

SCOUT: User Plane Telemetry for NextG Cellular Networks

Sushila Seshasayee¹, Ushasi Ghosh¹, Qingyuan Zhang¹, Umesh Bellur¹, and Dinesh Bharadia¹

¹ University of California San Diego, CA, USA

{sseshasayee, ughosh, qiz066, ubellur, dineshb}@ucsd.edu

Abstract

5G’s shift to disaggregated O-RAN architectures and high-capacity radio introduces unprecedented transport-layer complexity - precisely as the ecosystem demands predictable, consistent low latency transport and mission-critical use-cases. To understand and ultimately control latency, we need an accurate view of a packet’s journey through this stack. This raises a central question: can network-centric methods suffice, or does an application-centric view demand a broader lens? The user plane is where application performance is realized, yet existing telemetry remains largely control-plane and KPI focused, missing the compound effects of protocols, interfaces, and buffers that packets must traverse. We present Scout, a fine-grained telemetry framework that brings explainability to the 5G user plane. Scout captures packet-level transport dynamics—revealing transient queue buildup, scheduling behavior, and latency composition—to attribute performance variations directly to network decisions. By turning the opaque user plane into an interpretable substrate, Scout can enable data-driven network management, transport debugging, and user-centric control. We call for a shift from KPI monitoring to packet introspection—placing the user plane at the center of end-to-end observability and explainability for next-generation mobile systems.

1 Introduction

The architecture of cellular networks is undergoing a fundamental transformation. The disaggregation of the radio access network (RAN) into centralized, distributed, and radio units (CU/DU/RU)—connected by fronthaul, midhaul, and backhaul links over IP transport—has redefined the 5G and NextG data path [7]. These additional interfaces and protocols introduce new stages of packetization, routing, and prioritization, multiplying the points where delay and jitter can accumulate. At the same time, each components softwareization has replaced custom ASICs with RAN protocol stacks running on commodity hardware, often in virtualized or containerized environments. Platform scheduling, I/O, and compute overheads are now direct determinants of performance. The radio has evolved too, with new bands, wider bandwidths, and more diverse configurations. Collectively, these shifts have produced a network that is far more flexible but also far less predictable. For the first time, the

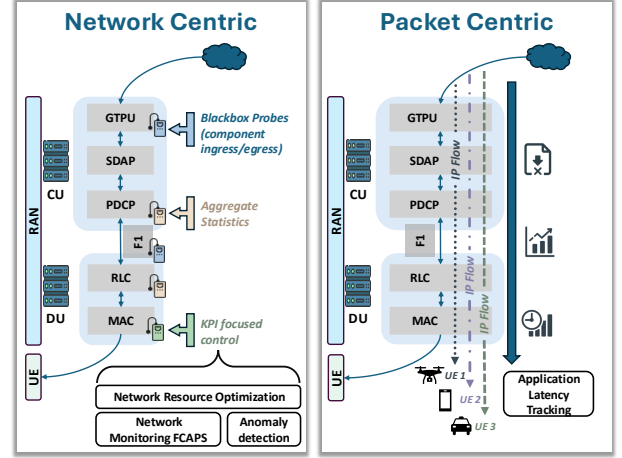


Figure 1: User Plane Telemetry: A Packet Centric View

most difficult part of wireless performance might not lie in the air interface, but in the system built around it.

At the same time, latency has emerged as the defining systems challenge for NextG networks. The tight demands of real-time and mission critical use cases are well recognized—but the problem runs deeper. Virtually every aspect of network operation now depends on precise, predictable delay behavior - not just ultra-reliable low-latency communication (uRLLC), but transport protocol design, congestion management, traffic engineering across heterogeneous service mixes to service placement and offloading. Delay is the control signal that governs bit-rate adaption and TCP congestion control algorithms like BBR [8] and COPA [11], drives scheduling and resource allocation, and increasingly shapes routing and management decisions. Latency has become the network’s most critical control variable.

Despite its central role, latency remains poorly **understood**, especially end-to-end. We study it extensively: at the PHY layer through channel characterization, link adaptation, and HARQ configuration; at Layer 2 through scheduling and resource management; and across the system through an expanding set of KPIs and counters. Each of these efforts is essential, but none present the full picture. They are largely **network-centric**—designed to manage, optimize, and tune the network itself—offering little insight into what actually happens to an application’s IP traffic as it moves through

the RAN. We can observe how the RAN responds to load, heterogeneous mixes, and channel variation, but not how an application's packets are shaped, delayed, or dropped along the way. There is still no framework that can systematically characterize performance—and specifically latency—across layers, components, and timescales, even within the RAN itself. This gap is especially critical today, as the 5G RAN has evolved from a single integrated box into a distributed, software-driven network.

Bridging this gap is not a matter of simply adding more measurement points—it requires a different perspective entirely. To understand how applications and the transport experience the network, we need a *packet-centric, top-down view* that follows a packet's journey as it moves through protocol layers, buffers, compute queues, and transport links, encountering stalls, contention, and cross-traffic along the way (Fig. 1). Only such an approach makes it possible to reveal where and how latency accumulates and how its causes evolve under changing load and radio conditions.

Among all components of the cellular architecture, the user plane is where these effects converge. It is the path every packet takes—the ground truth of performance—where traffic is queued, scheduled, delayed, or dropped, and where software and radio decisions translate directly into what the application experiences. Yet the user plane remains the least understood part of the 5G stack: its timing behavior is rarely captured, and almost never correlated end to end. In the sections that follow, we show why understanding the user plane now matters more than ever, examine the challenges in observing it, and build a framework to study it.

In this work, we present the first framework that systematically exposes and attributes latency within the 5G user plane. Our effort is the first to move beyond component-wise metrics and provide a per-layer, packet-level latency attribution across the full stack. We establish principles for what to measure, how to measure it, and why - guided by the objectives of network **observability** and **explainability**.

At the heart of our approach is deep, in-stack instrumentation that captures the end-to-end trajectory of each packet from ingress to transmission and break down per-packet experienced latency into fundamental sources: **Processing latency** - protocol processing functions, **Queuing** - buffer buildup, contention, and cross-layer interactions, **Scheduling latency** - resource allocation and timings effects.

We implement a proof-of-concept prototype on a standards-compliant 5G testbed, enabling cross-layer visibility that aligns software, protocol, and radio perspectives. This decomposition reveals where delay originates, how it evolves under varying load and channel conditions, and how disaggregation, platform scheduling, and radio dynamics jointly shape user-plane behavior. Together, these capabilities lay

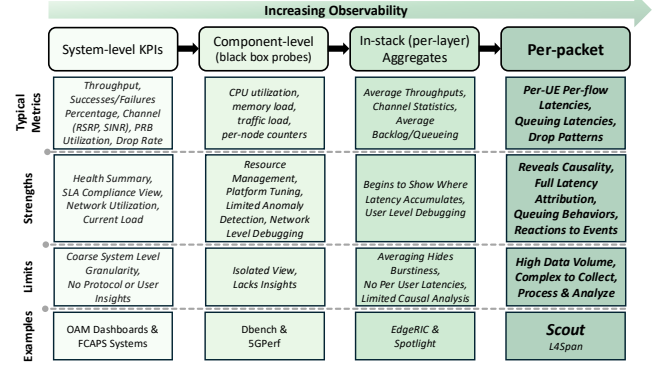


Figure 2: Existing Landscape on Network Observability the foundation for next-generation analytics tools that make latency observable, explainable, and ultimately controllable.

2 Related works

Latency itself has been widely studied, but almost always in silos. Most research focuses on physical- or MAC-layer aspects—such as slot configuration, scheduling mechanisms, or HARQ optimization [1], but ignore the transport stack and hence provide an incomplete picture. Complementary efforts modify RAN internals to mitigate latency but remain narrow in scope, tackling one problem at a time with custom instrumentation. [9] brings down bufferbloat with smaller buffers but adds optional buffering at PDCP to dynamically extend RLC buffers and absorb link variability. [6] analyzes bufferbloat and RLC segmentation as significant latency sources and proposes solutions.

Recent efforts have highlighted the growing importance of user-plane observability in O-RAN systems. [10] measures latency across CU/DU/RU components and shows that 99th-percentile delays can exceed the mean by an order of magnitude—evidence of large latency tails. But it treats each element as a black box without visibility into the variations hidden within aggregates or the ability to derive per-user latency profiles. [3] also instruments the user plane, profiling CPU utilization per layer to study the impact of workload placement and virtualization. This maps compute behavior, but it does not translate directly to packet latency, traffic shaping, or application layer impacts. [2] engineers anomaly detection, collecting hundreds of RAN and platform KPIs (including in the user plane) to identify cross-component faults in multi-vendor Open RAN deployments. Yet it remains aggregate-based and fault-centric, lacking per-user resolution or insight into latency composition. Together, these studies argue that 5G ORAN introduces substantial operational complexity and hence demands fine-grained telemetry but their visibility stops at component-level metrics designed for network optimization rather than latency and application flow visibility.

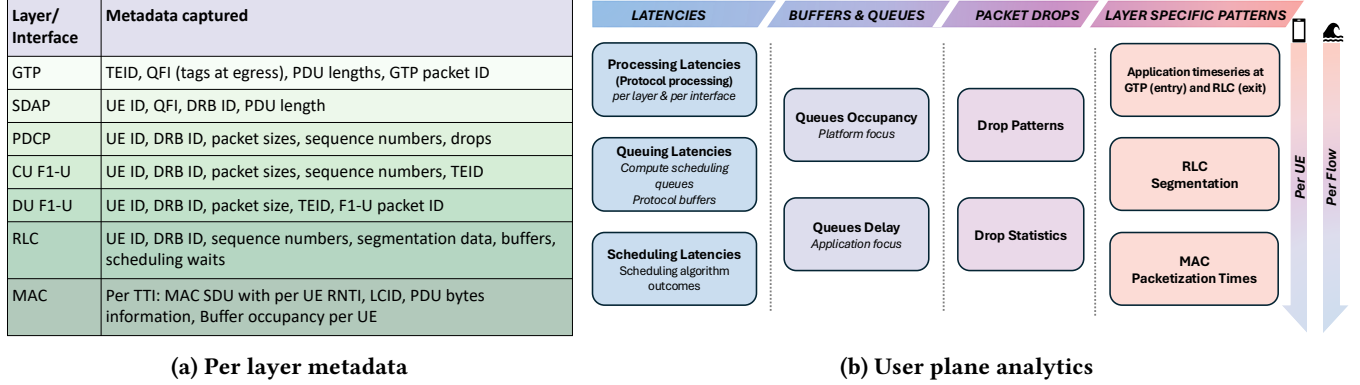


Figure 3: Design overview of the proposed system.

On the other hand, 3GPP measurement entities and some telemetry tools now include the user plane - but these too focus on aggregates, capturing averages but missing tails and reactions to events. Such horizontal per-layer or per-component statistics are valuable for network performance monitoring and platform optimizations, but doesn't provide hard data on application experienced latency. A big part of the limitation comes from a horizontal approach of collecting statistics at fixed points (illustrated in Fig. 1) as traffic flows, a 'watching from the sidelines' approach that only provides layer-level snapshots. What is needed instead is a *packet-centric* view—one that follows each packet as it traverses buffers, I/O paths, and flow-control mechanisms, capturing how its trajectory and timing evolve under varying load and radio conditions. Fig. 2 illustrates the current trends on observability today.

3 A Call for User Plane Telemetry

The 5G RAN is the most dynamic bottleneck in the data path, feeding into the most variable and constrained resource - the wireless channel. Yet its internal behavior remains largely hidden, and as application flows traverse this network, packets are shaped in non-obvious ways by buffering, scheduling, and protocol interactions.

The Challenge. Cellular specifications define multiple buffering points along the user plane — from PDCP and RLC queues to MAC scheduling and retransmission & discard buffers — intended to absorb and recover from channel fluctuations and support handoffs. Software implementations add further queues for task scheduling and I/O, now multiplied across the disaggregated stack. Collectively, these buffers arbitrate how packets are prioritized and delayed, and directly shape both transport- and application-layer performance.

In practice, vendors often overprovision these buffers for worst-case conditions [6] because designing and managing them is inherently difficult. Unlike congestion which evolves gradually, wireless capacity can fluctuate by orders of magnitude on sub-millisecond timescales [12]. Queues can swing

rapidly between buffer-bloated state and underprovisioned, and as [5] observes, buffer sizing and parameter tuning for AQM schemes in these environments is inherently unstable - no one solution will work for all applications, all network configurations. Further, multiple independent buffering and flow-control loops across layers propagate backpressure in complex, implementation-specific ways, making user-plane latency control both *crucial* and fundamentally *difficult*.

While many tools monitor the RLC buffer feeding into the MAC scheduler - the final holding stage before transmission - this alone provides an incomplete view. As our results show, packets can be discarded en masse well above this point, giving the scheduler an incorrect view of the application demand. Capturing true user plane behavior therefore requires instrumenting the full stack—identifying key processing functions at each layer, instrumenting significant holding points (in queues or buffers), and drop/discard locations—to reconstruct the complete packet trajectory through the system. These traces allow deterministic measurements of bursts, queuing latencies (critical data for mitigating buffer-bloat and executing techniques), model scheduler queues and characterize I/O operations.

When aligned across layers, such traces provide true *explainability*—revealing not just *where* latency occurs but *why*. They expose how flow-control actions influence downstream queuing, how RLC backlog growth relates to MAC scheduling decisions or transport-layer retransmissions, and how higher-layer dynamics shape buffering and drops. This cross-layer visibility is essential for understanding RAN behavior at scale and under heterogeneous traffic mixes.

4 SCOUT: Implementation

Scout is intended to look at each packet and understand where it spends time in the stack, turning delay into an explainable outcome. In this section, we describe our framework and proof-of-concept implementation, illustrated in

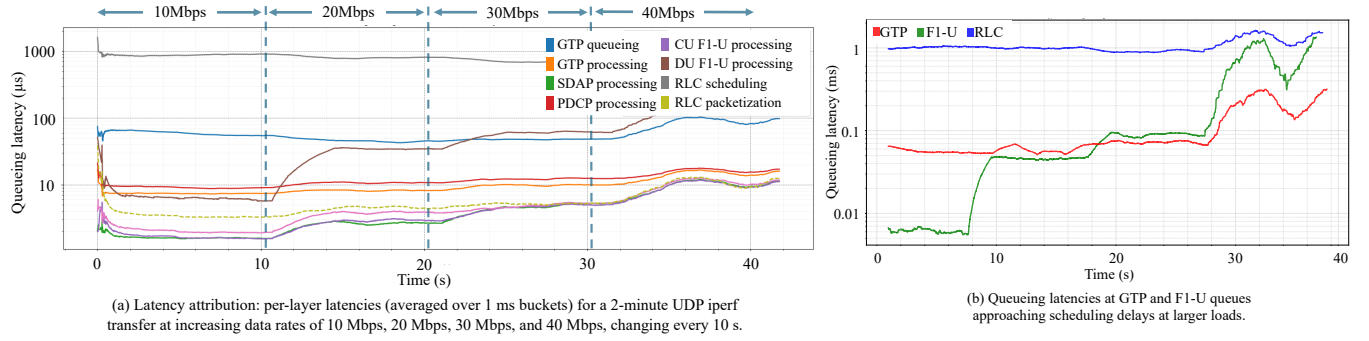


Figure 4: Full stack Latency Attribution

Fig 3. Scout is guided by three core principles: **explainability, observability, and attribution**. Rather than treating latency as a black-box outcome, Scout aims to reveal why and where it occurs.

These principles motivate an end-to-end testbed architecture where real applications can be run atop the instrumented RAN, allowing application-level events to be directly overlaid with internal transport and radio dynamics. **Testbed.** We implement Scout srsRAN (CU/DU/RU) 5G network running on a COTS server with USRP N320 as the RF frontend, Open5GS core network, and COTS UEs: Pixel phones and Amarisoft UE emulator, supporting over-the-air and emulated operation.

Scout is implemented as a four stage pipeline:

In-stack Instrumentation. We identify key checkpoints along the transport stack and instrument them to reconstruct each packet’s journey. Processing delays are measured via entry/exit timestamps on protocol functions (GTP, SDAP, PDCP, RLC, MAC), while enqueue/dequeue markers capture queueing and scheduling time in both protocol and compute queues. All drop points—explicit discards and implicit losses—are logged, and we additionally capture shaping events such as RLC segmentation and MAC packetization. Because the RAN lacks persistent UE IDs, Scout maintains per-UE continuity by reconciling C-RNTI and 5G-S-TMSI across CU/DU via F1-AP/RRC signaling. We leverage srsRANs logging framework to timestamp each packet with the metadata shown in Fig. 3a at each instrumentation point.

Data Collection and Processing. To unify analysis across stacks, Scout defines a common schema for all in-stack events. A multi-threaded parser converts these logs into normalized, time-aligned CSV datasets. A offline **analysis engine** (Fig. 3b) then computes per-UE and per-flow profiles, decomposes latency into processing, queuing, and scheduling components, and visualizes queue occupancy, delays, drops, and scheduling outcomes. Scout **exports results** at multiple granularities: per-packet, per-TTI, or aggregated providing reusable datasets and plots for benchmarking, QoS/QoE analysis, and ML-driven insights.

5 Latency Attribution with SCOUT

We first perform a full latency attribution across the stacks. To our knowledge, this is the first effort to present such a full-stack attribution, quantifying the contribution of each function and comparing them side by side.

The results in Fig. 4a reveal a clear hierarchy: scheduling latencies dominate, processing latencies (SDAP, PDCP, RLC) remain small and stable, while queuing latencies at GTP-U and F1-U fall in between and vary significantly with load.

In the bandwidth-constrained systems studied (<20 MHz, ≈60 Mbps), scheduling latency dominates as expected for capacity-limited systems. This is dictated by capacity and algorithm design—well studied and beyond the immediate scope of this work. With SCOUT, our focus is to provide the scheduler with richer and more accurate information while interpreting higher-layer behaviors in relation to observed scheduling outcomes.

Processing latencies, by contrast, remain small, stable, and largely invariant to load, contributing minimally to end-to-end delay and offering limited scope for optimization.

GTP and F1-U, although interface layers designed primarily for data forwarding, exhibit unexpectedly large delays. Code inspection reveals that these originate from executor queues where packets wait for worker threads i.e. these are queueing latencies. This confirms the impacts of softwarization i.e. compute-level effects such as threading models, and more generally software architecture and platform tuning can significantly influence user-plane latency.

This observation exposes an important and underexplored aspect of 5G systems. While queueing behavior is extensively studied in wired and Wi-Fi networks, it has remained largely opaque in 3GPP systems, where queues are buried deep inside protocol stacks. The disaggregated O-RAN makes this a timely opportunity to examine and optimize queueing within the gNB—now a distributed network rather than a single box. Our results showing that queueing latencies are large in magnitude and highly variable with load, even matching scheduling latencies at higher utilizations (Fig. 4b), identifies a valuable optimization space—particularly at a per-UE or per-application granularity. 3GPP specifications provide an

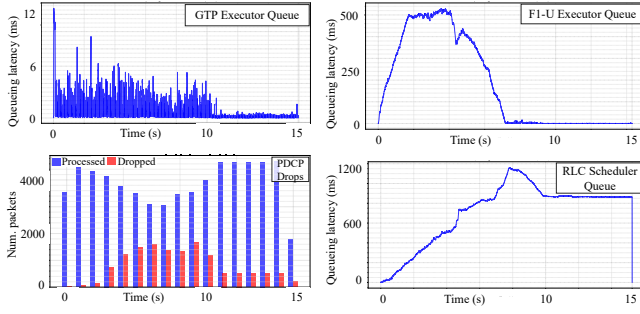


Figure 5: Explainability: Handling excess load, queuing, and drops

inherent advantage here: upstream of the RLC–MAC boundary, flows remain independent across the transport stack. Each UE and DRB maintains its own buffers, timers (e.g., discard, retransmission), and prioritization hooks (e.g., SDAP, F1-U), creating multiple levers before MAC arbitration. This enables (i) per-UE/flow shaping and prioritization to deliver higher-value packets earlier to the scheduler and (ii) buffer management policies (sizing, AQM, flow-control) to tune latency-throughput trade-offs per application.

These results establish a clear baseline for understanding how latency manifests across the user plane, validate Scout’s ability to decompose it end-to-end, and as a benchmarking platform, sets the stage for future exploration.

6 Network Explainability: UDP Flooding

6.0.1 Platform Evaluation: UDP Flooding Test. We begin with a baseline stress test using a fixed 60Mbps Downlink UDP flow exceeding the configured cell capacity. The goal is to validate Scout’s ability to provide cross-layer explainability — showing how internal RAN dynamics shape observable application outcomes.

Queuing Dynamics. At session start, excess traffic is absorbed by available buffers. Latency growth is layer-dependent: negligible at GTP (<15 ms), moderate at F1-U (hundreds of ms), and severe at RLC (exceeding 1s under sustained load) (Fig 5). These trends directly reflect where buffering occurs relative to the bottleneck.

Capacity Saturation and Drops. In the srsRAN implementation, backpressure between PDCP and RLC is a fixed configuration as opposed to a run time signal. PDCP configures a threshold approximating the downstream RLC buffer capacity, and discards new arrivals when the volume of outstanding SDUs exceeds this. With no dedicated buffer and no flow control mechanisms temporary bursts are dropped. This is visible in our traces: F1-U and RLC queuing latencies grow steadily until the RLC buffer limit is reached, at which point PDCP drops rise sharply and continues steadily, after which PDCP drops rise sharply and F1-U latency collapses as traffic is shed upstream (Fig. 5). Such fixed-threshold drop handling

is implementation-specific - precisely the kind of internal mechanism Scout is designed to expose and evaluate.

Steady State Overload. Once RLC saturates, PDCP shedding aligns input with available radio capacity, keeping the buffer full while sustaining maximum throughput. The system stabilizes at this steady state, with consistently high scheduling latency.

Drain Phase. When the source stops, queues drain rapidly across all layers, confirming that Scout captures both buildup and tail behavior. Although the application transmits for 15s, the receiver continues receiving for ≈ 17 s, reflecting the cumulative latency of stacked buffers that can be seen in the queue occupancy plots and statistical summaries.

Summary. This experiment demonstrates Scout’s ability to link application-level symptoms—delay stretch and packet loss—to their internal causes. By temporally aligning cross-layer events, it reveals not just where latency occurs, but *why*, exposing the mechanisms that shape performance.

7 Network Observability: TCP comparisons

We next demonstrate Scout’s ability to observe how transport-layer behavior manifests within the user plane. TCP experiments using *iperf* were conducted with two congestion control schemes CUBIC and BBR, evaluated separately.

The RLC queuing latencies recorded from inside of the RAN stack (Fig. 6 (b) for BBR and (e) for CUBIC) exhibit the same temporal pattern as the per-packet RTTs measured at the application layer (Fig 6 (a) for BBR and (d) for CUBIC). This alignment indicates that the latency dynamics seen by TCP correspond directly with RAN-internal buffering behavior, with RLC as the dominant contributor. GTP and F1-U queues contain correlated but smaller variations, reflecting upstream propagation of congestion from the bottleneck layer.

CUBIC, a loss-based controller, produces the characteristic sawtooth pattern—periodic and pronounced queue buildup and drain cycles—with slightly higher throughput but higher average latency ($p50 = 300.66$ ms) with pronounced tails add numbers and periodic drops add numbers. BBR, being delay-based, regulates its rate more conservatively, achieving comparable throughput (41.5 Mbps) with much lower average per-packet latency ($p50 = 29.11$ ms), no drops, and smoother RLC occupancy.

These deterministic measurements of queueing latencies are visible only through Scout’s timestamped per-packet, in-stack instrumentation, going beyond the limited scope of inferred metrics from per-layer aggregates of other systems. Such ground truth is essential for accurately capturing the temporal dynamics of transport behavior from inside the RAN. These baseline experiments validate Scout’s ability to observe TCP end-to-end and demonstrate its potential: congestion-control mechanisms, under diverse load and

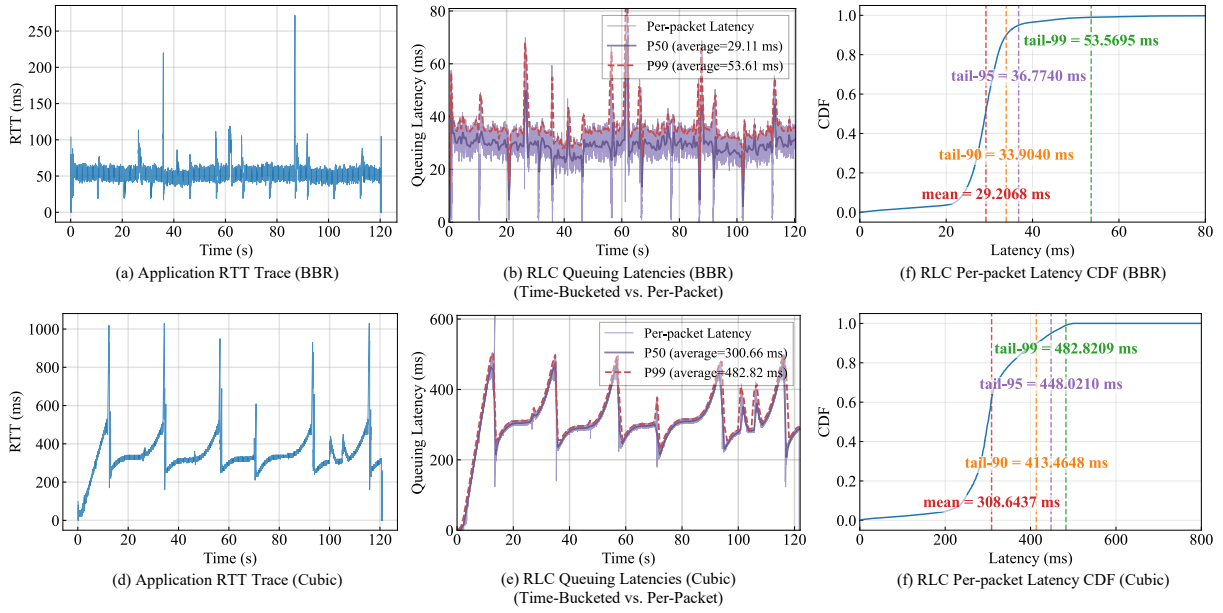


Figure 6: Network Observability

channel conditions, can now be systematically analyzed with true packet-level visibility into where and how their effects arise.

8 Discussion and Future Directions

Implementing fine-grained user-plane instrumentation is inherently challenging. Capturing per-packet events with microsecond precision risks perturbing the system being measured, and the resulting substantial data volumes make real-time analysis difficult. The current offline pipeline uses multi-threaded processing and vectorized NumPy operations to process million-line traces within tens of minutes on commodity servers, showing that Scout's fine-grained instrumentation is already practically scalable. This landscape is rapidly evolving: programmable observability frameworks such as [13], enable dynamic, in-stack instrumentation for Open RAN, demonstrate how low-overhead telemetry can be integrated directly into the RAN software plane. Alongside advances in high-performance analytics and streaming computation, real-time, scalable analysis of fine-grained telemetry is now within reach. With such capabilities, Scout can enable the next generation of RAN research and optimization. Prior efforts such as [4] and [9] have shown the value of in-stack visibility but remain limited to narrow use cases and custom instrumentation. Scout generalizes these approaches—providing a unified, cross-layer, per-packet framework that spans the entire user plane. This foundation supports reproducible benchmarking, comparative analysis across deployments, and the design of new congestion-control, buffering, and scheduling strategies grounded in real in-stack measurements.

References

- [1] A. Gong et al. 2025. Towards URLLC with Open-Source 5G Software (*OpenRIT6G '25, Sigcomm*). doi:10.1145/3750718.3750743
- [2] C. Sun et al. 2024. SpotLight: Accurate, Explainable and Efficient Anomaly Detection for Open RAN (*ACM MobiCom '24*).
- [3] C. Wei et al. 2022. 5GPerf: profiling open source 5G RAN components under different architectural deployments (*5G-MeMU '22, Sigcomm*).
- [4] H. Wan et al. 2025. L4Span: Spanning Congestion Signaling over NextG Networks for Interactive Applications. *arXiv preprint arXiv:2510.02682* (2025).
- [5] K. Winstein et al. 2013. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. 459–471.
- [6] M. Irazabal et al. 2021. Preventing RLC buffer sojourn delays in 5G. *IEEE access* 9 (2021), 39466–39488.
- [7] M. Polese et al. 2023. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411. doi:10.1109/COMST.2023.3239220
- [8] N. Cardwell et al. 2017. BBR: congestion-based congestion control. *Commun. ACM* 60, 2 (Jan. 2017), 58–66. doi:10.1145/3009824
- [9] R. Kumar et al. 2018. Dynamic control of RLC buffer size for latency minimization in mobile RAN. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–6. doi:10.1109/WCNC.2018.8377190
- [10] S. K. Permal et al. 2024. Towards Continuous Latency Monitoring through Open RAN Interfaces (*5G-MeMU '24, Sigcomm*). New York, NY, USA, 14–20.
- [11] V. Arun et al. 2018. Copa: practical delay-based congestion control for the internet (*NSDI'18*). USENIX Association, USA.
- [12] W.-H. Ko et al. 2024. EdgeRIC: empowering real-time intelligent optimization and control in nextg cellular networks (*NSDI'24*). USENIX Association, USA.
- [13] X. Foukas et al. 2023. Taking 5G RAN analytics and control to a new level (*ACM MobiCom '23*).