Demonstrating Slice-Aware Latency Decomposition in Cloud-Native 5G Systems

Yassir Amami
Inria, Université Côte d'Azur
Sophia Antipolis, France
yassir.amami@inria.fr

Chadi Barakat
Inria, Université Côte d'Azur
Sophia Antipolis, France
chadi.barakat@inria.fr

Thierry Turletti
Inria, Université Côte d'Azur
Sophia Antipolis, France
thierry.turletti@inria.fr

***Abstract*—This demo presents a live slice-aware latency decomposition framework for open-source cloud-native 5G systems. It builds on our earlier RAN-observability work, which demonstrated reproducible deployment and real-time radio monitoring for open-source 5G experiments. The present demo extends that foundation with user-plane latency measurement. From a declarative experiment description, the framework instantiates the radio access network, 5G core, user equipment, traffic generation, monitoring stack, and artifact collection pipeline. A passive UE mapper reconstructs UE, slice, IP address, and GTP-U tunnel context as sessions appear, while a Kopf-based controller deploys eBPF probes on demand at the gNB and UPF. These probes derive latency samples from existing 5G user-plane packets and export them with UE, slice, TEID, direction, and probe-location labels. During the demo, attendees will observe UEs attaching, TCP traffic starting, labeled latency samples appearing, and a far-UE radio condition being identified through synchronized per-UE latency, slice throughput, and RAN metrics. Interested users can apply for access to our infrastructure to rerun the same scripted scenario, and the open-source framework can be adapted to different use cases, traffic patterns, and artifact-collection policies. The main message is that the system both measures latency inside the 5G user plane and associates that latency with the network context required to interpret it.**

***Index Terms*—5G; latency; network slicing; eBPF; observability**

I. Motivation

Open-source 5G experimentation gives researchers direct access to realistic radio and core behavior, but it also creates substantial operational complexity. A complete experiment may require provisioning physical machines, configuring radio and core functions, attaching UEs, generating traffic, enabling monitoring, collecting logs, and correlating measurements across several layers. When these steps are executed manually, results can depend on hidden setup choices rather than on the network and RAN scenario under study.

This demonstration continues our previous work on reproducible RAN observability for open-source 5G systems [5], [6]. That work exposed real time radio metrics in controlled interference experiments. The missing layer is latency decomposition: when user-plane delay increases, the operator needs to determine whether the increase is consistent with radio conditions, user-plane processing, or the path beyond the 5G core. This demo adds that layer by combining RAN observability with per-UE, per-slice, and per-anchor latency measurements.

The measurement method is described in a companion MSWiM 2026 research paper [8]; this demo focuses on making the method observable, reproducible, and interactive in a live 5G testbed. The demo is motivated by a practical limitation of current monitoring views: radio metrics, core metrics, infrastructure metrics, and packet observations are often available, but they do not by themselves explain which UE, slice, direction, or observation point a latency sample belongs to. Our system makes latency a 5G-contextual measurement. It passively derives RTT samples from existing GTP-U traffic, and immediately binds them to the runtime UE and slice context. End-to-end RTT can increase because of radio retransmissions, scheduling, UPF processing, host overload, GTP-U path effects, or delay at the external traffic server. The system helps identify which part of the experiment is consistent with the observed increase, and which UE or slice is affected.

II. System Demonstrated

The demonstrated platform is an Ansible-driven framework for reproducible 5G experimentation. It provisions selected testbed nodes, deploys the chosen open-source 5G stack, configures radio and UE components, starts traffic generators, and collects experiment artifacts. A scenario file specifies what should happen during an experiment: which UEs are used, which traffic directions are active, how long each phase lasts, whether perturbations are injected, and which artifacts should be collected. The same scenario can therefore be replayed or modified without changing the code.

The open-source repository[1] provides the deployment scripts, scenario descriptions, monitoring configuration, and artifact-collection tasks used in the demo. Interested users can apply for an account on our infrastructure and rerun the scenario shown here, or adapt the scripts to different use cases, UE sets, traffic patterns, perturbations, and artifact policies. Each run regenerates the experiment timeline, exported Prometheus samples, iperf logs, pod logs, and dashboard-ready metrics. Prometheus and Grafana provide the storage and visualization layer, while the collected signals include RAN metrics, core metrics, infrastructure metrics, traffic logs, pod logs, and scenario timelines. This gives attendees a complete

[1]https://github.com/sopnode/5g_ansible.

979-8-3195-0167-7/26/$31.00 ©2026 European Union

view of the experiment lifecycle rather than a single isolated metric, and reuses the deployment and monitoring foundation introduced in our earlier RAN-observability demo [6].

The latency decomposition feature adds three components to this framework. First, a passive UE mapper reconstructs the relationship between IMSI, UE IP address, selected slice, RAN UE identifier, and uplink/downlink GTP-U Tunnel Endpoint Identifiers (TEIDs). This mapping is updated as UEs attach, establish PDU sessions, and disconnect. Second, a Kopf-based Kubernetes controller uses this runtime context to deploy measurement probes on demand. Third, eBPF probes [7] attached through Linux traffic control parse existing GTP-U traffic at the gNB and UPF anchors, match TCP data packets with reverse acknowledgments, and export latency samples.

The runtime path shown in the demo works as follows: a scenario description launches the deployment; UE attachment and traffic create dynamic UE/slice context; the controller and eBPF probes transform packet observations into labeled latency samples; and Grafana plus exported artifacts provide live and replayable analysis. The probes do not generate active traffic. Each exported sample is labeled with the UE, slice, TEID, logical direction, mode, and probe role. Prometheus stores these measurements, while Grafana visualizes per-UE and per-slice latency together with radio metrics, throughput, and infrastructure counters. This joint view allows latency changes to be interpreted alongside core, radio and infrastructure metrics.

The demonstrated contribution is a slice-aware latency decomposition capability for cloud-native 5G systems. It computes latency from live 5G user-plane packets, reconstructs 5G session context, deploys packet-level probes at selected user-plane anchors, enriches the resulting samples with UE/slice/tunnel labels, and correlates them with RAN, core, and infrastructure measurements. Compared with our earlier RAN-focused work [5], [6], the new feature is the ability to interpret measured user-plane latency jointly with UE identity, slice, direction, TEID, and probe-location context. Deployment automation and artifact collection make the live scenario reproducible and auditable, while dynamic UE mapping and on-demand probes keep measurements aligned with the runtime state of the network. The exported artifacts include timelines, Prometheus samples, iperf logs, pod logs, and optional packet captures for later analysis, and the supporting automation is available through the same open-source repository.

III. Live Demonstration

Fig. 1 shows the physical and cloud-native testbed. The demo runs on the SophiaNode testbed, a site of SLICES-RI [4] that includes R2lab [3], an open wireless experimentation platform designed for reproducible research. The setup uses Open5GS [1], OpenAirInterface [2], COTS Quectel RM500Q-GL-based UEs labeled qhat01–qhat03 in the dashboard, and an iperf traffic server beyond the UPF. We first launch a reproducible scenario from a declarative experiment file. The framework prepares the selected nodes, starts the 5G deployment, attaches the UEs, enables monitoring, and records a

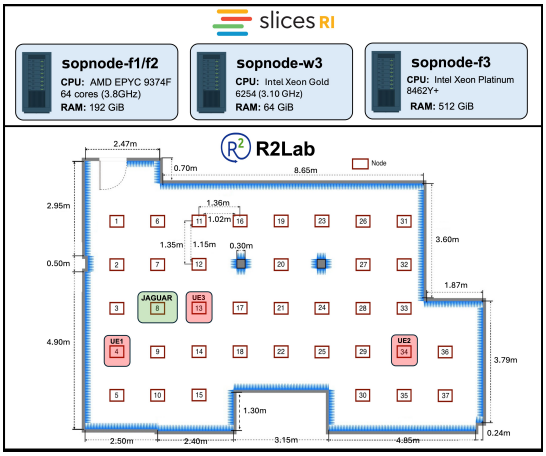


Fig. 1. Physical and cloud-native testbed. UE2 is farther from the RU than UE1 and UE3.

timeline of the run. Attendees then see the dynamic part of the system: as UEs establish sessions, the mapper discovers their slice and tunnel context, the controller keeps probes synchronized with this context, and Grafana updates with per-UE latency labels.

The live scenario uses statically positioned UEs and compares two traffic organizations under the same physical geometry. qhat02 is located farther from the radio unit than qhat01 and qhat03. In the reference phase, all three UEs exchange TCP traffic with an iperf3 server located beyond the UPF, using either one or two parallel streams. This phase establishes the behavior of the deployed configuration. In the far-UE characterization phase, parallel TCP traffic is organized into near-reference, far-UE-only, and joint time windows. This separation exposes whether a delay increase is primarily associated with qhat02's weaker radio condition or also appears on near-reference UEs through shared RAN resources. The dashboard is expected to show the strongest increase in the downlink/RAN-facing view, while MCS, BLER, and HARQ errors explain the radio-side signature.

The main lesson for attendees is that latency samples become useful only when they are interpreted with their 5G context. Fig. 2 shows the live view used in the demonstration. qhat02 is carried on slice 1, while qhat01 and qhat03 are carried on slice 2. The latency panels show the RAN-facing RTT observed at the gNB and UPF anchors, and the p95 panel summarizes downlink tail behavior. The throughput panels confirm the traffic windows, while HARQ errors, BLER, and MCS expose radio-layer behavior during the same interval. Since the tested deployment uses slice labels for attribution rather than strict data-plane QoS isolation, the dashboard can also reveal shared RAN effects: a far or weaker UE may consume radio resources under load and affect near-reference UEs such as qhat03. CPU and memory remain stable, which helps rule out a simple infrastructure-overload explanation.

The dashboard supports a practical diagnosis: when latency increases, the operator can inspect whether the increase appears in the RAN-facing or core-facing view, which UE

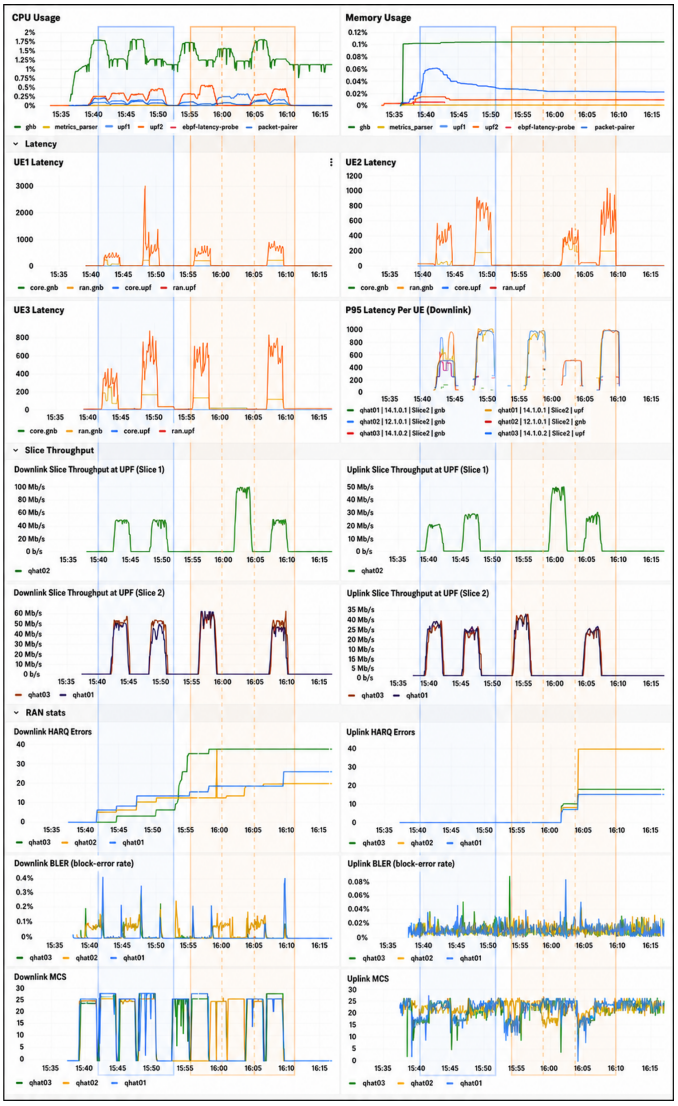


Fig. 2. Live Grafana dashboard for the reference and far-UE RAN-stress scenario. Blue shading marks the all-UE reference traffic window, orange shading marks the far-UE characterization window, and dashed orange lines separate its near-reference, far-UE-only, and joint-traffic subwindows. Per-UE latency, slice throughput, and RAN metrics are shown together to localize RAN-facing latency degradation.

and slice labels are involved, and whether the radio metrics change at the same time. In this run, the strongest visible signature is downlink/RAN-facing latency, accompanied by HARQ/BLER/MCS variations. The per-UE panels also show that the effect can be shared rather than confined to qhat02 alone, which is consistent with multiple UEs sharing the same RAN scheduler and TDD resources.

We will demonstrate the live Grafana view, the UE/slice labels attached to latency samples, the scenario timeline annotations, and the exported artifacts used to recreate the figures after the run. Attendees will see how a raw GTP-U packet observation becomes a labeled latency time series, how the same UE can be inspected across radio and core-facing panels, and how the scenario timeline separates the different phases of an experiment. If the live radio environment is unavailable during the demo session, the same dashboard and artifact pipeline can replay a recorded run collected on the SLICES/R2Lab infrastructure.

IV. Special Requirements

The demo requires a screen, table, power, and regular Internet access. The live testbed itself is accessed remotely. No special radio equipment is required at the conference venue. If conference network conditions prevent live access to the testbed, the demonstration can fall back to a local recorded dashboard replay and exported result artifacts.

V. Conclusion

This demo shows how latency diagnosis can be made practical in a real cloud-native 5G deployment. It extends our previous RAN-observability demonstration by adding user-plane latency measurement and decomposition, and by linking radio metrics with UE-, slice-, and anchor-labeled delay samples. The system derives latency from live 5G traffic and attaches the context needed to interpret it. The reference and far-UE RAN-stress scenario demonstrates this idea live: traffic phases, latency changes, and radio-layer behavior appear together in Grafana, while exported artifacts make the run reproducible after the demo. The result is a reproducible demonstration of how operators and researchers can move from observing a latency increase to identifying where the degradation appears and which network context explains it.

Acknowledgements

This work was supported by the French National Research Agency (ANR) under the France 2030 program through the NF-NAI project (grant ANR-22-PEFT-0003).

References

[1] Open5GS. “Open source project of 5G mobile core network.” https://open5gs.org/.
[2] F. Kaltenberger, R. Knopp, N. Nikaein, D. Nussbaum, and L. Gauthier, “OpenAirInterface: Democratizing wireless innovation,” in *Proc. ACM WiNTECH*, 2020.
[3] R2Lab. “An anechoic wireless testbed for reproducible research.” https://r2lab.inria.fr/.
[4] SLICES-RI. “Scientific LargeScale Infrastructure for Computing/Communication Experimental Studies.” https://www.slices-ri.eu/.
[5] Y. Amami, Z. Mabrouk, C. Barakat, and T. Turletti, “Toward real-time RAN observability in open-source 5G systems,” *ICIN 2026*.
[6] Y. Amami, Z. Mabrouk, C. Barakat, and T. Turletti, “Demonstrating Real-Time RAN Observability in Open-Source 5G: Controlled Interference Scenario,” *ICIN 2026*.
[7] The eBPF Foundation. “eBPF - Introduction, tutorials and community resources.” https://ebpf.io/.
[8] Y. Amami, C. Barakat, and T. Turletti, “On-demand slice-aware latency decomposition in cloud-native 5G systems,” in *Proc. MSWiM*, Paris, France, 2026, to appear.