L4Span: Spanning Congestion Signaling over NextG Networks for Interactive Applications

HAORAN WAN, Princeton University, USA
KYLE JAMIESON, Princeton University, USA

Design for low latency networking is essential for tomorrow's interactive applications, but it is essential to deploy incrementally and universally at the network's last mile. While wired broadband ISPs are rolling out the leading queue occupancy signaling mechanisms, the cellular Radio Access Network (RAN), another important last mile to many users, lags behind these efforts. This paper proposes a new RAN design, **L4Span**, that abstracts the complexities of RAN queueing in a simple interface, thus tying the queue state of the RAN to end-to-end low-latency signaling all the way back to the content server. At millisecond-level timescales, L4Span predicts the RAN's queuing occupancy and performs ECN marking for both low-latency and classic flows. L4Span is lightweight, requiring minimal RAN modifications, and remains 3GPP and O-RAN compliant for maximum ease of deployment. We implement a prototype on the srsRAN open-source software in C++. Our evaluation compares the performance of low-latency as well as classic flows with or without the deployment of L4Span in various wireless channel conditions. Results show that L4Span reduces the one-way delay of both low-latency and classic flows by up to 98%, while simultaneously maintaining near line-rate throughput. The code is available at **https://github.com/PrincetonUniversity/L4Span**.

CCS Concepts: • **Networks** → **Mobile networks**; **Signaling protocols**; **In-network processing**.

Additional Key Words and Phrases: 5G Network; Congestion Control; ECN Feedback; L4S Architecture.

ACM Reference Format:
Haoran Wan and Kyle Jamieson. 2025. L4Span: Spanning Congestion Signaling over NextG Networks for Interactive Applications. *Proc. ACM Netw.* 3, CoNEXT4, Article 25 (December 2025), 25 pages. https://doi.org/10.1145/3768972

1 Introduction

Today and tomorrow's interactive applications (*e.g.*, videoconferencing, AR/VR, cloud gaming) all depend on controlling content senders' rates to strike a balance between utilizing the network and avoiding latency-inducing queues. Indeed, recent congestion control algorithms have been evolving to design for this balance [8, 22, 32, 60]. To realize this higher performance bar, congestion control algorithms have explored feedback information from the receiver or network that is more timely and richer than simply packet loss (*e.g.*, ABC [36], XCP [45]). The key to realizing these protocols on the Internet, however, is end-to-end deployability, and for this, one front-running approach is the use of Explicit Congestion Notification (ECN) bits coupled with *Low Latency Low Loss Scalable* (L4S) congestion signaling [13, 16, 69]. L4S is gaining traction: it is currently rolling out in broadband cable service provider networks as *Low Latency DOCSIS* [26, 58], and the mobile standards body, 3GPP, has stated its intention to standardize L4S in the cellular Radio Access Network (RAN) [1].

Despite being a very widely-used last-mile network, the cellular RAN faces unique challenges in its ability to signal congestion end-to-end, to applications and transport protocols. This is broadly because of its complexity, but specifically because the many queues of 5G networks are hidden deep inside the highly-layered 5G RAN protocol suite, whose structure exists alongside, but is mostly opaque to, Internet protocols at all layers of the Internet's protocol stack [37, 47, 56, 57]. The result is that a very large, multi-hop network (the 5G Core sub-network and RAN sub-network) is hidden from L4S, unintentionally subverting the efficiencies of the L4S signaling architecture (see §2 next).

Furthermore, realizing L4S congestion signaling in the RAN faces additional challenges stemming

Authors' Contact Information: Haoran Wan, Princeton University, Princeton, NJ, USA, haoran.w@princeton.edu; Kyle Jamieson, Princeton University, Princeton, NJ, USA, kylej@princeton.edu.

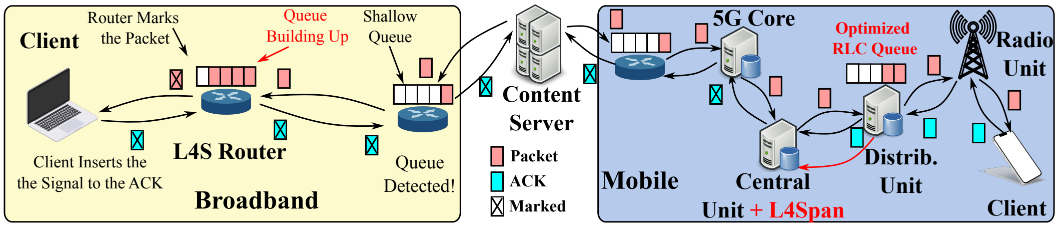


Fig. 1— L4S status quo: Some backbone and broadband ISPs deploy L4S routers (left), but for mobile users, the 5G network is key to performance, yet lacks L4S functionality (right).

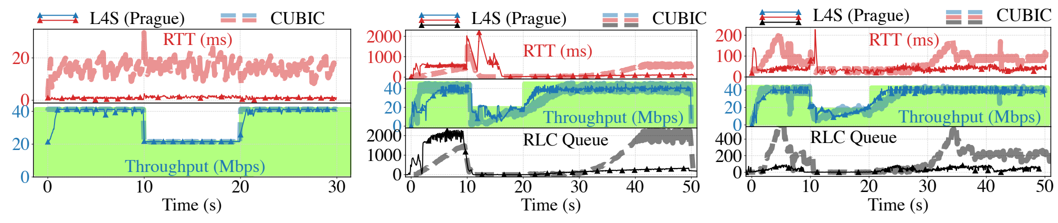
from the fundamental differences between wired and wireless networks. First, the vagaries of wireless propagation, complexities in the handoff process between cell towers, and non-uniform traffic scheduling results in a highly variable capacity. While existing wired L4S routers set a very low queuing delay target (*e.g.*, one millisecond) and mark the ECN bit of all packets if the sojourn time of the queue exceeds it, the cellular RAN thus demands a different approach. Second, compounding the challenge, the RAN is notably flexible in its configuration as a network, adapting to varying traffic demand from even a single client with the capability to dynamically add and remove cell towers from the set of those simultaneously serving that client (also known as carrier aggregation). Third, as we explain in more detail in §4, heterogeneous client capabilities interact with the limited quality-of-service queuing capabilities the 5G RAN currently offers, interacting with the end-to-end congestion control loop in subtle ways.

These challenges motivate a fresh look at the internal data flow signaling mechanisms of the 5G RAN to ask whether it is possible to bring the benefits of L4S in wired networks to 5G. We seek minimal changes to the 5G network that will allow us to monitor all relevant queue occupancies, and predict into the short-term, future queuing delay. Also required is an architectural design that spans the two networks' (5G and Internet) currently-disjoint signaling mechanisms to provide timely congestion signals to L4S-capable as well as legacy content senders.

This paper presents **L4Span**, a new design for the 5G RAN that passively estimates the queuing delay of each mobile user and signals the congestion to the content server sender through markings on the packets' header. With ECN feedback reflecting the 5G network condition, the L4S sender can accurately adjust its sending rate to minimize queuing delay in the 5G Radio Link Control (RLC) layer (see Fig. **1**), while maintaining high throughput. Since wireless throughput is so volatile, L4Span needs to make real-time queuing delay estimates and predictions, for each queue in the 5G network. Our design abstracts the complex queuing and scheduling functions of the Distributed Unit (DU) into a single, agile rate estimate that the DU communicates to the Central Unit (CU) of the RAN, as shown in Fig. **1**. Classic TCP and UDP traffic flows are still a major component of user traffic on the Internet [73], and L4Span also maintains, and improves performance for these flows.

Our design makes the following contributions: **1)** To cope with the wireless RAN's high jitter, L4Span permits slightly larger bounded queues, but also designs a novel packet marking algorithm (§4.2) that takes the telemetry of the RAN scheduler and both L4S and classic end-to-end congestion control algorithms' behavior into account. **2)** To minimize queuing delay, L4Span innovates a queuing sojourn time prediction algorithm (§4.3) that integrates with the 5G RLC layer. We further analyze its performance (§6.3), shedding light on this important fundamental problem. **3)** L4Span provides a reference implementation for the cellular RAN that is currently missing in the L4S ecosystem, achieving low latency and high throughput for both L4S and classic flows.

L4Span is a clean and practical-to-deploy design in the 5G RAN, making minimal changes and modifications to the current 3GPP standardized layer structure and the 5G network design as a



(a) CUBIC and L4S in a **wired network.** **(b)** CUBIC and L4S in a **5G network.** **(c)** CUBIC and L4S in 5G + **L4Span**.

Fig. 2— Performance of L4S and CUBIC in different networks: L4S routers in wired networks achieve line rate and extremely low latency, but are much less effective in 5G networks that don't expose their queues. Both L4S and CUBIC's latency is reduced with L4Span in 5G, and both maintain line rate (green area). In the latter two figures, the bottleneck shifts from the RAN to wired middleboxes at 10s and shifts back at 20s.

whole. L4Span requires only mandatory control messages in 3GPP for its egress rate estimation, and conforms to both 3GPP and O-RAN standards. L4Span performs only ECN bit marking in the TCP and IP headers, making its existence transparent to other components in the RAN. Further, our design only reads and reuses currently-existing messages and state in the 5G RAN to achieve its goal on queuing delay prediction, necessitating no intrusive modifications on the bulk of the RAN. Finally, L4Span considers all the corner RAN configuration cases that might affect the performance, such as different 5G RLC modes (§4.3) and different 5G DRB configurations (§4.2).

Section **2** of this paper goes into further technical detail of L4Span's motivation. Section **3** surveys related work, after which we present a detailed design in Section **4** and our implementation in Section **5**. Our evaluation follows in Section **6**, measuring L4Span operating with many different senders, including TCP Prague, CUBIC, and BBRv2. Top-line results show that L4Span reduces one way delay by up to 98% while maintaining near line-rate throughput in a busy 5G network with 64 concurrent L4S clients. Further results show that L4Span also optimizes the performance of classic flows, achieving up to 97% RTT reduction in a 16-client RAN and reducing the finish time of short-lived TCP flows competing with long-lived background traffic by a factor of 4×. With regards to video conferencing, our results show that L4Span improves the performance of SCReAM interactive video congestion control [29], reducing RTT time by a factor of 3× while maintaining throughput. Section **8** concludes the work. The authors attest that this work raises **no** ethical issues.

2 Background: L4S Signaling Meets the 5G RAN

When an internet content server sends a data segment to a client on an L4S-enabled network, any congested L4S-enabled router on the forward path marks the appropriate ECN bit when its queuing delay exceeds a *sojourn time* threshold τ_s (defaulting to one millisecond) [16, 69]. An L4S receiver then sets the *congestion experienced* (CE) bit in the TCP header of the ACK, and the information reaches the sender within a round-trip time, as shown in Fig. **1** (*upper*). An L4S sender treats the CE feedback as a "lightly-pressed brake," meaning the sender updates the slow start threshold:

$$\text{ssthresh} \leftarrow (1 - \alpha/2) \cdot \text{cwnd},$$

where α is an exponentially-weighted moving average of the fraction of acknowledged bytes with the CE bit marked out of the total bytes receiver over the previous RTT (similar to DCTCP [5]). However, classic TCP flows treat CE feedback as a packet loss, so potentially slow dramatically [31], and thus are prone to starvation relative to L4S flows, and so L4S routers separate L4S and classic flows into two different queues, ensuring the fair bandwidth share between them through a *dual queue coupled* (DualPi2) structure [69]. In a toy topology with one server, one router, and one client, running iperf3 with TCP Prague and CUBIC, Fig. **2(a)** shows TCP Prague (L4S)'s low RTT and line-rate throughput performance in this wired network thus the RLC buffer is not shown. The CUBIC has an RTT around 20 ms, which is the default target for classic flows in the L4S router.

Contrast this with the same content server sending data to a mobile client on a 5G network. The 5G Core (5GC) passes it to the RAN. In the RAN's CU, the *Service Data Adaptation Protocol* (SDAP) layer maps the packet by its quality of service identifier to a *Data Radio Bearer* (DRB), a logical channel that spans the 5G architecture from the 5GC all the way to the UE. The intervening *Packet Data Convergence Protocol* (PDCP) and *Radio Link Control* (RLC) entities are instantiated once per DRB. The PDCP assigns and records a sequence number for the packet, which is then known as a *Service Data Unit* (SDU), so that the RLC in the DU can run an ARQ protocol to ensure (more) reliable delivery over the wireless link. The SDU then moves to the DU where the RLC queues and retransmits it as necessary. It then passes down to lower layers of the Distributed and Radio Units as a *Protocol Data Unit* (PDU) where it is further (independently) reframed and retransmitted as necessary. One or several DRBs and the RLC queues are maintained for each UE, resulting in traffic isolation among and within UEs. The MAC layer in the base station schedules data from RLC queues of each UE, and then the traffic shares the over-the-air physical channel. Key here is the reframing and encapsulation of the datagram at the PDCP and lower layers, resulting in the relevant RLC queue in the DU being hidden to Layers 3 and above. Furthermore, the RLC buffer is designed to be deep for reliable delivery, while, on the contrary, it worsens the sojourn time. Hence, the same CUBIC and L4S flows both experience large RLC queuing delays in a 5G network (Fig. **2(b)**, *lower*), leading to high end-to-end RTT (Fig. **2(b)**, *upper*), thus impacting the performance of interactive applications. Also, the latency breakdown in Fig. **10** shows that the sojourn time in the RLC buffer makes up the majority portion of the one-way delay without L4Span. In contrast, L4Span minimizes queuing delay (Fig. **10**) while simultaneously maintaining throughput (Fig. **2(c)**).

3 Related Work

L4S and 5G Many existing works discuss enabling L4S feedback mechanism in the 5G network through simulation [19, 20, 63, 72], which can't capture the resource allocation dynamics between UEs' transmissions, while L4Span observes the allocation outcome and predicts the sojourn time passively. Brunello [19, 20] proposes to include the ECN feedback in the PDCP layer, but doesn't have a solution for the dominant classic flows [73], and L4Span proposes solutions for both types of flow. Pan [63] and Son [72] conduct trace-driven evaluations, without real RAN dynamics, unlike L4Span's over-the-air implementation. DChannel [70, 71] guides flows to different physical channels for performance gain, neglecting the impact of the internal queues, and L4Span aims to minimize the internal queue occupancy and achieve full throughput utilization for each UE. From the UE perspective, base station selection is explored to achieve performance gain [27] and energy saving [37]. XRC [48] is a rate controller in the RAN, requiring customized end-points, while L4Span optimizes for existing schemes. OutRAN[49] proposes to prioritize the short-lived flows over the long-lived flows in the RLC queue, while L4Span keeps RLC queue occupancy low and both types of flow benefit, as shown in our evaluation in Fig. **11**. RAPID[28] designs a proxy between the 5G core and the content server to prevent RLC buffer overshooting, but RAPID is too far away from the RAN and thus can't adapt as the RAN's changing throughput as L4Span does. TC-RAN[41, 42] implements the Linux queuing discipline, such as CoDel and ECN-CoDel, inside the RAN between SDAP and PDCP layer, and uses a fixed threshold to drop or mark the packets. Evaluation results show that TC-RAN underutilizes the RAN's capacity while L4Span, adapting with the dequeue rate, better utilizes the capacity and achieves low latency. Non-queue-building (NQB) [79] proposes a low-latency architecture similar to L4S, but NQB doesn't aim for full bandwidth utilization, and L4Span can serve similarly. L4Span is the first design that enables congestion feedback for both

L4S and classic flows, and is evaluated with live RAN dynamics and various channel conditions.

Congestion Control. Sprout [80] employs a stochastic model to forecast the cellular queue occupancy. Verus [88] uses a packet delay profile for congestion avoidance, and Copa [8] uses the RTT measures to minimize the queue occupancy. CopaD [40] adapts to cellular networks with parameter tuning. PBE-CC [83] utilizes the RAN telemetry information to regulate the sender, and a similar design is adopted in piStream [81] and CLAW [82]. BBR [21] probes the bandwidth and delay periodically, and BBRv2 and 3 [22] improve the procedure and take the ECN as feedback. Venkat [7] analyzes the starvation in end-to-end congestion controls, while the RAN's behavior is one source of the non-congestive delay. Ferreira [30] proposes a reverse-engineering tool for congestion control algorithms, which better enables middleboxes to assist end-to-end transport. Separated in different RAN's queues, flows have contentions for resource allocation partially based on the backlogged data, not fully complying with the observations in the wide area network [18, 89]. Unlike a new congestion control algorithm that customizes the logic of both sender and receiver, L4Span in the last-mile hop (5G network) improves the latency performance for existing algorithms.

In-network Design. XCP [45] uses a control theory framework to achieve high utilization and low queuing delay. ABC [36] repurposes the ECN field in the IP header as "accelerate" or "brake" command to adjust the sending rate of the sender, making it hard to fit into the L4S architecture. TACK [55] reduces the frequency of ACK in wireless networks and improves throughput and RTT, while L4Span relies on the ACKs for congestion notification. Zhuge [60] delays and/or modifies the ACK in the Wi-Fi for a more timely feedback to the sender, which inspires our RAN short-circuiting design. Octopus [23] drops the packets actively for delay reduction, but requires modifications on the client, server, and RAN, incurring a heavier burden than L4Span. SMUFF [77] aims to achieve the line rate in Wi-Fi direct by filling the router's buffer with the optimal amount of data, and L4Span aims for low latency on top of it. Sidekick [87] provides in-wireless-network feedback for QUIC flow, incurring new out-of-band feedback for the sender, while L4Span leverages only in-band feedback in the existing ACK packets, thus it is more bandwidth efficient.

4 Design

We first summarize L4Span's architecture (§4.1), then drill down into its packet marking strategy (§4.2), sojourn time prediction (§4.3), and feedback short-circuiting mechanism (§4.4). We develop our design in the context of the O-RAN 7.2x split [65] as it is the dominant design direction in 5G currently, and our design generalizes to other O-RAN splits.

4.1 L4Span Functionality

As shown in Fig. **3**, we situate most of L4Span's functionality in modules inside the RAN CU, above the SDAP and PDCP layers that constitute the CU's per-UE state. A new queue feedback path directs downlink data delivery status messages from the RLC in the DU. Three classes of events trigger L4Span's functions: **1)** receiving a downlink datagram from the 5GC, **2)** receiving RAN feedback, and **3)** receiving an uplink ACK—pseudocode can be found in Appendix **A**.

Receiving a downlink datagram: when it receives a datagram from the 5GC, L4Span creates a mapping between the five-tuple [59], and a UE and Data Radio Bearer (DRB) tuple—this separates classic and L4S flows by their ECN field (01 for L4S ECN flows, 10 for classic ECN flows), as introduced above in Section **2**. The five-tuple contains the source and destination IP addresses, ports, and transport protocol, and is unique for each flow and UE. The (UE, DRB) tuple indexes L4Span's *packet profile table* (§4.3.1) for egress rate prediction (§4.3.3) and marking decisions.

Receiving RAN queue feedback: as shown in Fig. **3**, L4Span receives downlink RLC data delivery

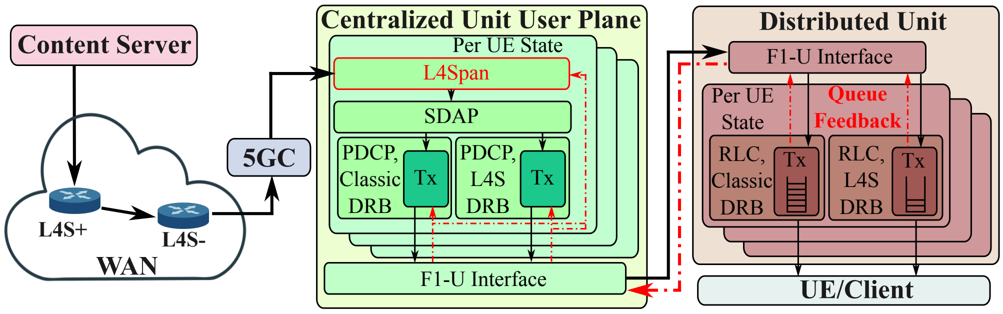


Fig. 3— L4Span in the end-to-end data path from the content server to the client, where the red text and arrows inside the CU-UP and DU illustrate our system design, and the black elements inside the RLC entities mark where queues build up inside the 5G network. L4Span reuses the existing feedback in the RAN. **L4S+**/**L4S-** denotes a router with/without L4S functionality; RU, MAC and PHY layers are not shown.

events over the F1-U interface (§4.3.1), a mandatory 3GPP API from the RLC to the PDCP entity. Upon receiving the feedback, the L4Span layer uses this data to update the corresponding packets' status in the packet profile table and make a marking decision (§4.2).

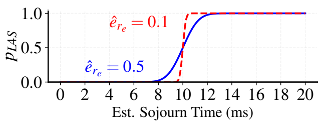
Receiving an uplink ACK: L4Span only operates on the uplink packet if the feedback short-circuiting is possible (TCP, see §4.4). L4Span first reverse-maps the ACK to the correct DRB , and then updates the packet's relevant fields based on the marking decision.

4.2 Marking Strategy

ECN marks inform L4S and classic senders of congestion, so they can change their behavior. But as observed above (§2), different senders react to ECN markings differently, and so we mark packets with different strategies, in analogy to DualPi2 *Active Queue Management* (AQM) [69]. In an L4Span 5G network, a DRB may serve L4S flows only (§4.2.1), classic flows only (§4.2.2), or a mix of both (§4.2.3). L4Span classifies the above scenarios by checking the ECN fields, and work with each.

4.2.1 L4S-Only DRB. Here L4Span drives the queuing delay low by marking aggressively, because the L4S sender treats the CE ECN feedback as a "slightly-pressed brake," resuming additive increase immediately upon receiving non-CE ACKs after the congestion window (cwnd) is cut (§2). Hence an L4S sender's cwnd changes frequently and converges to a small saw-tooth around the optimal operation point [17]. **Marking algorithm:** Given a predicted egress rate $\hat{r}_e$ and its error distribution e_{r_e} (§4.3 describes how to make the estimates), L4Span marks the packet with probability p_{L4S}, the likelihood the actual egress rate satisfies the sojourn time threshold τ_s, given N_{queue} queued bytes:

$$p_{\text{L4S}} = P\left\{ r_e \geq \frac{N_{\text{queue}}}{\tau_{\text{thr}}} \middle| \hat{r}_e, e_{r_e} \right\} = P\left\{ e_{r_e} \leq \frac{N_{\text{queue}}}{\tau_{\text{thr}}} - \hat{r}_e \right\}, \quad (1)$$



where $e_{r_e} \sim \mathcal{N}(0, \hat{e}_{r_e}^2)$ and $\hat{e}_{r_e}$ is the standard deviation of the egress rate over the latest estimation window. In a wired network, DualPi2 estimates the sojourn time by subtracting the ingress timestamps of the queue head packet and tail packet, and marks all packets when the sojourn time exceeds one millisecond. However, the 5G network has a volatile egress rate, making such estimation infeasible. Instead, L4Span uses a varying distribution to calculate mark probability (Eq. 1): if the egress rate is volatile ($\hat{e}_{r_e} \uparrow$), the distribution has a flatter edge at τ_{thr} to avoid potential

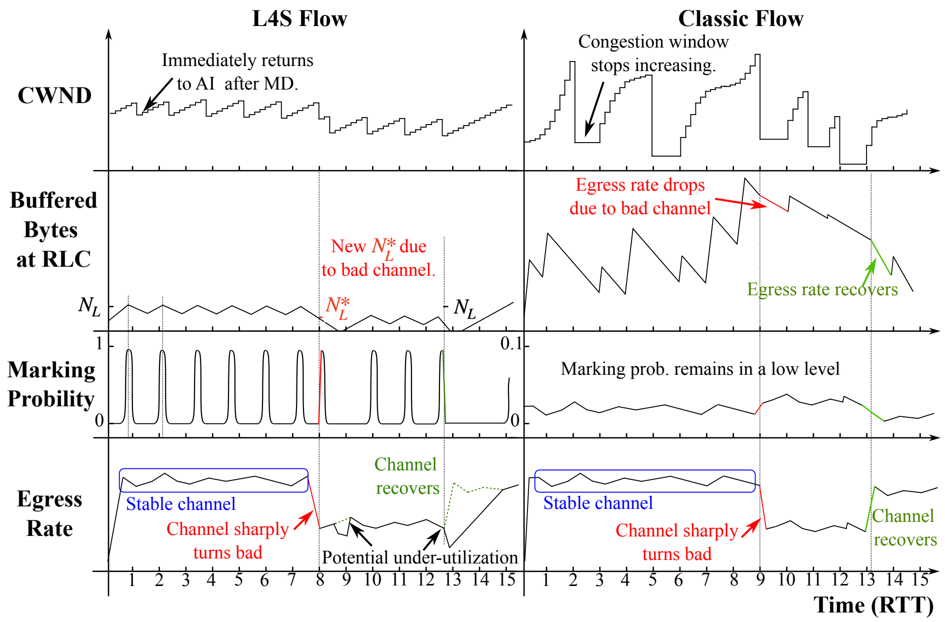


Fig. 4— L4Span's behavior on L4S and classic flows with wireless channel variations.

under-utilization. If the egress rate is stable ($\hat{e}_{r_e} \downarrow$), the distribution has a sharper edge to pursue low latency more aggressively. If the egress rate is invariant ($\hat{e}_{r_e} = 0$), L4Span's marking strategy reduces to the DualPi2 strategy. We set a sojourn time threshold of 10 milliseconds (see §6.3 for justification), as the 5G MAC layer requires an adequately filled buffer for resource scheduling.

The behavior of L4Span, the L4S sender, and the RAN is shown in the running example of Fig. 4 *(left)*. We define a *bytes threshold* $N_L = \hat{r}_e \cdot \tau_s$ in the figure, equivalent to the sojourn time threshold. At the first RTT, the RLC buffer builds up to the threshold, as the L4S sender paces packets in *additive increase* (AI) [51]. In the second RTT, the buffered bytes trigger the marking in Eq. 1, then the L4S sender observes CE signal and conducts a cwnd *multiplicative decrease* (MD). Immediately after the MD, the sender returns to AI, until the next CE signal. In the stable channel, L4Span and the sender maintain a small sawtooth pattern around the delay threshold.

In the seventh RTT, the wireless channel sharply degrades, worsening the egress rate. L4Span detects this and adjusts its mark threshold in the subsequent RTT, forcing the sender to once more cut cwnd. Here, the RLC buffered bytes may drop to zero, and the UE would experience a brief throughput under-utilization, but this is promptly remedied by the sender immediately returning to AI and refilling the RLC buffer. In the 13th RTT, the channel recovers, and increased throughput drains buffers causing potential under-utilization until the L4S sender uses AI to refill the buffer.

4.2.2 Classic-Only DRB. Unlike L4S, a classic sender (*e.g.* CUBIC, Reno) treats the congestion feedback the same way as the packet loss, and reacts by cutting its slow start threshold multiplicatively. With this in mind, we should not aim for a shallow queue for the classic flows, as the TCP endpoint would frequently receive the CE feedback, cut its slow start threshold and suffers from severe starvation [16]. The design goal is to prevent the well-documented buffer bloat [33, 38, 43] and in the meantime, maintain a suitable amount of bytes in the buffer to avoid underutilization.

To prevent buffer bloat, L4Span marks the packets with the probability that matches the average ingress rate with the RAN egress rate to balance the buffer size. The TCP throughput is modeled as $r_{\text{classic}} \approx \text{MSS} \cdot K/(\text{RTT}\sqrt{p_{\text{classic}}})$, where $K = \frac{1+\beta}{2}\sqrt{\frac{2}{1-\beta^2}}$ and β is the MD parameter [62] (0.5 for Reno [64]). L4Span predicts the RAN egress rate $\hat{r}_e$ from its packet profile table (see §4.3.3). To complete the equation, L4Span estimates the initial $\widehat{RTT}^*$ using the interval between the first two forward TCP packets (SYN and the subsequent ACK) of each flow. In the further operation, we add $\widehat{RTT}^*$ with the predicted sojourn time $\hat{\tau}_s^r$ over the last coherence time window (§4.3.3) as the RTT estimates $\widehat{RTT} = \widehat{RTT}^* + \hat{\tau}_s^r$. The marking probability is calculated as:

$$\hat{r}_e = \frac{\text{MSS} \cdot K}{\widehat{RTT}\sqrt{p_{\text{classic}}}} \rightarrow p_{\text{classic}} = \left(\frac{\text{MSS} \cdot K}{\widehat{RTT}\hat{r}_e}\right)^2. \tag{2}$$

$\widehat{RTT}$ is an overestimation of the RTT, as the UE's DRB may have other backlogged data when the handshake happens. Instead of harming the performance, the slightly lower p_{classic}, resulting in a higher ingress rate, helps to build an adequate RLC buffer size and prevent under-utilization (see the evaluation in §6.3.1). If $\widehat{RTT}$ is not available, *e.g.* UDP flows, we use $2\hat{\tau}_s^r$ as the RTT estimates.

Our running example continues in Fig. **4** *(right)* with the behavior of L4Span, a classic sender, and the RAN. Initially, the classic sender sends the packet burst to the RAN, and increases its cwnd with the CUBIC function. The marking probability, calculated from Eq. (2), remains low. When receiving a congestion signal marked by L4Span, the classic sender cuts its cwnd and pauses the AI for one RTT, during which the RAN dequeues the packets. As the channel turns bad, the dequeue rate decreases, resulting in a higher marking probability to regulate the ingress rate. In this period, the sender receives relatively more congestion signals, thus the RAN can drain the queue. As the channel recovers, and the L4Span's's marking probability returns to a lower value, and the sender returns to AI. Compared with the L4S flow's short and more volatile marking behavior, the marking behavior for a classic flow spans a longer time and operates at relatively small values, where the differences are due to the distinct behaviors of the classic and L4S senders.

4.2.3 L4S-Classic Shared DRB. To achieve optimal performance, best practice is to keep L4S and classic flows in separate DRBs of each UE. However, some lower-end UEs do not support multi-DRB configuration, and a new marking scheme is needed, as both types of flows experience performance drop (in RTT or throughput) in such a scenario if unattended (see §6.2.6). To achieve good resource utilization, we keep the classic packet marking probability (p_{classic}). The L4S flow's throughput (MBytes/s) is inversely proportional to the packet mark rate (r_{L4S}) [17, 68]: $r_{\text{L4S}} \approx 2\text{MSS}/(\text{RTT}\cdot p_{\text{L4S}})$, and the r_{classic} is discussed above. To balance two types of flows' throughputs, we mark the L4S flow with a coupled probability $p_{\text{L4S}} = \alpha\sqrt{p_{\text{classic}}}$, where α is the solution of $r_{L4S} = r_{\text{classic}}$ equation, assuming an equal RTT. Here, L4S and classic flows compete fairly in one UE's DRB, while the fairness among different UEs is achieved by the MAC scheduler, working orthogonally with L4Span.

4.3 RAN Queue Occupancy Prediction

Here we introduce the design details of L4Span. To begin with, L4Span re-purposes the existing F1-U messages in the base station. L4Span utilizes the F1-U feedback messages from the RLC to the PDCP entity (§4.3.1) to estimate the dequeue rate, as well as the estimation errors based on a *shadow table* described next (§4.3.2). Then, L4Span uses the egress rate estimations to predict the sojourn time (§4.3.3), and based on the estimations, makes the marking decisions.

4.3.1 F1-U Feedback Loop. The RAN's behavior and the feedback information vary with the RLC mode configured in each DRB: there are two RLC modes for user data delivery: RLC *Acknowledged Mode* AM and RLC *Unacknowledged Mode* (UM), where the latter omits retransmissions and the retransmission queue, and doesn't provide delivery time feedback. The purpose of the F1-U feedback

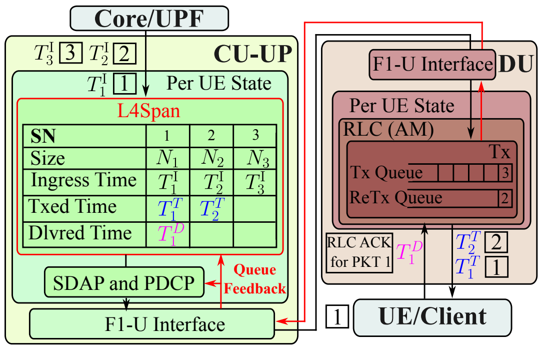


Fig. 5— L4Span's packet profile table based on the RAN feedback information. $T_i^{\{I,T,D\}}$ denotes the ith packet's **I**ngress, **T**ransmission, and **D**elivery timestamps. RU, MAC and PHY components are not shown.

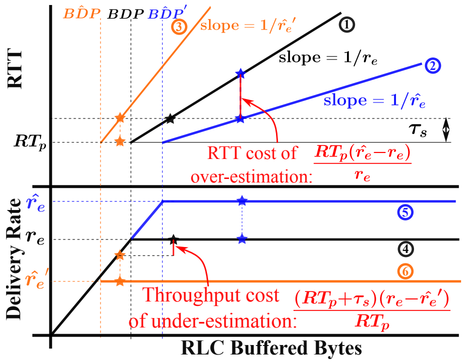


Fig. 6— Cost of errors. L4Span marks the packet using the estimated egress rate and a queuing delay threshold τ_s. Stars (★) mark the ideal operation points with the estimated egress rate.

loop is to expose the RAN's congestion queue to the upper L4Span layer. There are three types of 3GPP-standard [4] F1-U messages: downlink user data, downlink data delivery status, and assistance information. To make our design as general as possible, we use only the mandatory fields of the downlink data delivery message: the highest transmitted PDCP *sequence number* (SN) and the highest delivered PDCP SN. The RLC triggers a timestamped feedback message when it transmits the PDCP SDU down to the MAC/PHY, and when it receives an RLC ACK that indicates SDU delivery from the UE in RLC AM. As we explain next, L4Span handles both AM and UM.

4.3.2 Packet Profile Table: Timekeeping. The L4Span packet profile table tracks packets' progress through the RLC in order to predict queue occupancy. Figure 5 provides a running example, packets with sequence numbers 1, 2 and 3 go through different procedures with a timestamp on each:
1) Entry to CU-UP L4Span layer, recording of *ingress timestamps* T_1^{I}, T_2^{I}, and T_3^{I}.
2) MAC transmits Packets 1 and 2, RLC layer reports the status to PDCP and L4Span, with recorded *transmission timestamps* (T_1^{T}, T_2^{T}).
3) RAN delivers Packet 1 to UE, which sends an RLC acknowledgment. RLC sends the highest delivered sequence number and *delivery timestamp* T_1^{D} to the PDCP and L4Span.

From the packet profile table, we calculate the actual queuing delay of each packet by subtracting the transmitted time and the ingress time (*i.e.*, $T_1^{\mathrm{T}} - T_1^{\mathrm{I}}$ and $T_2^{\mathrm{T}} - T_2^{\mathrm{I}}$), and retransmission delay (*i.e.*, $T_1^{\mathrm{D}} - T_1^{\mathrm{T}}$), if the DRB is configured with RLC AM. To ensure L4Span works in both AM and UM, we estimate queue status with the intersection of the feedback information for both RLC modes—the packet transmit times in the feedback information. However, the calculated queuing delay reflects the status of the previously transmitted packets (Packets 1 and 2 in Fig. 5), while ideally we want to estimate the queuing delay of the current standing queue (Packets 3 and later in Fig. 5), and set the feedback accordingly. To do that, we need to predict the RAN egress rate.

4.3.3 Sojourn Time Prediction and Error Estimation. To predict the RLC queue's packet sojourn time, L4Span monitors and predicts the queue egress rate within a short time window. Upon receiving RAN feedback, L4Span remembers the highest transmitted packet sequence number, say k, then calculates the current egress rate of the DRB using a window with time duration τ_c:

$$r_k^T = \left(\sum\nolimits_{i \in \{i | T_k^T - \tau_c < T_i^T \le T_k^T\}} N_i\right) / \tau_c, \tag{3}$$

where N_i is the packet size and T_i^{T} is the transmission time of the i^{th} packet. We choose τ_c to be half a pre-set *channel coherence time* (the time that the channel response is the same), where the τ_c is measured by [78] in a driving scenario, and can cover most of the daily usage scenario. Then we use another τ_c-long window to calculate the average egress rate, as the smoothed egress rate:

$$\hat{r}_e = \left(\sum\nolimits_{i \in \{i | T_k^T - \tau_c < T_i^T \le T_k^T\}} r_i^T\right) / \left|\{i | T_k^T - \tau_c < T_i^T \le T_k^T\}\right|, \tag{4}$$

where $|\cdot|$ denotes the number of elements. In this way, all the packets used for egress rate estimation are transmitted within $2\tau_c$, the channel coherence time, during which the wireless channel is considered stable. With $N_{queue} = \sum_{i \in \{i | T_i^I > T_k^I\}} N_i$, the estimated sojourn time thus becomes:

$$\hat{\tau}_s^r = \frac{N_{queue}}{\hat{r}_e}. \tag{5}$$

L4Span uses the estimated sojourn time for RTT estimation and marking decision making (§4.2).

Here we analyze the cost of egress rate estimation errors if the generic DualPi2 strategy is adopted. Fig. **6** shows a snapshot of the queue, where the r_e is the dequeue rate and RT_p is the round-trip propagation time. Ideally, DualPi2 operates on the line ① and ④ in the figure, achieving sojourn time and throughput targets. However, when it over-estimates the egress rate ($\hat{r}_e > r_e$, line ② and ⑤ in the figure), causing an under-estimated sojourn time and further more queued bytes, the RTT would increase by $\frac{RT_p(\hat{r}_e - r_e)}{r_e}$, marked by the blue star on the line ①. While if the egress rate is under-estimated ($\hat{r}'_e > r_e$, line ③ and ⑥ in the figure), resulting in overaggressive markings, the throughput would decrease by $\frac{(RT_p + \tau_s)(r_e - \hat{r}'_e)}{RT_p}$, marked by the orange star on the line ④.

To adapt to the volatile wireless throughput, L4Span considers the egress rate estimation errors for the marking strategy. Based on our evaluation in §6.3, the egress rate estimation error ($e_{r_e} = \hat{r}_e - r_e$) follows the Gaussian distribution with a near-zero mean. Given $\hat{r}_e \approx \mathbb{E}(r_e)$ over the latest estimation window, e_{r_e} has the similar variation as r_e. We estimate e_r's standard deviation using the groundtruth dequeue rate's standard deviation within the last τ_c-long window ($\hat{e}_{r_e}$), getting $e_{r_e} \sim N(0, \hat{e}_{r_e}^2)$. We take the sojourn time and egress rate error estimations for the marking decision (§4.2). ML based forecasters [11, 34, 66] could potentially improve the prediction accuracy, but are unsuitable for microsecond-level packet and RAN feedback processing in L4Span (see §6.3.3).

4.4 Feedback Short-circuiting

In the L4S architecture [14], the sender relies on feedback from the client, experiencing the entire RTT, but the 5G network delays packet delivery further, as shown in Fig. **7**.[1] To prevent RAN jitter and delays from delaying the feedback, we propose to short-circuit the RAN by modifying the uplink feedback (Fig. **7**), inspired by prior work [60]. L4S adopts several feedback format variations; L4Span supports all of them: ***1)*** AccECN [15] uses ECN-Echo and option field in the ACK TCP header (Prague [17] or BBRv2 [22]), ***2)*** classic ECN uses [31] ECN-Echo in the ACK's TCP header as feedback [9], and ***3)*** data inside the data payload (SCReAM [29] or QUIC [35]).

Short-circuiting the RAN. L4Span first classifies between AccECN and classic ECN, by checking the TCP header's option field, used by AccECN. If AccECN protocol is adopted, the feedback bits in the TCP header contain the number of packets marked as CE and bytes marked as CE, ECT-1, and ECT-0 [15]. Upon marking, L4Span tentatively marks a downlink packet by storing the number of packets and bytes of all three types. Instead of changing the packet's ECN field, L4Span uses the latest ratio to split the ACKed bytes, and updates the ACK's TCP header accordingly [15]. For

[1]The MAC/PHY and RLC ARQ delay the transport block by eight ms [76, 83, 86], and up to 100 ms [2, 85], respectively, and the scheduling delay approaches tens of milliseconds as UE numbers increase (§6.2).

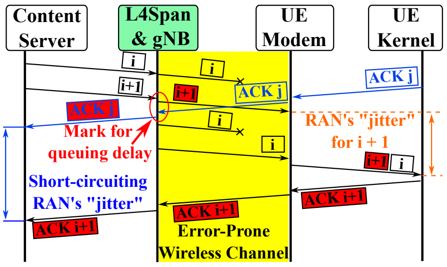


Fig. 7— L4Span's can short-circuit the feedback by modifying the ACK instead of the IP packet for timely adjustments.

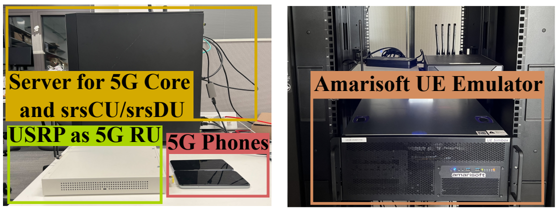


Fig. 8— Hardware used in L4Span implementation. One desktop server runs the 5G Core, srsCU and srsDU, and the USRP is served as the RU. We use both real phones and Amarisoft UE emulator as the UEs for the evaluation.

classic ECN, L4Span keeps marking the ECN-Echo field in the ACK header upon decision, until the CWR flag is set in the downlink packets [59]. L4Span serves as a "bookkeeper" for the client, and short-circuits the RAN's complex jitters, resulting in a better performance in RTT (§6.2.5).

Fallback to Mark Downlink Packet. For the UDP flow, the feedback could be encrypted (QUIC [53]) or presented in customized format (SCReAM [29, 44] and UDP Prague [51]), L4Span fallbacks to mark downlink packets' IP ECN field. Furthermore, L4Span can also be configured to drop packets selectively instead of ECN marking to provide feedback to non-ECN flow senders/

5 Implementation

We implement L4Span on top of srsRAN [74] with approximately 2,000 lines of C++ code (excluding reused code). The L4Span layer is in the CU-UP and inside the UE context, meaning that during the PDU session creation for each UE upon its initial connection, the RAN creates one L4Span entity and connects L4Span to the GTP-U interfance and lower SDAP layer. When a packet is sent from the 5G core to the CU, L4Span checks the packet's ECN field to identify its type. Then, L4Span makes the marking decision with the information from the lower layers and the estimated egress rate and sojourn time. In the meantime, L4Span keeps a copy of the QoS flow and DRB mapping table, which will be used for packet profile table creation and feedback short-circuiting. For the feedback information, we reuse the RAN's message by spawning a dedicated thread in the F1-U interface to call L4Span's feedback handler function upon receiving messages from the RLC/DU, which we use to predict the queue occupancy and make marking decisions. For the coherence time τ_c, we use a measurement in [78] with a 3.5 GHz base station and a moving UE with 70 km/h speed (τ_c = 24.9 ms), covering most of the sub-6 GHz scenarios (evaluated in §6.3), as the higher the center frequency and the faster the UE, the shorter the coherence time. As for the downlink packet processing, L4Span marks the packets on its ECN field if using downlink marking (for UDP or QUIC flows), then it recalculates the CRC checksum on its IP header. For the uplink feedback short-circuiting, L4Span updates the TCP header's Nonce, CWR, and ECN-Echo and TCP options for the AccECN feedback. L4Span then calculates and updates the TCP checksum.

6 Evaluation

We proceed top-down, first introducing our overall methodology, then evaluating it with end-to-end congestion control performance improvements, and we evaluate L4Span's in micro-benchmarks.

6.1 Methodology

We evaluate L4Span in the open-source srsRAN 5G network and Open5GS [61] as the 5G Core, as shown in Fig. 8. We run srsCU and srsDU on a desktop machine, where L4Span is integrated into the CU. The cell is on a TDD band n78 with 30kHz subcarrier spacing, and the cell's center frequency is 3750 MHz with 20 MHz bandwidth, yielding a 40 Mbits/s capacity. We use two types of UEs to generate traffic in the RAN – ***1)*** commercial 5G phones, and ***2)*** a production grade test equipment – Amarisoft UE emulator, which can emulate up to 64 UE over-the-air and emulate different channels. Senders are two Microsoft Azure instances with ping times of 38ms and 106ms.

To begin with, we evaluate the following congestion control schemes' performance in L4Span:

- Prague [17]. TCP Prague is implemented for the L4S architecture, where the client sends the feedback using TCP header fields with AccECN [15].
- CUBIC [39]. CUBIC sender cuts its congestion window to a fixed ratio upon loss or CE. In steady state, it increases its congestion window following the cubic function.
- BBRv2 [22]. BBRv2 includes the DCTCP (or L4S)-like congestion window adjustments upon receiving the AccECN signal. It proactively probes the bandwidth and RTT.

We also evaluate BBR [21] and Reno [64] with L4Span. Please check the Appendix B for their result.

Furthermore, We evaluate two application-level algorithms to analyze L4Span's performance on interactive applications:

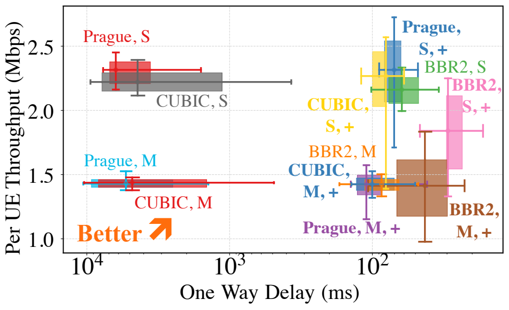
- SCReAM [29, 44]. SCReAM is a congestion control algorithm designed for webRTC [12] over UDP and supports L4S congestion feedback. The receiver reads the number of CE bytes and feedback through RTP feedback.
- UDP Prague [51]. UDP Prague is designed for interactive application in the L4S architecture, where the receiver sends the feedback to the sender in the UDP payload.

6.2 Transport Layer Performance

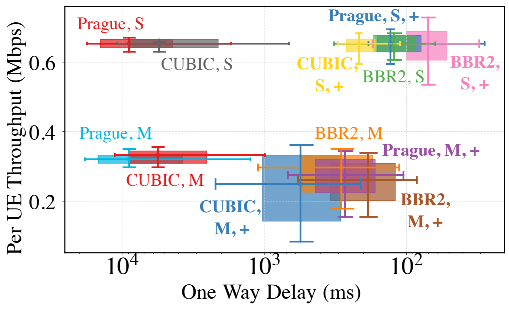
In this evaluation, we evaluate: L4Span's performance impact on ***1)*** widely deployed TCP and ***2)*** interactive video congestion control algorithms; ***3)*** L4Span's impact on the fairness of the RAN, ***4)*** the short-circuiting design's effectiveness, and ***5)*** when L4S and classic flows share the same DRB.

6.2.1 Performance Impacts on TCP Congestion Control. Here we evaluate L4Span's performance impact on congestion control algorithms, including Prague (L4S), BBRv2 (L4S), and CUBIC (classic). We connect 16 and 64 UEs into the RAN through the Amarisoft UE emulator through the over-the-air channel, and all the UEs are doing concurrent TCP downloading through `iperf3`, making the RAN extremely congested. We emulate different channels with the UE emulator, including static, pedestrian- and vehicular-speed channels, and combine the latter two as mobile. We compare different RLC layer queue size settings, including the default srsRAN RLC queue length of 16384 and 256 SDUs. Fig. **9** shows the evaluation result. Across all scenarios, L4Span can massively reduce the one-way delay and maintain a high throughput level. In the default RLC queue setting and with the server of 38ms RTT, L4Span reduces the median one-way delay of Prague by 98.87% and 97.93% in static and mobile channels for 16 UEs with a median throughput drop of less than 1%. As for BBRv2, L4Span reduces the RTT by 52.48% and 52.27% in static and mobile channels, with the cost of 9.8% and 0.08% throughput drop. For CUBIC flow, L4Span can also reduce its one-way delay by 98.85% and 97.11% in static and mobile channels with little median throughput drop. Similar trends can be found in 64 UEs and different RLC queue settings, as well as the 106ms RTT server, shown in Fig. **9(e)** to Fig. **9(h)**. One thing to note is that 256 RLC queue reduces the one-way delay but is less effective than L4Span, as packet drops and retransmissions happen more frequently.

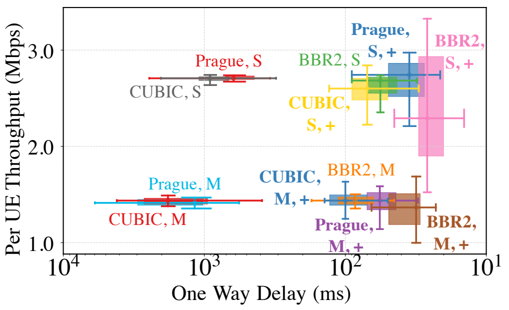
Here we break down Prague's one-way delay, including propagation, scheduling, and queuing delays. The propagation delay is calculated from the NTP-synchronized server and the 5G core's packet timestamp. We breaks the sojourn time into queuing and scheduling delay. The scheduling delay is the wait time for a packet in the queue head for the next transmission opportunity, collected



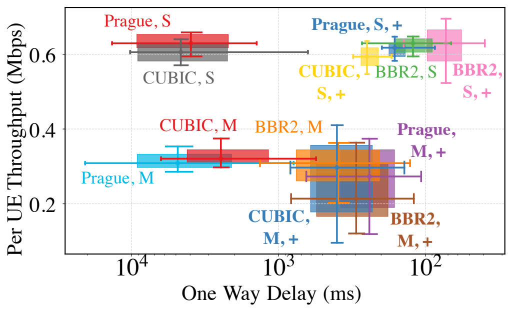
(a) 16 UEs, default RLC queue length, 38 ms RTT.



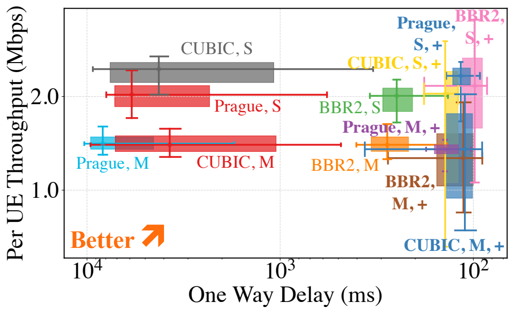
(b) 64 UEs, default RLC queue length, 38 ms RTT.



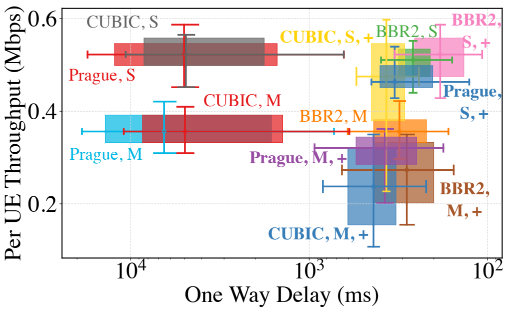
(c) 16 UEs, RLC queue length 256, 38 ms RTT.



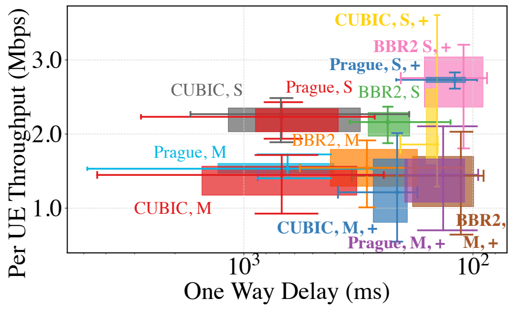
(d) 64 UEs, RLC queue length 256, 38 ms RTT.



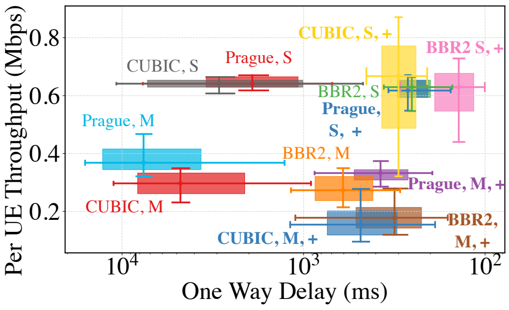
(e) 16 UEs, default RLC queue length, 106 ms RTT.



(f) 64 UEs, default RLC queue length, 106 ms RTT.



(g) 16 UEs, RLC queue length 256, 106 ms RTT.



(h) 64 UEs, RLC queue length 256, 106 ms RTT .

Fig. 9— L4Span improves the performance of various congestion control algorithms in reducing RTT while maintaining throughput, under severely congested RAN and different channel conditions (S: Static, M: Mobile). + indicates L4Span is deployed. Senders are Azure instances with uncongested RAN ping time marked in the caption. Center point: median, box edges: 25th- and 75th-percentile, and whiskers: 10th- and 90th-percentile.

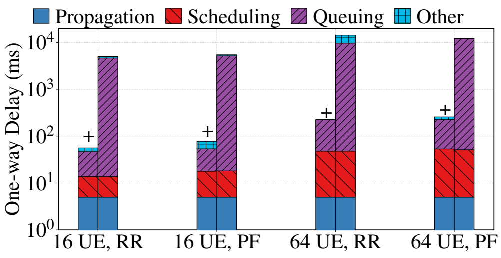


Fig. 10— Average delay break-down for different scheduling algorithms in the srsRAN (RR: round-robin, PF: proportional fair) with different numbers of UEs in the RAN. + marks the L4Span is deployed.

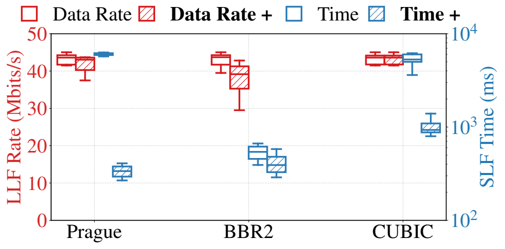
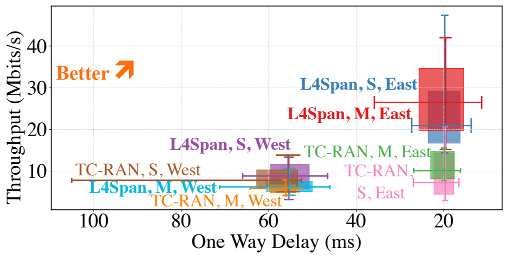
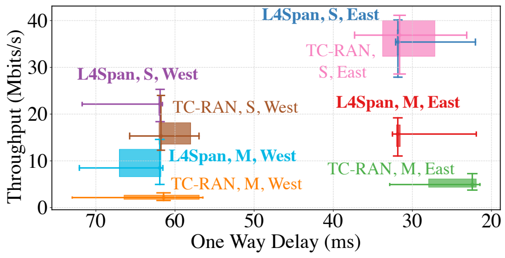


Fig. 11— L4Span improves the finish time of the SLF (14 kilo-bytes) while keeping the LLF' throughput usage (center: median, box edges: 25th- and 75 th-percentile, whiskers: 10th- and 90th-percentile).



(a) Performance of Prague under L4Span and TC-RAN



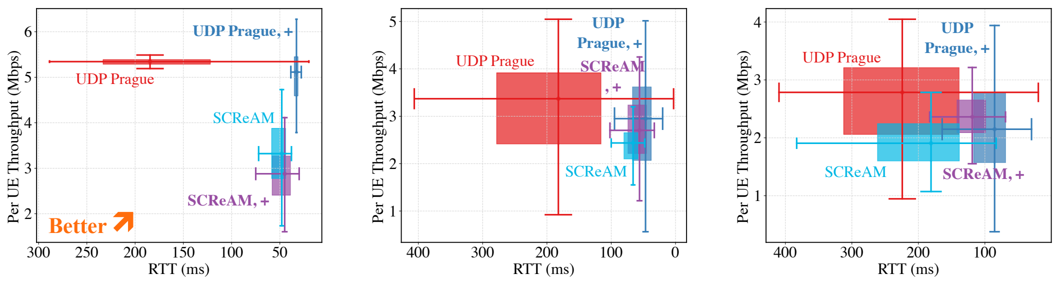
(b) Performance of CUBIC under L4Span and TC-RAN

Fig. 12— Comparison between L4Span and TC-RAN(S: Static, M: Mobile, East: 38ms RTT, West: 106ms RTT).

from the MAC layer log. Queuing delay is the time between when a packet enters the queue and it reaches the head of the queue, calculated by subtracting the scheduling delay from the sojourn time. We compare two scheduling methods – round robin and proportional fair, with 16 and 64 UEs in the RAN. Fig. **10** shows the result of this evaluation, L4Span works with both scheduling schemes.

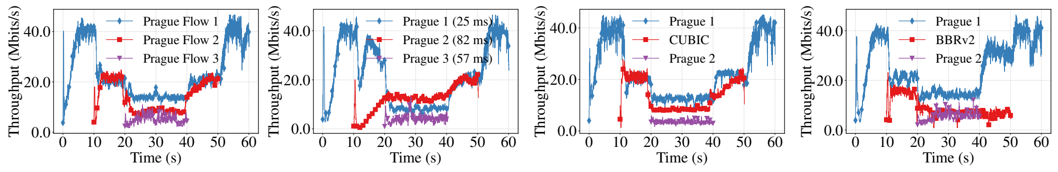
Applications host both *long-lived flows* (LLF) and *short-lived flows* (SLF) are prevalent, such as video on-demand, and web browsing, where the LLF delivers the content and SLF delivers the interactions. Here we evaluate the impact of L4Span on such applications, by running two TCP flows within one commercial UE – one LLF and one SLF, where the SLF's size is 14 kilo bytes. Fig. **11** shows the result; L4Span can reduce the finish time of the SLF, while keeping the LLF's throughput. For TCP Prague, L4Span reduces the SLF's finish time by 94.59%, with 10% of throughput drop for the LLF. Similar performance improvements can be found in BBRv2 and CUBIC flows.

6.2.2 Comparison with Baseline. Here we compare L4Span with the baseline method TC-RAN[42], by connecting one UE into both RANs. We use the same RAN configurations (cell bandwidth, MCS table, etc.) in both L4Span and TC-RAN for a fair comparison. We evaluate the performance of TCP Prague and CUBIC, with default ECN-CoDel and CoDel configuration in the TC-RAN, respectively, and deploy senders in the two Azure instances mentioned above (west with 106ms RTT and east with 38ms RTT). Fig. **12** shows the result of this evaluation. L4Span achieves a similar delay performance with TCP Prague as TC-RAN, but achieves better throughput utilization with improvements of 148% for the static channel and 6% for the mobile channel, as shown in Fig. **12(a)**. Note that there is a latency spike in the TC-RAN, which happens at the beginning of the session due to the longer propagation delay, while L4Span mitigates this with the RAN short-circuiting design. For CUBIC, L4Span achieves a better channel utilization as it tries to align the throughput



(a) Static channel condition. **(b)** Pedestrian channel condition. **(c)** Vehicular channel conditiopn.

Fig. 13— SCREAM and UDP Prague's performance with L4Span under different channel conditions.



(a) Three L4S Prague flows with similar RTT in L4Span. **(b)** Three L4S Prague flows with distinct RTT in L4Span. **(c)** Two L4S Prague flows and one Cubic flow in L4Span. **(d)** Two L4S Prague flows and one BBRv2 flow in L4Span.

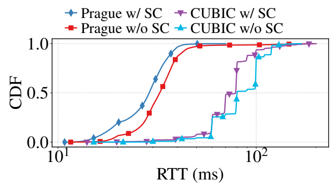
Fig. 14— L4Span's impact on the TCP throughput fairness among different flows.

of the sender and RAN, while TC-RAN uses a fixed threshold in CoDel, leading to under-utilization.

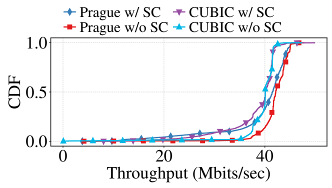
6.2.3 Interactive Application's Congestion Control. Here we evaluate the performance SCReAM and UDP Prague, designed for interactive video applications. Both algorithms support L4S over UDP, thus L4Span disables the RAN short-circuiting. We connect 8 Amarisoft UEs into the network to perform concurrent downlink transmission from a local server, under different channel conditions. Fig. **13** shows the results, where L4Span improves the the RTT performance of both schemes under all channel conditions. Specifically, L4Span reduces the RTT of UPD Prague in static, pedestrian and vehicular channel by 76.33%, 38.04%, and 44.83%, while slightly reduces its throughput by 5.64%, 8.25%, and 17.74%. For SCReAM, L4Span reduces its RTT by 12.60%, 11.20%, and 38.44% in static, pedestrian and vehicular channel conditions, with a little bit higher throughput variations.

6.2.4 Fairness. We evaluate L4Span's impact on the TCP fairness between different UEs in the network. In this evaluation, we connect three commercial UEs into the 5G network with starting time of 0, 10 and 20 seconds and ending time of 60, 50, 40 seconds, and each UE carries one TCP flow. As a result, three Prague flows can maintain their fair share rate as shown in Fig. **14(a)**, and the Prague flow with higher RTT would need more time to converge to the fair share throughput as shown in Fig. **14(b)**. When sharing the RAN with another UE using buffer filling TCP congestion control algorithm (CUBIC), the L4Span maintains the balance among three UEs Fig. **14(c)**. BBRv2, also treated as L4S flow, takes longer to recover, as shown in Fig. **14(d)**. When three flows share the RAN simultaneously between 20 and 40s, the MAC scheduler determines the resource allocation based on each UE's configurations and channel conditions, resulting in slight different throughputs.

6.2.5 Feedback Short-circuiting. Here we evaluate the effectiveness of L4Span's short-circuiting design. During this evaluation, one UE is connected to the 5G network with L4S Prague or classic



(a) RTT performance.



(b) Throughput performance.

Fig. 15— L4Span's performance comparison when it uses feedback short-circuiting (SC in the figure) or not for L4S and classic flows.

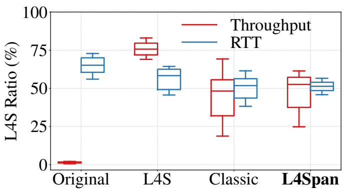


Fig. 16— When L4S and classic flows share a DRB, L4Span achieves fair throughput and RTT.

CUBIC flow and communicates with a local server to rule out other delay impacts. Fig. **15** shows the results of this evaluation. L4Span can achieve a lower RTT for both Prague (28.52 ms v.s. 33.87 ms) and CUBIC (75.27 ms v.s. 85.42 ms) flows on average and on 99.9 percentile tail for Prague (52.97 ms v.s. 179.34 ms) and CUBIC (160.42 ms v.s. 190.53 ms) with short-circuits, as shown in Fig. **15(a)**. Also, the short-circuiting doesn't affect the throughput performance much (Fig. **15(b)**).

6.2.6 DRB shared by L4S and classic flows. L4Span has a separate marking strategy when both flows share the same DRB (§4.2.3). During this evaluation, we connect one phone into the 5G network and start two flows – one for Prague and one for CUBIC. Fig. **16** shows the result of this evaluation, where the y-axis is the ratio of L4S in RTT ($RTT_{L4S}/(RTT_{L4S} + RTT_{Classic})$) and throughput ($r_{L4s}/(r_{L4s} + r_{Classic})$). We evaluate four marking methods, as listed in the x-axis labels. Firstly, we keep the original marking strategy as they don't share the queue ("Original" in Fig. **16**), where the L4S flow starves. Secondly, we mark both flows with the L4S strategy in §4.2.1 ("L4S" in Fig. **16**), where the classic flow is starved with a throughput ratio of round 25%, as the classic flow slows it sending more than the L4S flow. Thirdly, we mark both flows with the classic strategy in §4.2.2 ("Classic" in Fig. **16**), causing a larger throughput variation. The marking strategy in §4.2.3 ("L4Span in Fig. **16**") performs the best – fair share capacity and the smallest variations.

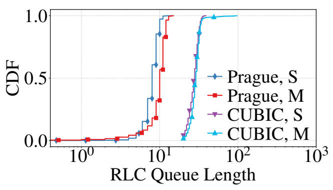
6.3 Microbenchmarks

In this evaluation, we ***1)*** illustrate the Dualpi2's marking strategy is unsuitable for wireless networks, ***2)*** validate the parameter selections in our design, and ***3)*** evaluate L4Span's processing time.

6.3.1 Marking Behavior. We compare L4Span against ***1)*** Dualpi2 [69], and ***2)*** DualPi2 + 10ms threshold to show the marking strategy in the wired network can't be applied directly to the wireless scenario. We reimplement DualPi2 to replace L4Span and evaluate TCP Prague and BBRv2. DualPi2's sojourn time calculation can't capture the wireless channel variations, resulting in severe under-utilization – 73% and 28% lower throughput with thresholds of 1 and 10 ms, respectively.

We evaluate the queue length of the L4S flow and classic flow with L4Span to demonstrate that the queue occupancy rarely reaches zero, leading to underutilization. Fig. **17** shows the result, where in all the scenarios, the classic flow RLC queue length doesn't fall to zero. And the L4S flow maintains a low queue occupancy and achieves ultra-low queuing delay.

6.3.2 Parameter Selection. To verify our window selection – half of the pre-set coherence time (24.9 ms), for egress rate estimation, we leverage a telemetry tool called NR-Scope [76] to collect the DCIs in the two commercial base stations (a 2.5GHz TDD cell and a 600 MHz FDD cell) near our lab. We count the period during which the MCS index's deviation is within 5 as an estimation of channel stable period, and include periods shorter than 1 s in the statistics. Fig. **18** shows the measurement result, and the dashed vertical line marks our window sizes, where most of the time


(a) 16 UE scenario.

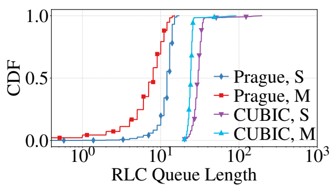

(b) 64 UE scenario.

Fig. 17— RLC queue CDF (S: Static, M: Mobile).

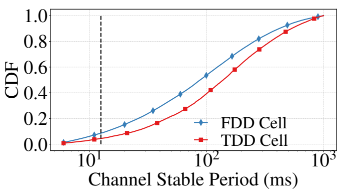


Fig. 18— Channel stable period of T-Mobile cells.

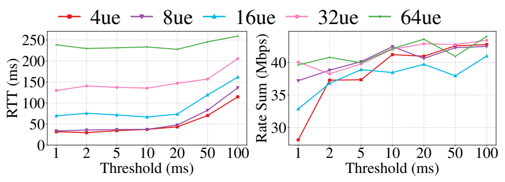

(b) τ_s's impact on RTT. (c) τ_s's impact on rate.

Fig. 19— Impact of τ_s on performance.

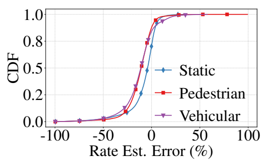


Fig. 20— Egress rate estimation errors.

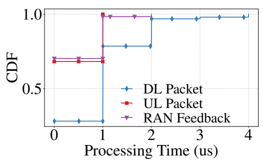


Fig. 21— Processing time of L4Span on three events.

Client Types	srsRAN (idle)	srsRAN+L4Span (idle)	srsRAN (busy)	srsRAN+L4Span (busy)
CPU Usage	2.54%	4.44%	13.44%	14.96%
Memory Usage	6.93%	6.94%	7.28%	7.30%

Table 1— L4Span's CPU (13700K) and memory (64GB) usage compared with the original srsRAN.

(>90%) the channel stable period is higher than our window size[2].

To keep a low queuing delay for the L4S flow, L4Span utilizes a sojourn time threshold (τ_s), which is 10 millisecond in the paper. Here we evaluate how τ_s affects the performance of Prague. Fig. **19** shows the result of this evaluation with each point showing the mean value. As the queuing delay threshold grows higher, the throughput reaches a good level at $10ms$ threshold with a low RTT.

To evaluate the RLC layer egress estimation accuracy, we connect 16 UEs into the network with three channel conditions and compare the RLC layer log with the estimated egress rate at L4Span. Fig. **20** shows the measurements, and most of the time the RLC layer egress rate errors at L4Span are near 0% across three channel conditions.

6.3.3 System Performance. Here we evaluate the processing time of the three events of L4Span (§4), on RAN feedback, on downlink and uplink packets. We run srsRAN with L4Span on a 24-Core i7-13700K machine with 64 GB memory, connect 64 UEs to the RAN, and collect the processing time of L4Span when all the UEs are downloading simultaneously. Fig. **21** shows the results, L4Span finishes its job within 2 microseconds for uplink packet and feedback information processing, and above 50% of the processing is done within 1 microsecond. For the downlink packet processing, L4Span finishes 97% of its process in less than 2 microseconds, and in rare cases, it takes 4 microseconds.

We evaluate the CPU and memory usage of L4Span by comparing it with the original srsRAN on the same machine and in two different operation states: 1) idle – no user, and 2) busy – 64 UEs downloading concurrently. Table **1** shows the result of this evaluation, and L4Span incurs less than 2% of more CPU and less than 0.02% of more memory usage compared with the original srsRAN.

[2]One thing to note is that MCS depends on both channel condition and buffered bytes, not fully reflecting the channel conditions. For example, if the buffered bytes are low, the RAN would use a low MCS even if the channel condition is great.

7 Discussion

Compatibility with other RAN technologies. L4Span is compatible with other RAN lower or upper layer technologies, such as *carrier aggregation* (CA) [54, 84], MIMO, slicing [6, 24, 25], and handover. CA and MIMO only change the workflow of MAC and PHY layers, captured by L4Span egress rate prediction. Slicing maps users into different UE context groups [10], while L4Span works with the physical resources allocated to the client. Upon handover, the buffered bytes are sent to a new RAN, and the markings are already done based on the old estimates. The negative effect won't last long as the buffer is kept low by L4Span, we leave the evaluation as future work.
QUIC Transport Protocols. QUIC flow is another prevalent Internet traffic type, equipped with end-to-end data encryption. To harness the advantage of L4S architecture, Google implements the Prague [35] and BBRv2 as the underlying congestion control schemes, where the receiver reads the clear IP ECN field marked by middleboxes and bounces back the feedback. L4Span can improve QUIC flows' performance by marking on their outer IP headers, similar to UDP flows (§6.2.3).
5G uplink. The design in this paper focuses on downlink; we leave the complementary uplink design to future work. In the uplink, the buffer is inside each client [46], and different strategies, such as pacing the packet to follow the RAN's transmission patterns, can be adopted.
Alternate Designs. Here we list design possibilities other than the foregoing for the placement of L4Span's and arguments against their adoption: ***1)*** *5G Core/UPF:* There is no DRB state stored inside the 5G Core/UPF, so the gNB has to send that state and the queue status, to the UPF, which incurs extra communication delay, lengthening the feedback loop. ***2)*** *SDAP (CU-UP):* While we could implement marking in the SDAP layer, L4Span operates in the TCP/IP layer, while the SDAP is its own layer (with an associated header) in the 5G protocol stack, so layering principles favor this design less. ***3)*** *PDCP (CU-UP) and DU:* While each PDCP entity only has visibility into its own packet queue, 5G may route an uplink feedback packet via a different DRB than the downlink packet that elicited it. Hence, this design choice precludes our proposed feedback short-circuiting design (§4.4). IP header compression in the PDCP layer [3] precludes packet marking in lower layers. ***4)*** *O-RAN RICs:* RICs incur extra networking delay [65], real-time RICs only work in DU [50, 52].
Malicious Behaviors. In the 5G scenario, if a sender or a middlebox mingles the ECN field or adversarially ignores the ECN feedback, the damage to the network is minimal. UEs' queues are isolated (§2), and filling one queue only affects one specific UE's traffic in one DRB, while traffic in other DRBs and UEs still get their fair share of resources guaranteed by the MAC scheduler.
Incentives for ISPs. Deploying L4S in the 5G and wired networks improves the latency performance, and benefits the ISPs' competitiveness. Furthermore, L4S functionality only needs to be deployed at the bottleneck of the data path with a controllable cost. Thus, many applications (Apple FaceTime[75], Google QUIC[35]) and ISPs (Comcast[26], T-Mobile[67]) are already rolling out L4S.

8 Conclusion

In this paper, we have proposed L4Span, the first network architecture proposal that spans L4S signaling over 5G networks in a practical and high performance implementation. Our experimental evaluation demonstrates that L4Span can optimize the performance of both L4S and classic flows in different applications and transport layer technologies. Furthermore, L4Span's design covers the many possible 5G configurations, making it practical for deployment today.

Acknowledgements

This material is based upon work supported by the National Science Foundation under grants AST-2232457, CNS-2223556, ITE-2326928, and OAC-2429485. We gratefully acknowledge a gift from Princeton NextG industrial affiliate program member Qualcomm Corporation.

References

[1] 3GPP. 5G architecture support for XR and media services, 2023.

[2] 3GPP. Specification # 38.322, 2025.

[3] 3GPP. TS#38.321, 2025.

[4] 3GPP. TS#38.425, 2025.

[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan. Data center TCP (DCTCP). *ACM SIGCOMM Computer Communication Review*, **40**(4), 63–74, 2010. ISSN 0146-4833. doi:10.1145/1851275.1851192.

[6] Q. An, D. Pandey, R. Doost-Mohammady, A. Sabharwal, S. Shakkottai. Helix: A RAN Slicing Based Scheduling Framework for Massive MIMO Networks. *Proc. ACM Netw.*, **2**(CoNEXT4), 27:1–27:22, 2024. doi:10.1145/3696399.

[7] V. Arun, M. Alizadeh, H. Balakrishnan. Starvation in end-to-end congestion control. *Proceedings of the ACM SIGCOMM 2022 Conference*, 177–192. ACM, Amsterdam Netherlands, 2022. ISBN 978-1-4503-9420-8. doi:10.1145/3544216.3544223.

[8] V. Arun, H. Balakrishnan. Copa: Practical Delay-Based Congestion Control for the Internet. *Proceedings of the Applied Networking Research Workshop*, 19–19. ACM, Montreal QC Canada, 2018. ISBN 978-1-4503-5585-8. doi:10.1145/3232755.3232783.

[9] F. Baker, G. Fairhurst. IETF Recommendations Regarding Active Queue Management. Request for Comments RFC 7567, Internet Engineering Task Force, 2015. doi:10.17487/RFC7567. Num Pages: 31.

[10] A. Balasingam, M. Kotaru, P. Bahl. {Application-Level} Service Assurance with 5G {RAN} Slicing. *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 841–857, 2024. ISBN 978-1-939133-39-7.

[11] O. Basit, P. Dinh, I. Khan, Z. J. Kong, Y. C. Hu, D. Koutsonikolas, M. Lee, C. Liu. On the Predictability of Fine-Grained Cellular Network Throughput Using Machine Learning Models. *2024 IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*, 47–56, 2024. doi:10.1109/MASS62177.2024.00018. ISSN: 2155-6814.

[12] N. Blum, S. Lachapelle, H. Alvestrand. WebRTC: real-time communication for the open web platform. *Communications of the ACM*, **64**(8), 50–54, 2021. ISSN 0001-0782, 1557-7317. doi:10.1145/3453182.

[13] B. Briscoe, K. De Schepper. Resolving Tensions between Congestion Control Scaling Requirements, 2019. doi:10.48550/arXiv.1904.07605. ArXiv:1904.07605 [cs].

[14] B. Briscoe, K. De Schepper, M. Bagnulo, G. White. *RFC 9330: Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture.* RFC Editor, USA, 2023.

[15] B. Briscoe, M. Kühlewind, R. Scheffenegger. More Accurate ECN Feedback in TCP. Internet Draft draft-ietf-tcpm-accurate-ecn-22, Internet Engineering Task Force, 2022. Num Pages: 63.

[16] B. Briscoe, K. D. Schepper, M. Bagnulo, G. White. Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture. Request for Comments RFC 9330, Internet Engineering Task Force, 2023. doi:10.17487/RFC9330. Num Pages: 36.

[17] B. Briscoe, K. D. Schepper, O. Tilmans, M. Kuhlewind, J. Misund. Implementing the 'Prague Requirements' for Low Latency Low Loss Scalable Throughput (L4S). *Briscoe Bob*, 2018.

[18] L. Brown, Y. Kothari, A. Narayan, A. Krishnamurthy, A. Panda, J. Sherry, S. Shenker. How I Learned to Stop Worrying About CCA Contention. *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, HotNets '23, 229–237. Association for Computing Machinery, New York, NY, USA, 2023. ISBN 9798400704154. doi:10.1145/3626111.3628204.

[19] D. Brunello. *L4S in 5G networks.* KTH ROYAL INSTITUTE OF TECHNOLOGY, 2020.

[20] D. Brunello, I. Johansson S, M. Ozger, C. Cavdar. Low Latency Low Loss Scalable Throughput in 5G Networks. *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 1–7. IEEE, Helsinki, Finland, 2021. ISBN 978-1-72818-964-2. doi:10.1109/VTC2021-Spring51267.2021.9448764.

[21] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson. BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue*, **14**(5), 20–53, 2016. ISSN 1542-7730. doi:10.1145/3012426.3022184.

[22] N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, V. Jacobson. BBR v2 A Model-based Congestion Control. *IETF*, 2019.

[23] Y. Chen, A. Tahir, F. Y. Yan, R. Mittal. Octopus: In-Network Content Adaptation to Control Congestion on 5G Links. *2023 IEEE/ACM Symposium on Edge Computing (SEC)*, 199–214, 2023. doi:10.1145/3583740.3628438. ISSN: 2837-4827.

[24] Y. Chen, R. Yao, H. Hassanieh, R. Mittal. {Channel-Aware} 5G {RAN} Slicing with Customizable Schedulers. 1767–1782, 2023. ISBN 978-1-939133-33-5.

[25] H. Cheng, S. D'Oro, R. Gangula, S. Velumani, D. Villa, L. Bonati, M. Polese, T. Melodia, G. Arrobo, C. Maciocco. ORANSlice: An Open Source 5G Network Slicing Platform for O-RAN. *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, 2297–2302. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400704895. doi:10.1145/3636534.3701544.

[26] Comcast. Comcast is rolling out 'ultra-low lag' tech that could fix the internet | The Verge, 2025.

[27] H. Deng, Q. Li, J. Huang, C. Peng. iCellSpeed: increasing cellular data speed with device-assisted cell selection. *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, 1–13. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 978-1-4503-7085-1. doi:10.1145/3372224.3419201.

[28] M. Diarra, W. Dabbous, A. Ismail, B. Tetu, T. Turletti. RAPID: A RAN-aware performance enhancing proxy for high throughput low delay flows in MEC-enabled cellular networks. *Computer Networks*, **218**, 109,357, 2022. ISSN 1389-1286. doi:10.1016/j.comnet.2022.109357.

[29] EricssonResearch. EricssonResearch/scream, 2025. Original-date: 2015-01-27T08:32:57Z.

[30] M. Ferreira, R. Ware, Y. Kothari, I. Lynce, R. Martins, A. Narayan, J. Sherry. Reverse-Engineering Congestion Control Algorithm Behavior. *Proceedings of the 2024 ACM on Internet Measurement Conference*, IMC '24, 401–414. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400705922. doi:10.1145/3646547.3688443.

[31] S. Floyd, K. K. Ramakrishnan, D. L. Black. The Addition of Explicit Congestion Notification (ECN) to IP. Request for Comments RFC 3168, Internet Engineering Task Force, 2001. doi:10.17487/RFC3168. Num Pages: 63.

[32] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, K. Winstein. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI).* USENIX Association, 2018.

[33] J. Gettys, K. Nichols. Bufferbloat: Dark Buffers in the Internet: Networks without effective AQM may again be vulnerable to congestion collapse. *Queue*, **9**(11), 40–54, 2011. ISSN 1542-7730. doi:10.1145/2063166.2071893.

[34] S. M. Gholian, C. Fiandrino, N. Vallina-Rodríguez, M. Fiore, J. Widmer. DeExp: Revealing Model Vulnerabilities for Spatio-Temporal Mobile Traffic Forecasting With Explainable AI. *IEEE Transactions on Mobile Computing*, **24**(6), 5245–5263, 2025. ISSN 1558-0660. doi:10.1109/TMC.2025.3531544.

[35] Google. quiche/quiche/quic/core/congestion_control/bbr2_sender.cc at main · google/quiche.

[36] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, H. Balakrishnan. {ABC}: A Simple Explicit Congestion Controller for Wireless Networks. *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 353–372, 2020. ISBN 978-1-939133-13-7.

[37] K.-J. Grinnemo. Optimizing Energy Consumption in NB-IoT Networks through Enhanced Cell Selection and Reselection Strategy. *IEEE WoWMoM*, 2025.

[38] Y. Guo, F. Qian, Q. A. Chen, Z. M. Mao, S. Sen. Understanding On-device Bufferbloat for Cellular Upload. *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, 303–317. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 978-1-4503-4526-2. doi:10.1145/2987443.2987490.

[39] S. Ha, I. Rhee, L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, **42**(5), 64–74, 2008. ISSN 0163-5980. doi:10.1145/1400097.1400105.

[40] H. Haile, K.-J. Grinnemo, S. Ferlin, P. Hurtig, A. Brunstrom. Copa-D: Delay Consistent Copa for Dynamic Cellular Networks. *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 508–513, 2023. doi:10.1109/EuCNC/6GSummit58263.2023.10188233. ISSN: 2575-4912.

[41] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt, N. Nikaein. Preventing RLC Buffer Sojourn Delays in 5G. *IEEE Access*, **9**, 39,466–39,488, 2021. ISSN 2169-3536. doi:10.1109/ACCESS.2021.3063769.

[42] M. Irazabal, N. Nikaein. TC-RAN: A Programmable Traffic Control Service Model for 5G/6G SD-RAN. *IEEE Journal on Selected Areas in Communications*, **42**(2), 406–419, 2024. ISSN 1558-0008. doi:10.1109/JSAC.2023.3336162.

[43] H. Jiang, Z. Liu, Y. Wang, K. Lee, I. Rhee. Understanding bufferbloat in cellular networks. *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, CellNet '12, 1–6. Association for Computing Machinery, New York, NY, USA, 2012. ISBN 978-1-4503-1475-6. doi:10.1145/2342468.2342470.

[44] I. Johansson. Self-clocked rate adaptation for conversational video in LTE. *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, CSWS '14, 51–56. Association for Computing Machinery, New York, NY, USA, 2014. ISBN 978-1-4503-2991-0. doi:10.1145/2630088.2631976.

[45] D. Katabi, M. Handley, C. Rohrs. Congestion control for high bandwidth-delay product networks. *SIGCOMM Comput. Commun. Rev.*, **32**(4), 89–102, 2002. ISSN 0146-4833. doi:10.1145/964725.633035.

[46] I. Khan, M. Ghoshal, J. Angjo, S. Dimce, M. Hussain, P. Parastar, Y. Yu, C. Fiandrino, C. Orfanidis, S. Aggarwal, A. C. Aguiar, O. Alay, C. F. Chiasserini, F. Dressler, Y. C. Hu, S. Y. Ko, D. Koutsonikolas, J. Widmer. How Mature is 5G Deployment? A Cross-Sectional, Year-Long Study of 5G Uplink Performance. *2024 IFIP Networking Conference (IFIP Networking)*, 276–284, 2024. doi:10.23919/IFIPNetworking62109.2024.10619877. ISSN: 1861-2288.

[47] I. Khan, T. X. Tran, M. Hiltunen, T. Karagioules, D. Koutsonikolas. An Experimental Study of Low-Latency Video Streaming over 5G, 2024. doi:10.48550/arXiv.2403.00752. ArXiv:2403.00752 [cs].

[48] M. Kheirkhah, M. M. Kassem, G. Fairhurst, M. K. Marina. XRC: An Explicit Rate Control for Future Cellular Networks. *ICC 2022 - IEEE International Conference on Communications*, 2290–2296, 2022. doi:10.1109/ICC45855.2022.9839150. ISSN: 1938-1883.

[49] J. Kim, Y. Lee, H. Lim, Y. Jung, S. M. Kim, D. Han. OutRAN: co-optimizing for flow completion time in radio access network. *Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies*, CoNEXT '22, 369–385. Association for Computing Machinery, New York, NY, USA, 2022. ISBN 978-1-4503-9508-3. doi:10.1145/3555050.3569122.

[50] W.-H. Ko, U. Ghosh, U. Dinesha, R. Wu, S. Shakkottai, D. Bharadia. {EdgeRIC}: Empowering Real-time Intelligent Optimization and Control in {NextG} Cellular Networks. *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 1315–1330, 2024. ISBN 978-1-939133-39-7.

[51] L4STeam. L4STeam/udp_prague, 2025. Original-date: 2024-07-20T17:52:19Z.

[52] A. Lacava, L. Bonati, N. Mohamadi, R. Gangula, F. Kaltenberger, P. Johari, S. D'Oro, F. Cuomo, M. Polese, T. Melodia. dApps: Enabling Real-Time AI-Based Open RAN Control, 2025. doi:10.48550/arXiv.2501.16502. ArXiv:2501.16502 [cs].

[53] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, Z. Shi. The QUIC Transport Protocol: Design and Internet-Scale Deployment. *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, 183–196. Association for Computing Machinery, New York, NY, USA, 2017. ISBN 978-1-4503-4653-5. doi:10.1145/3098822.3098842.

[54] Q. Li, Z. Zhang, Y. Liu, Z. Tan, C. Peng, S. Lu. CA++: Enhancing Carrier Aggregation Beyond 5G. *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '23, 1–14. Association for Computing Machinery, New York, NY, USA, 2023. ISBN 978-1-4503-9990-6. doi:10.1145/3570361.3592500.

[55] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, K. Tan. TACK: Improving Wireless Transport Performance by Taming Acknowledgments. *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '20, 15–30. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 978-1-4503-7955-7. doi:10.1145/3387514.3405850.

[56] Y. Liu, C. Peng. A Close Look at 5G in the Wild: Unrealized Potentials and Implications. *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 1–10, 2023. doi:10.1109/INFOCOM53939.2023.10229016. ISSN: 2641-9874.

[57] Y. Liu, C. Peng. Handling Failures in Secondary Radio Access Failure Handling in Operational 5G

Networks. *IEEE Transactions on Mobile Computing*, **24**(2), 956–969, 2025. ISSN 1558-0660. doi:10.1109/TMC.2024.3477462.

[58] J. Livingood. Comcast Kicks Off Industry's First Low Latency DOCSIS Field Trials, 2023.

[59] P. Matthews, I. v. Beijnum, M. Bagnulo. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. Request for Comments RFC 6146, Internet Engineering Task Force, 2011. doi:10.17487/RFC6146. Num Pages: 45.

[60] Z. Meng, Y. Guo, C. Sun, B. Wang, J. Sherry, H. H. Liu, M. Xu. Achieving consistent low latency for wireless real-time communications with the shortest control loop. *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, 193–206. Association for Computing Machinery, New York, NY, USA, 2022. ISBN 978-1-4503-9420-8. doi:10.1145/3544216.3544225.

[61] Open5GS. https://open5gs.org/.

[62] J. Padhye, V. Firoiu, D. Towsley, J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '98, 303–314. Association for Computing Machinery, New York, NY, USA, 1998. ISBN 978-1-58113-003-4. doi:10.1145/285237.285291.

[63] G. Pan, S. Xu, P. Jiang. Optimizing 5G-Advanced Networks for Time-Critical Applications: The Role of L4S. *IEEE Wireless Communications*, 1–8, 2024. ISSN 1558-0687. doi:10.1109/MWC.004.2400094. Conference Name: IEEE Wireless Communications.

[64] V. Paxson, M. Allman, W. R. Stevens. TCP Congestion Control. Request for Comments RFC 2581, Internet Engineering Task Force, 1999. doi:10.17487/RFC2581. Num Pages: 14.

[65] M. Polese, L. Bonati, S. D'Oro, S. Basagni, T. Melodia. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials*, **25**(2), 1376–1411, 2023. ISSN 1553-877X. doi:10.1109/COMST.2023.3239220. Conference Name: IEEE Communications Surveys & Tutorials.

[66] P. F. Pérez, C. Fiandrino, E. P. Gómez, H. Mohammadalizadeh, M. Fiore, J. Widmer. AIChronoLens: AI/ML Explainability for Time Series Forecasting in Mobile Networks. *IEEE Transactions on Mobile Computing*, 1–15, 2025. ISSN 1558-0660. doi:10.1109/TMC.2025.3554035.

[67] J. Saw. T-Mobile Is First to Unlock L4S in Wireless — A Key Step Toward a Smarter, Programmable 5G - T-Mobile Newsroom, 2025.

[68] K. D. Schepper. Understanding Prague for L4S.

[69] K. D. Schepper, B. Briscoe, G. White. Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S). Request for Comments RFC 9332, Internet Engineering Task Force, 2023. doi:10.17487/RFC9332. Num Pages: 52.

[70] W. Sentosa, B. Chandrasekaran, P. B. Godfrey, H. Hassanieh, B. Maggs. {DChannel}: Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels. 419–436, 2023. ISBN 9781939133335.

[71] W. Sentosa, B. Chandrasekaran, P. B. Godfrey, H. Hassanieh, B. Maggs, A. Singla. Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels. *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, HotMobile '21, 1–7. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 978-1-4503-8323-3. doi:10.1145/3446382.3448357.

[72] J. Son, Y. Sanchez, C. Hampe, D. Schnieders, T. Schierl, C. Hellge. L4S Congestion Control Algorithm for Interactive Low Latency Applications over 5G. *2023 IEEE International Conference on Multimedia and Expo (ICME)*, 1002–1007, 2023. doi:10.1109/ICME55011.2023.00176. ISSN: 1945-788X.

[73] S. Sundberg, A. Brunstrom, S. Ferlin-Reiter, T. Høiland-Jørgensen, R. Chacón. Measuring Network Latency from a Wireless ISP: Variations Within and Across Subnets. *Proceedings of the 2024 ACM on Internet Measurement Conference*, IMC '24, 29–43. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400705922. doi:10.1145/3646547.3688438.

[74] S. R. System. srsRAN Project: Open Source RAN, 2023.

[75] A. Team. Testing and Debugging L4S in Your App, 2023.

[76] H. Wan, X. Cao, A. Marder, K. Jamieson. NR-Scope: A Practical 5G Standalone Telemetry Tool. *Proceedings of the 20th International Conference on emerging Networking EXperiments and Technologies*,

CoNEXT ’24, 73–80. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400711084. doi:10.1145/3680121.3697808.

[77] C. Wang, H. Wang, Y. Zhou, Y. Ni, F. Qian, C. Xu. {SMUFF}: Towards Line Rate {Wi-Fi} Direct Transport with Orchestrated On-device Buffer Management. *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 1369–1383, 2024. ISBN 978-1-939133-39-7.

[78] S. Wang, K. Guan, D. He, G. Li, X. Lin, B. Ai, Z. Zhong. Doppler Shift and Coherence Time of 5G Vehicular Channels at 3.5 GHz. *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, 2005–2006, 2018. doi:10.1109/APUSNCURSINRSM.2018.8608340. ISSN: 1947-1491.

[79] G. White, T. Fossati. A Non-Queue-Building Per-Hop Behavior (NQB PHB) for Differentiated Services. Internet Draft draft-ietf-tsvwg-nqb-19, Internet Engineering Task Force, 2025. Num Pages: 29.

[80] K. Winstein, A. Sivaraman, H. Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[81] X. Xie, X. Zhang, S. Kumar, L. E. Li. piStream: Physical Layer Informed Adaptive Video Streaming over LTE. *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom ’15, 413–425. Association for Computing Machinery, New York, NY, USA, 2015. ISBN 978-1-4503-3619-2. doi:10.1145/2789168.2790118.

[82] X. Xie, X. Zhang, S. Zhu. Accelerating Mobile Web Loading Using Cellular Link Information. *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys ’17, 427–439. Association for Computing Machinery, New York, NY, USA, 2017. ISBN 978-1-4503-4928-4. doi:10.1145/3081333.3081367.

[83] Y. Xie, F. Yi, K. Jamieson. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements. *Proc. of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM, 451–464. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 978-1-4503-7955-7. doi:10.1145/3387514.3405880.

[84] W. Ye, X. Hu, S. Sleder, A. Zhang, U. K. Dayalan, A. Hassan, R. A. K. Fezeu, A. Jajoo, M. Lee, E. Ramadan, F. Qian, Z.-L. Zhang. Dissecting Carrier Aggregation in 5G Networks: Measurement, QoE Implications and Prediction. *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM ’24, 340–357. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400706141. doi:10.1145/3651890.3672250.

[85] F. Yi, H. Wan, K. Jamieson, O. Michel. Automated, Cross-Layer Root Cause Analysis of 5G Video-Conferencing Quality Degradation, 2025.

[86] F. Yi, H. Wan, K. Jamieson, J. Rexford, Y. Xie, O. Michel. Athena: Seeing and Mitigating Wireless Impact on Video Conferencing and Beyond. *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, HotNets ’24, 103–110. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400712722. doi:10.1145/3696348.3696889.

[87] G. Yuan, M. Sotoudeh, D. K. Zhang, M. Welzl, D. Mazières, K. Winstein. Sidekick: {In-Network} Assistance for Secure {End-to-End} Transport Protocols. 1813–1830, 2024. ISBN 978-1-939133-39-7.

[88] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, C. Görg. Adaptive Congestion Control for Unpredictable Cellular Networks. *Proc. of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM, 2015.

[89] D. Zeynali, E. N. Weyulu, S. Fathalli, B. Chandrasekaran, A. Feldmann. BBRv3 in the public Internet: a boon or a bane? *Proceedings of the 2024 Applied Networking Research Workshop*, ANRW ’24, 97–99. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400707230. doi:10.1145/3673422.3674889.

A L4Span Layer Pseudo Code

Here, we list the pseudo code of L4Span triggered by three events: on receiving a downlink IP packet (Fig. **22**), on receiving the RAN feedback (Fig. **23**, top) and on receiving an uplink packet (Fig. **23**, down).

```
void on_dl_pkt(buffer pkt, qos_flow_id qfi) {
  /* Map the QFI to DRB */
  auto drb_id = qfi_to_drb(qfi);
  /* Save the five-tuple mapping to DRB */
  auto five_tuple = extract_five_tuple(pkt);
  five_tuple_to_drb[five_tuple] = drb_id;
  /* Read the ECN bits and update DRB flows */
  auto flow_type = classify_flow(pkt);
  update_drb_flows(drb_id, flow_type);
  /* Save to packet profile queue */
  add_to_profile_queue(drb_id, pkt);
  /* Perform marking if it's a UDP packet */
  if (is_udp(pkt))
    mark_pkt(drb_id, pkt);
  /* Pass the packet to lower SDAP layer */
  to_sdap(pkt, qfi);
}
```

Fig. 22— The L4Span layer's processing upon receiving a downlink IP packet.

```
void on_ran_feedback(uint32_t txed_sn, uint32_t
     dlvred_sn, drb_id_t drb_id, timestamp ts) {
  /* Update the pkt profile queue */
  update_pkt_prof(txed_sn, dlvred_sn, drb_id, ts);
  /* Standing packets queuing delay prediction */
  auto stand_pkt_qdelay = qdelay_predict();
  /* Update marking decision*/
  update_drb_mark_state(drb_id, stand_pkt_qdelay);
}
```

```
void on_ul_packet(buffer pkt) {
  if(is_tcp_ack(pkt)) {
    /* Extracts the ACK's downlink five-tuple */
    auto five_tuple = extract_ack_five_tuple(ack);
    /* Reverse map the five-tuple to DRB id */
    auto drb_id = five_tuple_to_drb[five_tuple];
    /* Mark ECN bits based on the mark state */
    mark_pkt(drb_id, pkt);
  }
  to_upf(pkt); /* Pass the packet to the core UPF */
}
```

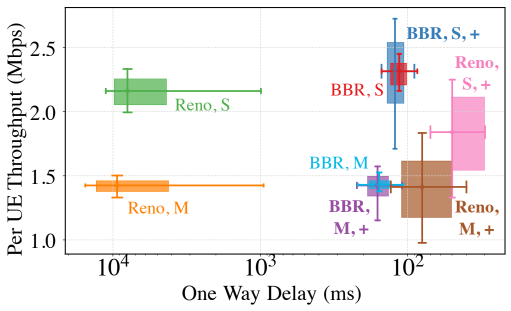
Fig. 23— The L4Span's processing on receiving the RAN feedback (top), and the uplink packet (down).

B BBR and Reno's Evaluation Result with L4Span

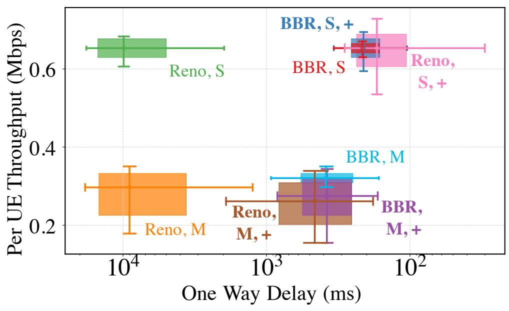
Here we present the evaluation result of BBR and Reno:

- BBR [21]. BBR sender periodically switches between bottleneck bandwidth and propagation delay probing state, and doesn't react much to the packet loss or CE feedback.
- Reno [64]. TCP Reno cuts its congestion window to half upon packet loss and adds one onto its congestion window size per RTT in steady state.

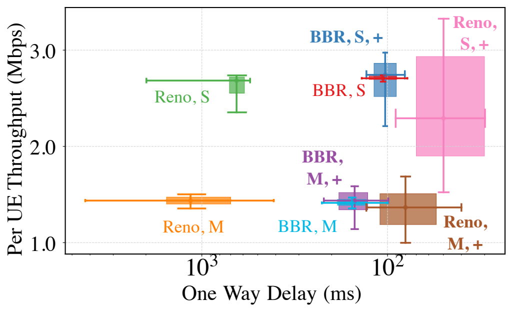
L4Span can decrease the RTT of Reno by 97.61% and 97.12% in static and mobile channels, yielding a higher variation in throughput. For other channels and different UE numbers, L4Span can constantly improve the performance of RTT by more than 90% and incurs very little performance drop in throughput. BBR, doesn't react much to the ECN or packet loss, yields higher variations in RTT and throughput with L4S, while the median values stay the same most of the time.



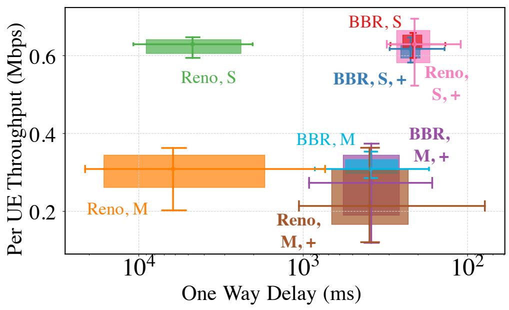
(a) 16 UEs, default RLC queue length, 38 ms RTT.



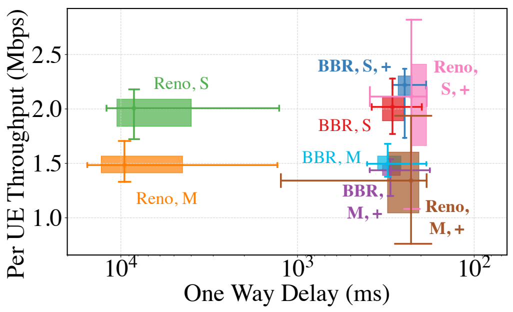
(b) 64 UEs, default RLC queue length, 38 ms RTT.



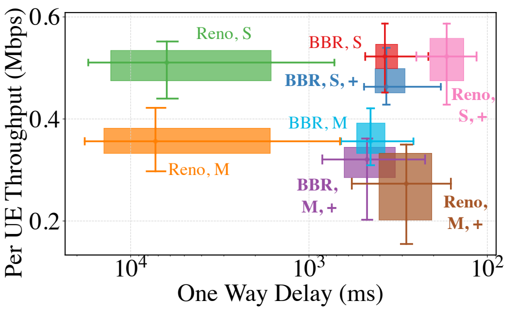
(c) 16 UEs, RLC queue length 256, 38 ms RTT.



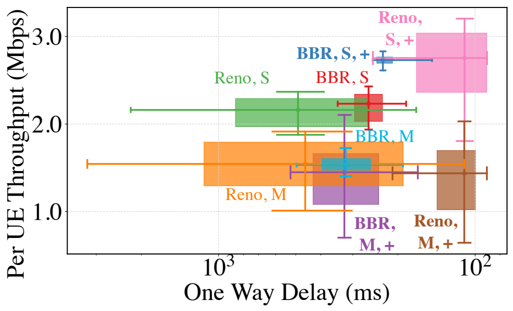
(d) 64 UEs, RLC queue length 256, 38 ms RTT.



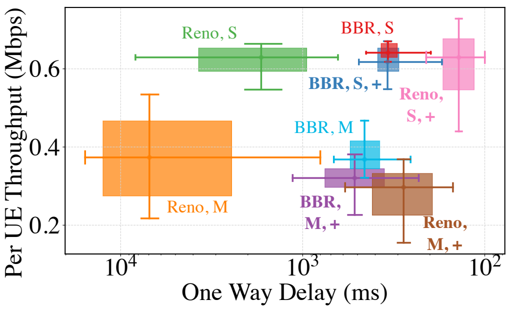
(e) 16 UEs, default RLC queue length, 106 ms RTT.



(f) 64 UEs, default RLC queue length, 106 ms RTT.



(g) 16 UEs, RLC queue length 256, 106 ms RTT.



(h) 64 UEs, RLC queue length 256, 106 ms RTT.

Fig. 24— L4Span improves the performance of various congestion control algorithms in terms of reducing RTT while maintaining throughput, under severely congested RAN and different channel conditions (S: Static, M: Mobile). **Bold** font and adding symbol (+) indicate L4Span is deployed. Senders are Azure instances with uncongested RAN ping time marked in the caption. Center point: median, box edges: 25th- and 75th-percentile, and whiskers: 10th- and 90th-percentile.