

I. Shayea

Istanbul Technical University, Istanbul, Turkey

e-mail: ibr.shayea@gmail.com

BUFFERBLOAT, NOT THE RADIO: LATENCY IN A MULTI-USER 5G STANDALONE NETWORK

Abstract. End-to-end latency, not raw throughput, increasingly limits the quality of experience in modern mobile networks. This paper presents an experimental characterization of the latency bottleneck in a fully software-defined 5G Standalone (SA) network built from srsRAN, Open5GS, and a ZeroMQ-based radio emulator. Across twelve deterministic scenarios (S1-S12) that independently vary the number of active users, the offered load, the radio quality, and the resource-management policy, we measure downlink throughput, round-trip time (RTT), and packet loss with second-level aggregation over repeated 60-second runs. Three findings stand out. First, scaling the cell from one user to five or seven triggers severe bufferbloat: mean RTT rises from roughly 0.59 s to 5–7 s, with a 95th-percentile reaching 13.4 s and an anomalous run peaking near 29 s, while packet loss remains at zero. Second, degrading the radio channel from good to poor conditions does not reduce throughput or raise latency, because MAC-layer adaptation and HARQ/ARQ compensate for the radio impairment under deterministic load; the dominant latency factor is buffer saturation, not the air interface. Third, a static resource-management policy worsens latency by about 12% relative to no control at moderate load. We conclude that mitigating queue buildup, rather than improving the radio link, is the primary lever for latency control in this regime, and we discuss the implications for active queue management and learning-based RAN control.

Keywords: 5G Standalone, bufferbloat, latency, round-trip time, TCP, srsRAN, Open5GS

1. Introduction

For most of the history of cellular networks, performance has been quantified primarily in terms of throughput. Yet for the interactive and real-time services that dominate contemporary traffic, such as video conferencing, cloud gaming, and industrial control, the binding constraint is latency, and particularly the tail of the round-trip-time distribution. A network that delivers high average throughput but multi-second delay under load is unusable for these applications. Understanding what produces such delay in a 5G system is therefore of direct practical importance.

A well-known but frequently underappreciated source of latency is bufferbloat: the accumulation of packets in oversized network buffers that keeps queues persistently full, inflating delay without improving goodput [1]. Bufferbloat was first characterized in the wired Internet [1], [7] and later shown to be especially severe in cellular networks, where base stations maintain large buffers to smooth the bursty, time-varying radio channel [2], [3]. Measurement campaigns over commercial 3G and 4G networks reported end-to-end delays reaching several seconds under load [3], and subsequent work

confirmed that the problem persists and can worsen in 5G, including mmWave deployments where rapid rate fluctuations interact badly with loss-based congestion control [5], [6]. A large literature on active queue management (AQM) RED [19], CoDel [7], PIE [9], and FQ-CoDel [8] exists precisely to counter this effect, as do delay- and rate-based congestion-control algorithms such as BBR [21].

Despite this body of work, most cellular bufferbloat studies treat the network as a black box measured from the device, without visibility into the protocol stack that produces the queueing. The emergence of open-source, software-defined 5G stacks srsRAN [14], Open5GS [15], and O-RAN-compliant controllers such as FlexRIC [13] changes this. It is now possible to build a fully instrumented 5G SA network on commodity hardware [16] – [18] and to observe, under controlled and repeatable conditions, exactly how load, user count, radio quality, and control policy combine to produce latency. This capability motivates a measurement study that isolates the latency bottleneck rather than merely reporting it.

Two further considerations make a fresh, instrumented study worthwhile. First, much of the foundational cellular bufferbloat literature predates



5G NR and was collected on 3G and 4G networks whose buffering and scheduling differ materially from the NR MAC [2] – [4]; whether the same dominance of queueing over the radio holds in a 5G SA stack is not obvious a priori. Second, the interaction between concurrent TCP flows and a shared radio scheduler is inherently a multi-user phenomenon, yet single-device measurements cannot expose how contention among several users drives buffers into saturation. A controlled multi-user environment in which the number of users is itself a swept variable is therefore essential to locating the bottleneck. The software-defined testbed used here provides exactly this control while retaining full visibility into throughput, delay, loss, and resource utilization at one-second resolution.

Unlike earlier cellular bufferbloat studies based primarily on end-device measurements in commercial 3G/4G networks, the present work uses a fully instrumented 5G SA testbed in which user count, offered load, radio quality, and control policy can be varied independently. This design separates the effects of multi-user contention and radio impairment under repeatable conditions while providing direct access to RAN telemetry and control through the Near-RT RIC.

This paper asks a deliberately sharp question: in a multi-user 5G SA cell, is the latency experienced under load governed by the radio channel or by buffer dynamics?

To answer it, we run twelve deterministic scenarios on a software-defined testbed, varying one factor at a time: user count, offered load, radio quality, or resource-management mode, and measuring throughput, RTT, and loss with repeated runs for reproducibility.

Our contributions are:

- (i) a controlled experimental decomposition of the 5G SA latency bottleneck that attributes the dominant effect to bufferbloat rather than the air interface;
- (ii) quantitative evidence that radio degradation from good to poor is compensated by MAC adaptation and HARQ/ARQ under deterministic load, leaving latency essentially unchanged;
- (iii) the observation that a static resource-management policy can be worse than no control at

moderate load, motivating adaptive, queue-aware control.

We frame the results in terms of their implications for AQM and learning-based RAN control.

2. Materials and methods

2.1. Testbed architecture

The measurements were performed on a fully software-defined 5G SA network assembled from open-source components and requiring no dedicated radio hardware. The radio access network is provided by the srsRAN Project gNodeB [14], the core network by Open5GS [15], and the air interface is emulated by exchanging baseband IQ samples over ZeroMQ sockets rather than over physical radios, which guarantees bit-exact repeatability and isolation from external interference [16]. A GNU Radio flow graph sits between the gNodeB and the user-equipment (UE) instances, applying controllable attenuation and additive white Gaussian noise and exposing an XML-RPC interface so that channel quality can be varied at runtime. Each emulated UE runs in its own Linux network namespace, and an O-RAN-compliant Near-RT RIC (FlexRIC [13]) attaches to the gNodeB over the E2 interface to collect KPM telemetry and optionally apply RAN control. The overall structure is shown in Figure 1. A complete description of the platform and its configuration is given in a companion paper; here we summarize only the elements needed to interpret the measurements.

The cell operates in 5G NR Standalone mode on band n3 with a 10 MHz channel (52 physical resource blocks). To minimize scheduler jitter, the latency-critical physical-layer and ZeroMQ-broker threads are pinned to CPU cores isolated from the operating-system scheduler, and network-interface interrupts are confined to non-isolated cores. System and hardware telemetry CPU utilization, memory, interrupt counts, and processor power via the Intel RAPL interface are sampled once per second by a lightweight custom daemon that reads kernel pseudo-files directly to avoid the overhead of standard monitoring utilities.

5G SA Software-Defined Testbed

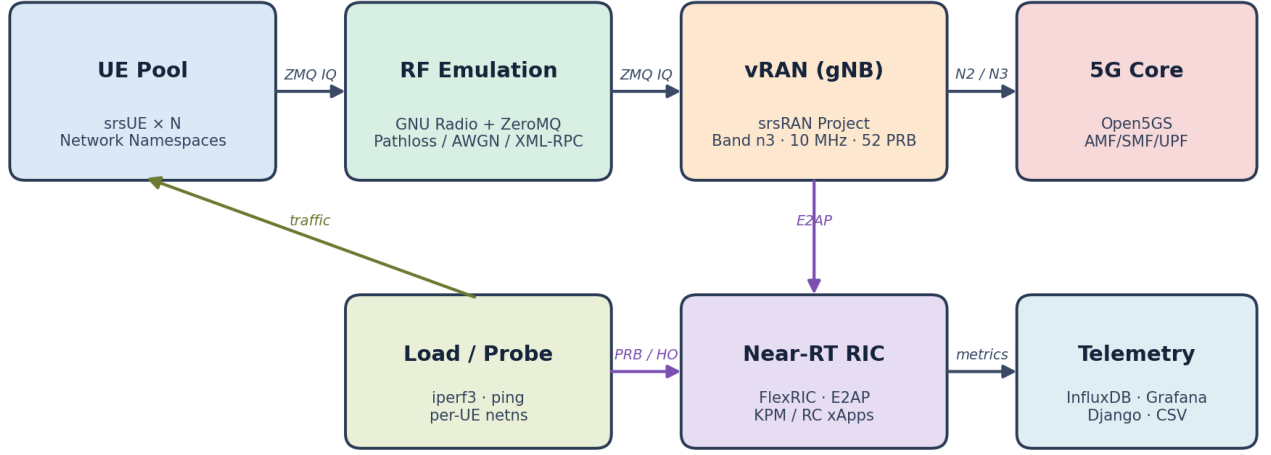


Figure 1. Architecture of the software-defined 5G SA testbed. The UE pool communicates with the srsRAN gNodeB through the ZeroMQ/GNU Radio RF-emulation layer, while Open5GS provides the 5G core. The Near-RT RIC collects KPM telemetry through the E2 interface and applies resource-management control in the controlled scenarios.

2.2. Experimental scenarios and KPI definitions

Twelve deterministic scenarios, summarized in Table 1, were defined to isolate the effect of one factor at a time. Scenarios S1-S3 vary the offered load for a single user; S4-S6 scale the number of simultaneously active users; S7-S8 change the traffic direction and symmetry; S9-S10 degrade the radio quality; and S11-S12 enables a static resource-management

policy. In every scenario, all factors other than the one under study are held fixed. Five and seven UEs were selected to represent the transition from moderate multi-user contention to a more heavily loaded regime approaching saturation in the available 10 MHz cell. These configurations were therefore used to examine how latency changes as shared radio resources become increasingly contended.

Table 1. Deterministic experimental scenarios (S1-S12).

ID	UEs	Load	Radio	Control	Purpose
S1	1	Low (5 Mbps)	Good	Off	Single-UE baseline, minimal load
S2	1	Medium (20 Mbps)	Good	Off	Single-UE, moderate load
S3	1	High (50+ Mbps)	Good	Off	Single-UE, maximal load
S4	5	Medium	Good	Off	Multi-UE contention for PRBs
S5	7	Medium	Good	Off	Buffer limit at higher user count
S6	7	High	Good	Off	Channel saturation under high load
S7	5	Medium (UL)	Good	Off	Uplink-only traffic profile
S8	5	Medium (symm.)	Good	Off	Symmetric DL+UL traffic
S9	5	Medium (DL)	Moderate	Off	Moderate radio degradation
S10	5	Medium (DL)	Poor	Off	Severe radio degradation
S11	5	Medium (DL)	Good	Static	Static RRM policy
S12	7	Medium (DL)	Good	Static	Static RRM at higher user count

The performance indicators are defined as follows. Downlink throughput is the byte count delivered per one-second interval. Round-trip time is measured by a parallel ping stream (300 packets at 0.2 s spacing) to the core user-plane gateway, from which the mean and the 95th percentile are computed:

$$RTT_{p95} = Q_{RTT}(0.95) \quad (1)$$

where Q_{RTT} is the empirical quantile function of the per-interval RTT samples. Packet loss is the fraction of ping packets sent but not returned. Physical-resource-block (PRB) utilization is the ratio of scheduled to available PRBs out of the 52 available in the 10 MHz channel, and the block error rate (BLER) is the ratio of erroneous to total transport blocks reported by the gNodeB physical layer.

2.3. Measurement methodology

Each scenario was executed for at least three independent 60-second runs, and key performance indicators were aggregated at one-second resolution. Downlink traffic was generated with iperf3 over TCP, while a concurrent ping stream measured latency to the core gateway.

During the deterministic S1-S12 runs, iperf3 generated TCP traffic with the offered rate set by the experimental load level: 5 Mbps for Low, 20 Mbps for Medium, and 50+ Mbps for High. The traffic direction was configured as downlink, uplink, or symmetric DL+UL according to the scenario definition, while each run lasted 60 s and the KPI aggregation interval was 1 s. A concurrent ping stream sent 300 packets at 0.2 s intervals to the UPF gateway (10.45.0.1) for RTT and loss measurement. To characterize the energy cost of the radio access stack, we

also report a normalized efficiency metric defined as the radio throughput delivered per watt of processor power drawn by the gNodeB:

$$E_{eff} = \frac{R_{radio}}{P_{RAPL}} \quad (2)$$

where R_{radio} is the gNodeB radio throughput in Mbit/s and P_{RAPL} is the instantaneous processor power in watts read from the Intel RAPL counters. To relate buffer dynamics to delay, it is useful to recall the bandwidth-delay product, which sets the congestion-window size beyond which a loss-based sender merely fills queues rather than increasing goodput:

$$BDP = R_{link} \cdot RTT_{min} \quad (3)$$

with R_{link} the link rate and RTT_{min} the unloaded round-trip time. When a sender's in-flight data exceeds the BDP, the excess accumulates in buffers and manifests directly as added delay, the mechanism we observe below. Because the standard deviation of the delay matters as much as its mean for interactive traffic, we additionally report the mean deviation of the per-packet RTT within a run.

3. Results and discussion

3.1. Scaling collapse: the onset of bufferbloat

Table 2 reports the aggregate KPIs for all twelve scenarios. With a single active user (S1-S3), increasing the offered load from 5 to over 50 Mbps does not degrade the channel: throughput remains in a narrow 23.3-25.0 Mbit/s band the practical ceiling of the emulated radio driver at 10 MHz and mean RTT stays near 0.58-0.61 s with zero loss. A single user cannot overrun the scheduler or fill the core buffers.

Table 2. Aggregate KPI matrix for scenarios S1-S12. Highlighted rows exhibit multi-second bufferbloat.

ID	UEs	Load / Radio	Avg DL (Mbps)	Avg RTT (ms)	P95 RTT (ms)	Loss	Status
S1	1	Low / Good	25.0	591	856	0%	OK
S2	1	Med / Good	24.5	583	881	0%	OK
S3	1	High / Good	23.3	612	931	0%	OK
S4	5	Med / Good	18.5	7251	13393	0%	bloat
S5	7	Med / Good	29.0	5042	7852	0%	bloat
S6	7	High / Good	28.8	5099	7881	0%	bloat
S7	5	UL / Good	0.0	376	456	0%	OK
S8	5	Symm / Good	26.8	2820	5522	0.2 pk	bloat
S9	5	Med / Moder.	27.5	2958	5514	0%	bloat

Table continued

ID	UEs	Load / Radio	Avg DL (Mbps)	Avg RTT (ms)	P95 RTT (ms)	Loss	Status
S10	5	Med / Poor	27.3	2960	5504	0.1 pk	bloat
S11	5	Med / Good (Static)	27.1	3173	5641	0%	bloat
S12	7	Med / Good (Static)	29.2	5040	7845	0%	bloat

The picture changes abruptly when the cell is scaled to five or seven users (S4-S6). Mean RTT jumps by an order of magnitude, to 5-7 s, and the 95th percentile reaches 13.4 s in S4, far above the latency levels expected for interactive 5G services. Critically, packet loss remains zero throughout: the delay is produced by packets waiting in full buffers, not by packets being dropped. This is the signature of bufferbloat. Aggregate throughput rises with user count (to 29.0 Mbit/s) because the scheduler aggregates more independent queues, supporting the interpretation that, in this operating regime, queue saturation rather than radio capacity is the dominant latency constraint.

The per-run breakdown in Table 3 sharpens this. The seven-user scenario S5 is highly reproducible, with inter-run RTT spread below 4%. The five-user scenario S4 is bimodal: two runs behave normally (~2.8-3.4 s) while one run collapses to 1.2 Mbit/s with a 15.5 s mean RTT and a 29 s tail. This behavior is consistent with TCP entering repeated retransmission timeouts and slow-start near buffer saturation. However, given the limited number of repeated runs, the experiment does not establish the exact mechanism of this anomalous event. The observation nevertheless indicates that operation near the buffer limit can produce substantial run-to-run latency variability.

Table 3. Per-run reproducibility for the critical scaling scenarios. The anomalous S4 run is retained to illustrate the observed instability near the saturation regime

Scenario	Run	Avg DL (Mbps)	Avg RTT (ms)	P95 RTT (ms)	Loss
S4 (5 UE, Med, Good)	Run 1	27.0	2816	5543	0
	Run 2 (anomaly)	1.2	15524	28958	0
	Run 3	27.2	3413	5678	0
S5 (7 UE, Med, Good)	Run 1	29.0	4924	7831	0
	Run 2	29.0	5104	7872	0
	Run 3	29.1	5099	7854	0

Table 4 reports the per-user latency statistics for this run. The means cluster tightly around 3.0 s with one user (UE 5) somewhat lower at 2.40 s, and the mean deviations are large (1.5-1.9 s), quantifying

the heavy oscillation of phase three. The broadly comparable delay across users suggests that the observed latency is a system-level property of the saturated regime rather than an isolated per-user artifact.

Table 4. Per-user RTT statistics for scenario S9, run 1 (five users, moderate radio)

User	Min RTT (ms)	Avg RTT (ms)	Max RTT (ms)	MDev (ms)	Loss
UE 1	20.0	3073	5744	1846	0/300
UE 2	18.9	3009	5753	1843	0/300
UE 3	19.6	3062	5636	1783	0/300
UE 4	20.2	3173	5799	1887	0/300
UE 5	20.8	2396	5437	1469	0/300

3.2. *The radio channel is not the dominant bottleneck in the investigated regime*

Scenarios S8-S10 deliberately degrade the radio channel from good to moderate to poor by reducing signal power in the emulator. Strikingly, this has almost no effect on the measured KPIs: downlink throughput holds at 27.3-27.5 Mbit/s and mean RTT stays near 2.96 s across all three radio conditions. Under deterministic, fixed load, the gNodeB's link-adaptation machinery—modulation-and-coding-scheme selection together with HARQ and RLC ARQ retransmissions—compensates for the worse channel. In this regime, the additional retransmission cost does not substantially reduce delivered throughput because sufficient scheduler capacity remains available to absorb the impairment. Consequently, radio degradation does not become the dominant contributor to end-to-end RTT. Under the deterministic traffic load, 10 MHz configuration, and radio impairments investigated here, the results therefore indicate that latency is governed primarily by queue dynamics rather than radio quality.

A finer-grained look at a single representative run (S9, five users, moderate radio) explains the mechanism behind the multi-second tails. The per-packet latency over a run unfolds in three phases. Initially, while TCP completes its handshake and the buffers are still empty, ping packets are served almost immediately and the RTT sits between roughly 19 and 50 ms. As the senders enter congestion-window expansion and the offered rate exceeds the cell capacity, the RTT climbs essentially linearly from about 0.13 s to nearly 5.8 s as buffers fill at a rate equal to the gap between input load and radio delivery. Finally, with buffers saturated, the RTT oscillates between roughly 4.0 and 5.8 s with a mean deviation near 1.8 s, the oscillation locked to periodic packet drops and the consequent congestion-window reductions. Throughput over the same run is shared almost symmetrically across the five users (each averaging about 5.4 Mbit/s) but is highly pulsed, with bursts up to nearly 29 Mbit/s interleaved with second-long drops to zero as individual flows stall and recover. This behavior is consistent with the bandwidth-delay product interpretation in Equation (3) once in-flight data persistently exceeds the BDP.

The uplink-only scenario S7 reinforces the queueing interpretation from the other direction. With no downlink traffic, the mean RTT is 0.376 s, far below the loaded values but still well above the unloaded baseline because uplink scheduling on PUSCH introduces its own modest queueing delay.

Latency in this system is consistently a queue phenomenon, present whenever resources are contented and explosive when buffers saturate.

3.3. *Static control underperforms; implications for adaptive control*

Scenarios S11-S12 test a static resource-management policy. At moderate load with five users (S11), enabling static control raises mean RTT from 2.82 s to 3.17 s, a 12% latency penalty relative to no control at all. A policy that reserves time-frequency resources by a fixed rule, without reacting to the instantaneous occupancy of the TCP buffers, simply adds waiting time. When the cell is pushed to seven users and is already saturated (S12), the static policy neither helps nor hurts: its RTT (5.04 s) is essentially unchanged from the uncontrolled seven-user case (5.04 s in S5). Static control is thus at best neutral and at worst harmful in this regime.

The static RRM mode applies a fixed, hard resource-reservation rule rather than adapting allocation to the instantaneous state of the TCP queues. In contrast to the default scheduler, which can react to changing demand, the static policy does not track buffer occupancy or redistribute resources in response to short-term traffic dynamics.

Taken together, the three findings point to a clear engineering conclusion. Because the dominant latency factor in the investigated regime is buffer saturation rather than radio quality, and because static control cannot track buffer dynamics, the effective levers for latency control are those that act on the queues themselves: active queue management such as CoDel, PIE, or FQ-CoDel deployed at the RLC/PDCP layers [7], [8], [9], delay- or rate-based congestion control such as BBR at the transport layer [21], and, most relevant to the O-RAN setting, closed-loop RAN control that reacts to buffer-occupancy telemetry in near real time. Such a controller could combine buffer occupancy with RTT trends, PRB utilization, throughput, and BLER as state information for adaptive resource allocation. The fact that radio degradation is fully masked under the deterministic conditions tested also cautions against over-investing in link-level optimization when the operating regime is queue-bound. The same instrumented testbed provides a natural vehicle for evaluating such queue-aware controllers under identical, repeatable conditions, which we identify as the immediate next step.

This study has several limitations. The radio interface was software-emulated rather than evaluated over an over-the-air channel, the cell used a fixed

10 MHz band-n3 configuration, and the experiments focused primarily on deterministic TCP traffic without mobility or external interference. In addition, each scenario was repeated a limited number of times. Accordingly, the results characterize the investigated operating regime rather than all possible 5G deployment conditions.

4. Conclusions

We presented a controlled experimental characterization of the latency bottleneck in a multi-user 5G Standalone network realized entirely in software with srsRAN, Open5GS, and a ZeroMQ radio emulator. By varying user count, offered load, radio quality, and control policy one factor at a time across twelve deterministic scenarios, we showed that, within the investigated operating regime, latency under load is dominated by bufferbloat rather than by radio degradation. Scaling from one user to five or seven inflates mean round-trip time from about 0.59 s to 5-7 s, with tails exceeding 13 s and an anomalous run near 29 s, all at zero packet loss; degrading the channel from good to poor leaves

throughput and latency essentially unchanged because MAC adaptation and HARQ/ARQ absorb the impairment; and a static resource-management policy adds roughly 12% latency at moderate load while providing no benefit at saturation. The practical implication is that latency control in this regime should target queue buildup through active queue management, delay-based congestion control, and closed-loop RAN control reacting to buffer telemetry rather than focusing exclusively on the air interface. These conclusions apply to the deterministic, software-emulated operating regime investigated here; validation under over-the-air, mobile, and more variable traffic conditions remains an important direction for future work.

Funding

This research was funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. BR24993211).

Conflicts of Interest

The authors declare no conflict of interest.

References

1. J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," *Commun. ACM*, vol. 55, no. 1, pp. 57–65, 2012.
2. H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling bufferbloat in 3G/4G networks," in *Proc. ACM Internet Meas. Conf. (IMC)*, 2012, pp. 329–342, doi: 10.1145/2398776.2398810.
3. H. Jiang, Z. Liu, Y. Wang, K. Lee, and I. Rhee, "Understanding bufferbloat in cellular networks," in *Proc. ACM SIGCOMM Workshop Cellular Netw. (CellNet)*, 2012, pp. 1–6.
4. C. A. Grazia, N. Patriciello, M. Klapez, and M. Casoni, "Impact of TCP congestion control on bufferbloat in cellular networks," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, 2013.
5. M. Polese, M. Mezzavilla, and M. Zorzi, "Will TCP work in mmWave 5G cellular networks?," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 65–71, 2019.
6. M. Mezzavilla et al., "milliProxy: A TCP proxy architecture for 5G mmWave cellular systems," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, 2017.
7. K. Nichols and V. Jacobson, "Controlling queue delay," *Commun. ACM*, vol. 55, no. 7, pp. 42–50, 2012.
8. T. Høiland-Jørgensen, P. McKenney, D. Täht, J. Gettys, and E. Dumazet, "The flow queue CoDel packet scheduler and active queue management algorithm," IETF RFC 8290, 2018.
9. R. Pan et al., "PIE: A lightweight control scheme to address the bufferbloat problem," in *Proc. IEEE High Perform. Switching Routing (HPSR)*, 2013.
10. 3GPP, "NR; Overall description; Stage-2," 3GPP TS 38.300, V18.0.0, 2024.
11. 3GPP, "NR; Medium Access Control (MAC) protocol specification," 3GPP TS 38.321, V18.0.0, 2024.
12. O-RAN Alliance, "O-RAN architecture description," O-RAN.WG1.OAD, v11.0, 2024.
13. R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: An SDK for next-generation SD-RANs," in *Proc. ACM CoNEXT*, 2021, pp. 411–425.
14. srsRAN Project, "srsRAN Project Documentation," Software Radio Systems, 2024. [Online]. Available: <https://docs.srsran.com>
15. Open5GS, "Open5GS Documentation: Open-source 5G core network," 2024. [Online]. Available: <https://open5gs.org>
16. L. Mamushiane, A. Lysko, H. Kobo, and J. Mwangama, "Deploying a stable 5G SA testbed using srsRAN and Open5GS: UE integration and troubleshooting towards network slicing," in *Proc. Int. Conf. Adv. Big Data, Comput. Data Commun. Syst. (icABCD)*, 2023, doi: 10.1109/icABCD59051.2023.10220512.
17. A. E. Ferguson, U. Pawar, T. Wang, and M. K. Marina, "Campus5G: A campus scale private 5G Open RAN testbed," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 55, no. 3, pp. 19–28, 2025.

18. NIST, “Blueprint for deploying 5G O-RAN testbeds,” NIST Tech. Note 2311, 2024.
19. S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.
20. S. Ha, I. Rhee, and L. Xu, “CUBIC: A new TCP-friendly high-speed TCP variant,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.
21. N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, “BBR: Congestion-based congestion control,” *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.
22. A. Aggarwal, S. Savage, and T. Anderson, “Understanding the performance of TCP pacing,” in *Proc. IEEE INFOCOM*, 2000.
23. A. Saeed et al., «Annulus: A dual congestion control loop for datacenter and WAN traffic aggregates,» in *Proc. ACM SIGCOMM*, 2020.
24. A. Mirzaeinia, M. Mirzaeinia, and A. Rezgüi, «Latency and throughput optimization in modern networks: A comprehensive survey,» 2020, arXiv:2009.03715.

Information about the author:

Ibraheem Shayea – Senior Scientific Researcher, Electronics and Communication Engineering Department, Faculty of Electrical and Electronics Engineering, Istanbul Technical University, ibr.shayea@gmail.com, ORCID ID: 0000-0003-0957-4468.

Submission received: 16 July, 2026

Revised: 20 August, 2026

Accepted: 20 August, 2026