling, whereas the socket-based configuration has resource consumption comparable to the baseline systems. We note that the cost of a busy-polling core on each worker node is amortized across all the INSANE-based applications on the same node, which can share access to the high-performance data path. Moreover, the number of polling cores is configurable, allowing the system to balance parallelism and resource usage based on application demands.

Overall, these results illustrate how INSANE enables developers to select the networking stack that best matches the QoS requirements of a given control flow, achieving performance levels that are impossible to reach with existing systems, without requiring low-level expertise or significant changes to application logic. This performance analysis motivates our proposal to adopt INSANE as the internal inter-application communication backend in O-RAN systems.

V. Conclusion and Future Work

This letter presents our vision for unified communication support across O-RAN applications, addressing the limitation of current messaging systems to effectively differentiate communication QoS. By adopting INSANE as IMI at each layer, we propose to simplify application development while enabling explicit control over latency, throughput, and resource usage through selectable networking backends. Our evaluation indicates that INSANE can support diverse performance requirements through a single API, while exposing the necessary interfaces for integration with the O-RAN ecosystem. Future work includes integrating and evaluating INSANE as a backend for RMR and, in the longer term, exploring AI and ML pipelines built atop INSANE-powered dApps to further validate the approach.

References

[1] J. Luis Herrera, S. Montebugnoli, D. Scotece, L. Foschini, and P. Bellavista, "A tutorial on O-RAN deployment solutions for 5G: From simulation to emulated and real testbeds," *IEEE Commun. Surveys Tuts.*, vol. 28, pp. 1709–1748, 2026.

[2] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "DApps: Distributed applications for real-time inference and control in O-RAN," *IEEE Commun. Mag.*, vol. 60, no. 11, pp. 52–58, Nov. 2022.

[3] A. Lacava et al., "DApps: Enabling real-time AI-based open RAN control," *Comput. Netw.*, vol. 269, Sep. 2025, Art. no. 111342.

[4] O-RAN Software Community. (2025). *RIC Message Router Documentation*. Accessed: Jul. 2025. [Online]. Available: https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr

[5] Linux Foundation. (2023). *The Data Plane Development Kit*. [Online]. Available: www.dpdk.org

[6] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacífico, E. R. S. Santos, E. P. M. C. Júnior, and L. F. M. Vieira, "Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–36, Feb. 2020.

[7] L. Rosa, L. Foschini, and A. Corradi, "Empowering cloud computing with network acceleration: A survey," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 4, pp. 2729–2768, 2024.

[8] Q. Cai, S. Chaudhary, M. Vuppalapati, J. Hwang, and R. Agarwal, "Understanding host network stack overheads," in *Proc. ACM SIGCOMM Conf.* New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 65–77, doi: 10.1145/3452296.3472888.

[9] L. Rosa, A. Garbugli, A. Corradi, and P. Bellavista, "INSANE: A unified middleware for QoS-aware network acceleration in edge cloud computing," in *Proc. 24th Int. Middleware Conf. ZZZ*. New York, NY, USA: Association for Computing Machinery, Nov. 2023, pp. 57–70.

[10] The Linux Foundation. *State of the Edge Report 2023*. Accessed: Feb. 28, 2025. [Online]. Available: https://stateoftheedge.com/reports/state-of-the-edge-report-2023/

[11] I. Zhang et al., "The demikernel datapath OS architecture for microsecond-scale datacenter systems," in *Proc. ACM SIGOPS 28th Symp. Operating Syst. Princ.* New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 195–211.

[12] D. Peng, C. Liu, T. Palit, A. Vahldiek-Oberwagner, M. Vij, and P. Fonseca, "Pegasus: Transparent and unified kernel-bypass networking for fast local and remote communication," in *Proc. 20th Eur. Conf. Comput. Syst.* New York, NY, USA: Association for Computing Machinery, Mar. 2025, pp. 360–378.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement