César: Cellular Resource Scheduling-Aware Congestion Control

Juhun Shin*, Goodsol Lee*, Jeongyeup Paek†, Saewoong Bahk*

*Department of Electrical and Computer Engineering and INMC, Seoul National University, Seoul, Republic of Korea
†Department of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea
jhshin@netlab.snu.ac.kr, gslee2@netlab.snu.ac.kr, jpaek@cau.ac.kr, sbahk@snu.ac.kr

***Abstract*—*Delay-based congestion control algorithms* (CCAs) have been proposed to tackle the *bufferbloat* problem of traditional *loss-based CCAs*. However, existing delay-based CCAs either fail to adequately consider the non-congestive delay caused by scheduling characteristics of modern cellular networks leading to improper congestion control, or face practical deployment issues, which results in an inability to fully utilize the high bandwidth and low latency that recent cellular systems provide. To resolve this problem, we propose *César*, a cellular resource scheduling-aware congestion control with only sender-side modification. *César* estimates *scheduling unit* through TCP ACK interval patterns to deduce the scheduling characteristics of the current cellular link, and adjusts the congestion window size based on *scheduling unit* in a *step-wise* manner to minimize the impact of the *scheduling delay* on congestion control. Experimental results on 5G and LTE cellular networks of three different mobile carriers show that *César* outperforms other state-of-the-art CCAs. Results show that throughput-over-latency performance improves by up to 2.89×, 10.09×, 1.39×, and 5.65× compared to ExLL, PropRate, BBR, and Cubic, respectively.**

***Index Terms*—Congestion control, cellular, transport protocol**

I. Introduction

Cellular communication has been advancing dramatically to simultaneously satisfy the high bandwidth and low latency requirements of emerging real-time mobile applications such as VR/AR, 360-degree video, and cloud gaming [1–3]. Despite these advancements, however, *loss-based* congestion control algorithms (CCAs) such as Cubic [4] fail to ensure low latency due to the *bufferbloat* [5] problem; pushing packets until the deep buffers at the cellular base station (BS) [6] are completely filled and packets are lost, resulting in long delays [7, 8]. To overcome this problem, *delay-based* CCAs that use delay information (e.g., round trip time (RTT) [9], jitter [10]) instead of packet loss for congestion control have been proposed.

Delay-based CCAs [9, 11] can simultaneously achieve high bandwidth and low latency in an environment free from non-congestive delays (e.g., wired network). However, cellular networks inevitably have time-varying non-congestive *scheduling delays* due to their wireless resource allocation and scheduling mechanisms. These *scheduling delays* result in contamination of RTT information, misleading delay-based CCAs to adjust the congestion window size (`cwnd`) in an unintended and inappropriate manner such as reducing the `cwnd` even though there is no congestion. As a result, delay-based CCAs fail to simultaneously achieve high link utilization and low latency over cellular networks, even in relatively stable wireless conditions. To observe this, we compare the performances of various CCAs on a commercial 5G network (Fig. 1). While Copa [9] and Vegas [12] achieve low latency, they exhibit significantly lower bandwidth utilization compared to other CCAs. BBR also fails to fully utilize the bandwidth. High bandwidth utilization is achieved only when the `cwnd` is manually set and fixed close to the BDP.

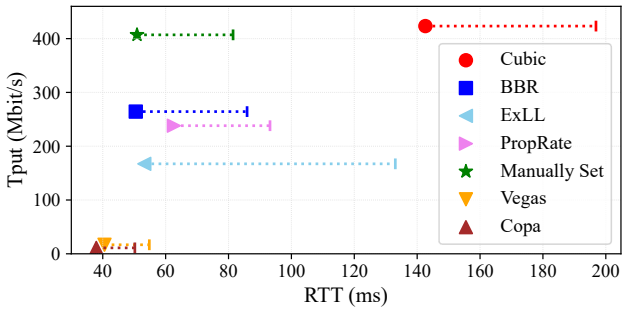


Fig. 1: Average throughput, average RTT, and 95th percentile RTT of different CCAs over a commercial 5G network. 'Manually Set' CCA manually sets the `cwnd` around the known BDP.

In response to the limitations of delay-based CCAs in cellular networks, researchers have proposed various end-to-end and in-network CCAs. However, as shown in Fig. 1, even the end-to-end CCAs specifically designed for low latency on cellular networks (ExLL [13], PropRate [14]) fail to achieve high bandwidth utilization. This is because end-to-end CCAs [13–16] do not sufficiently account for the scheduling characteristics of cellular networks. Furthermore, cross-layer CCAs [17, 18] or in-network CCAs [19] encounter practical deployment issues or require additional deployment costs such as the need to modify the cellular BS or devices of all users.

To address this problem, we propose *César*, an end-to-end *cellular resource scheduling-aware congestion control* scheme with only sender-side modification. Unlike other scheduling-agnostic CCAs, *César* is designed specifically with the scheduling characteristics of cellular links in mind to minimize the impact of *scheduling delays* on `cwnd` adjust-

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. IITP-2025-RS-2024-00398157 & No. IITP-2025-RS-2024-00405128 & No. IITP-2025-RS-2024-00418784).

ment. *César* defines *scheduling unit* as the minimum time difference between the *sets* of uplink and downlink resources. Based on the fact that *scheduling unit* correlates with the intervals between TCP ACKs at the sender (end-host), *César* estimates *scheduling unit* at the end host, bypassing the need for low-layer information. Then, *César* utilizes the estimated *scheduling unit* and the understanding of cellular scheduling to adjust `cwnd` in a *step-wise* manner. The *step-wise* approach is designed to minimize the impact of *scheduling delays*, which fluctuate over time, on `cwnd` control.

We implement *César* in the Linux Kernel on an Amazon Web Service (AWS) cloud server (only sender-side modification), and conduct experiments on commercial cellular networks (for both LTE and 5G) for three different mobile carriers. Results demonstrate that *César* outperforms other delay-based CCAs in achieving both high throughput and low latency across all mobile carriers and networks. *César* achieves up to 2.89×, 10.09×, 1.39×, and 5.65× throughput-over-latency performance compared to ExLL, PropRate, BBR, and Cubic, respectively.

The contributions of this work are as follows:

- We investigate how cellular scheduling impacts the delay-based CCAs on multiple commercial networks and analyze the underlying causes.
- We design *César* to fully utilize the high bandwidth and low latency over cellular using novel *scheduling unit* and *step-wise* `cwnd` adjustment with only sender-side modification.
- We implement *César* and evaluate it on commercial cellular networks to show a remarkable improvement in bandwidth utilization and low latency performance.

II. Background

We first provide a brief background on cellular link scheduling, network delay components, and delay-based CCA.

A. Cellular Link Scheduling

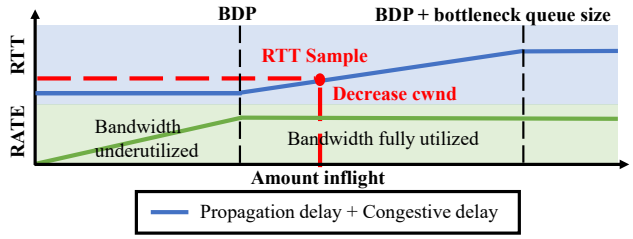
Cellular systems allocate wireless resources to user equipment (UE) based on slot units (typically 0.5 ms) [20]. For downlink, the BS schedules resources for multiple UEs every *transmission time interval* (typically 1 ms) based on the signal strength reported by each UE. For uplink, a UE sends a *scheduling request* (SR) to the BS to obtain the uplink scheduling grant. The BS then allocates uplink resources based on the SR. In LTE systems, SR periodicity, which is the period for the UE to send SRs, is typically selected from 5, 10, 20, 40, or 80 ms [21]. 5G system can use various slot lengths (e.g., 0.25 ms, 0.125 ms) and offer faster SR periodicity for shorter slot lengths [22]. Based on these basic mechanisms, the BS schedules wireless resources to optimize the performance in terms of spectral efficiency and fairness [23].

B. Network Delay Components

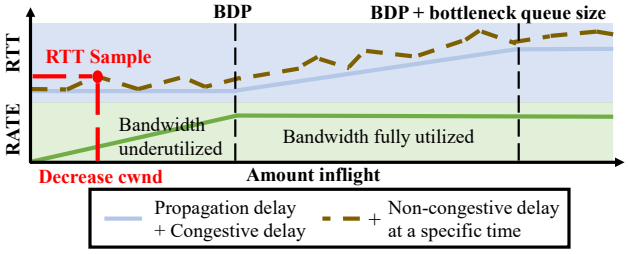
Delay (or RTT) of a packet consists of three components: propagation delay, congestive delay, and non-congestive delay [24][1]. Based on this classification, we tailor the definitions of each component for cellular networks.

- **Propagation delay** is the inevitable minimum delay at non-bottleneck links, which is the sum of *transmission delays* across non-bottleneck links and the *speed-of-signal delay* from the sender to the receiver along the end-to-end path.
- **Congestive delay** is the sum of the time a packet spends in the bottleneck queue when wireless resources are available and the transmission time over the bottleneck link.
- **Non-congestive delay** is the additional delay caused by network elements irrelevant to congestion (i.e. neither bandwidth nor queue). For example, this includes the time a packet is held due to cellular link scheduling, processing at the BS, delayed ACKs, etc.

The delay occurring when a packet is in the bottleneck link's queue without available wireless resources should not be interpreted as congestive delay because this delay is merely caused by not receiving resources due to cellular scheduling mechanism [25]. We refer to this non-congestive delay as *scheduling delay*.



(a) Non-congestive delay does not exist



(b) Time-varying non-congestive (scheduling) delay exists

Fig. 2: Delivery rate and RTT for the amount of inflight when (a) non-congestive delay does not exist, (b) time-varying non-congestive (scheduling) delay exists. Delay-based CCA in this example decreases the `cwnd` when RTT exceeds the minimum RTT.

C. Delay-based Congestion Control

Delay-based CCAs aim to adjust the `cwnd` to the optimal bandwidth-delay product (BDP) point where RTT is minimized and delivery rate is maximized using delay information (typically RTT increase) instead of loss [9, 12–16]. Most delay-based CCAs rely on the fact that RTT remains equal to the propagation delay (minimum RTT) and does not increase until the bandwidth is fully utilized as in Fig. 2a. Once the `cwnd` increases beyond fully utilizing the bandwidth, the queue builds up, causing an RTT increase. Delay-based CCAs use this RTT increase to estimate the congestive delay. For example, as depicted in Fig. 2a, delay-based CCAs decrease the `cwnd` when an RTT sample exceeds the minimum RTT,

[1]Definition and categorization of delay components may vary depending on the perspective and purpose, and we refer to [24].

interpreting this as congestive delay and the current `cwnd` being larger than the BDP.

Additionally, TCP congestion control can use *pacing* [26] to improve performance. Even in networks with the same BDP, bursty traffic can cause more frequent packet losses and higher queuing delays. To mitigate this problem, the pacing smoothens TCP traffic by evenly spacing (adding intentional delays to) bursty data transmissions.

III. Motivation and Related work

We explain why TCP should care about link layer *scheduling delays*, and show that existing CCAs even for cellular networks do not adequately consider *scheduling delays*.

A. Why should TCP consider the scheduling delay?

Scheduling delay is inevitable in today's cellular systems. Current cellular technology focuses on spectral efficiency to optimize link throughput with limited wireless resources [23]. To maximize spectral efficiency, the BS allocates resources to UEs with relatively good channel conditions. This means that a UE may not receive downlink resources consistently, but rather in bursts due to fluctuating channel quality[2]. Therefore, downlink packets for a UE may not be sent immediately upon arrival at the BS, but instead experience *scheduling delay*, unrelated to congestion, while waiting for downlink resources. This downlink *scheduling delay* is not constant but varies over time as the channel condition of UE and the status of BS changes. Furthermore, after receiving a downlink TCP packet, the UE cannot send the TCP ACK immediately but must wait for the BS to grant the uplink schedule. While waiting for the uplink schedule, the UE accumulates TCP ACKs and sends them in a batch when the uplink resource is granted, resulting in delays and bursty traffic.

Fig. 3 plots the RTT samples at the sender (end-host) when `cwnd` is fixed to be sufficiently below the known BDP. Uplink scheduling generates a discrete *line-like pattern* because TCP ACKs for packets that left the downlink queue earlier wait for uplink resources on the UE side, aligning their processing with TCP ACKs for later-departing downlink packets. Thus, the packets for which TCP ACKs are sent using the same uplink resource on the UE side appear to have the same RTT at the sender (end-host) due to the alignment at the uplink, resulting in a discrete *line-like RTT pattern* instead of random-looking continuous values. Furthermore, downlink scheduling results in downlink packets waiting at the BS until downlink resources are allocated to the UE regardless of congestion. Thus, even when `cwnd` is sufficiently below the BDP, RTT samples higher than the minimum are observed. Consequently, the combination of uplink scheduling, which discretizes RTT, and downlink scheduling, which increases RTT beyond the minimum RTT, leads to a multiple *line-like RTT pattern*.

[2]This occurs in short-term scale. The BS considers fairness as well as spectral efficiency, thus even the UEs with relatively poor channel conditions will receive resources in long-term scale.

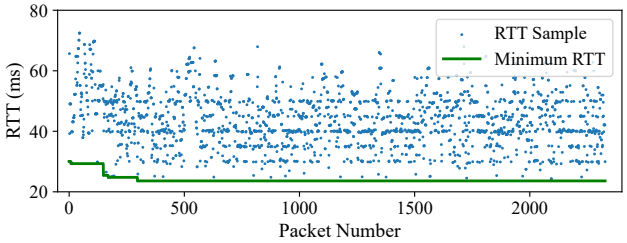


Fig. 3: RTT samples over a commercial cellular network when `cwnd` is fixed to utilize approximately 100 Mbps out of 550 Mbps. `cwnd` is fixed to reduce the effects of congestion control and better observe the RTT characteristics of the cellular link.

These time-varying *scheduling delays* cause the RTT to exceed the minimum even when there is no congestion. Delay-based CCAs misinterpret these RTT increases as congestive delays, and improperly reduce `cwnd` even when there is no queueing and the current `cwnd` is smaller than the BDP, thus resulting in bandwidth under-utilization. For example in Fig. 2b, delay-based CCAs that are agnostic to *scheduling delays* observe the RTT sample that exceeds the minimum RTT and determine that the current `cwnd` is larger than the BDP as in Fig. 2a. This leads to the decision to decrease `cwnd`, resulting in an even lower bandwidth under-utilization. Additionally, if temporarily increased *scheduling delay* decreases due to its time-varying nature, delay-based CCA can misinterpret this as improved network conditions, leading to an improper increase in `cwnd` to probe the maximum bandwidth. This increase results in queuing and subsequent increases in latency.

B. Limitation of Existing Works

Various CCAs have been proposed to improve TCP performance over cellular, but they fail to adequately consider the scheduling or face practical deployment issues, preventing UEs from fully achieving high bandwidth and low latency.

Some delay-based CCAs, such as Verus [15], C2TCP [16], Copa [9], and BBR [11] use the max/min RTT or estimated bandwidth *within a certain window* for congestion control. However, this window-based approach cannot accurately reflect the status of cellular links. Due to the time-varying nature of *scheduling delay*, the max/min RTT within a window changes over time, even in the absence of congestion. Estimated bandwidth within a window varies depending on how many times bursty scheduling occurs within that window, even if the bottleneck bandwidth remains the same [25]. Additionally, BS parameters and scheduling methods can vary not only between mobile carriers, but also by region and between LTE and 5G even within the same mobile carrier. This leads to diverse *scheduling delay* patterns, making it difficult to determine a window size that accommodates all these variations.

There are CCAs that use delay information directly to infer current congestion status. However, these methods are effective only if the *scheduling delay* is constant and an increase in RTT indicates queuing. Verus [15] uses the relationship between RTT and `cwnd`, but fails to accurately

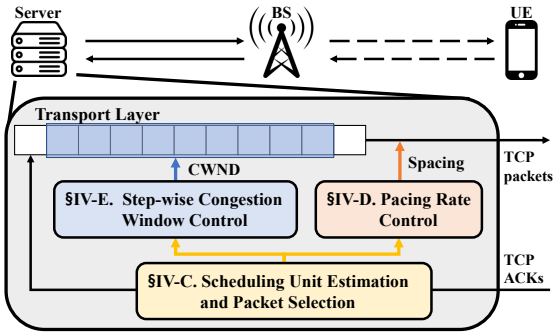


Fig. 4: Overview of *César* design.

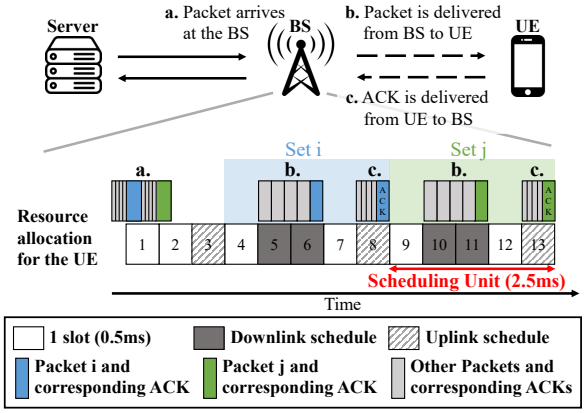


Fig. 5: An example of resource allocation for one UE and end-to-end TCP data/ACK packet delivery process in cellular networks.

reflect network conditions as *scheduling delay* varies over time. ExLL [13] considers the uplink SR period but neglects the burst characteristics of downlink, leading to an underestimation of *scheduling delay*. PropRate [14] and Copa [9] adjust `cwnd` based on whether RTT exceeds a certain threshold, but RTT can surpass the threshold regardless of actual network conditions due to time-varying *scheduling delays*.

There are CCAs that adjust `cwnd` based on lower- or higher-layer information [17, 18, 27, 28]. For example, CLAW [17] utilizes physical layer information (e.g., RSSI, RSRP), whereas PBE-CC [18] estimates available bandwidth by analyzing control messages. This information is delivered to the sender for `cwnd` adjustment. There are also CCAs (e.g., ABC [19], XCP [29]) that use explicit congestion notification (ECN) to provide improved performance compared to end-to-end methods. However, employing such schemes for cellular networks requires modifications to the BS, sender, and UE sides. Modifying the BS and adapting the sender code accordingly, across all mobile carriers, makes large-scale deployment impractical.

IV. César Design

We propose *César*, a cellular resource scheduling-aware CCA to achieve high throughput and low latency over cellular.

A. Overview

The overall structure of *César* is shown in Fig. 4. *César* estimates *scheduling unit* to represent the burst scheduling characteristics of the current cellular link through the TCP ACK interval pattern. If the ACK intervals do not appear to have a cellular-specific *line-like pattern* as in Fig. 3, *César* determines that the bottleneck is not at the cellular link and applies BBR [11] congestion control[3]. Using estimated *scheduling unit*, *César* selects the packet that is least affected by scheduling (§IV-C), and adjusts the `cwnd` in a *step-wise* manner to minimize the impact of *scheduling delays* (§IV-E). Furthermore, *César* controls the pacing rate to smooth out the traffic (§IV-D).

B. 'Scheduling Unit' Definition

Scheduling unit represents the bursty scheduling characteristic, and is defined as follows: the minimum end-time difference between the *sets* of uplink and downlink resources.

From an end-to-end perspective, downlink and uplink resources can be considered to be processed in batches, or in *sets*, due to the aligning characteristics of the uplink mechanism. For instance, in Fig. 5, all packets allocated to the downlink resources in slots 5 and 6 share the same uplink resource in slot 8. These packets can be considered as one *set* because, from the perspective of the sender (end-host), it appears as if the TCP ACKs of *set* i are processed as a single batch and arrive all at once. In the same manner, the packets in *set* j also share the same *set* of downlink and uplink resources.

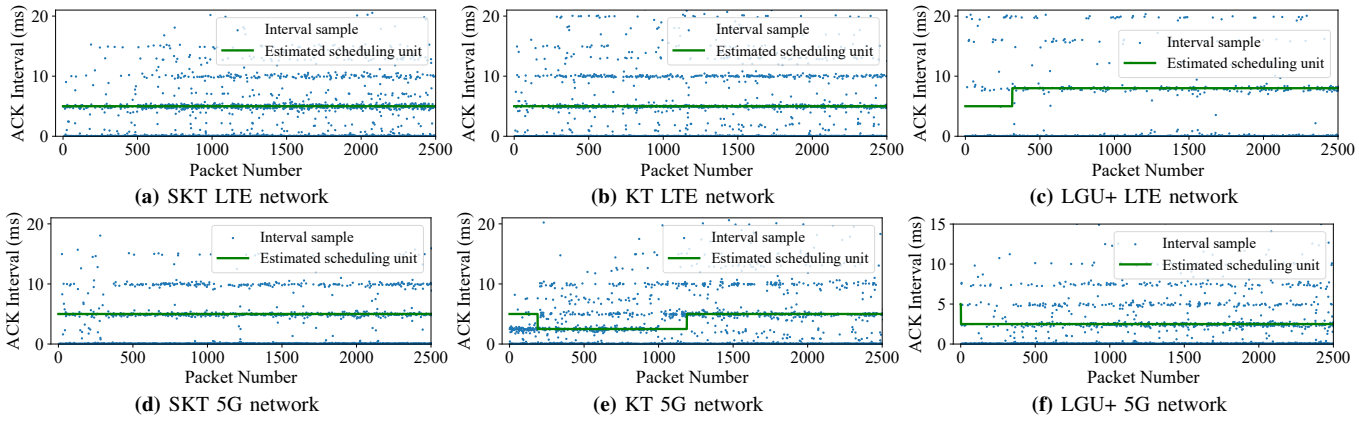
The end-time difference between two consecutive *sets* displays a bursty scheduling pattern. If a BS allocates downlink resources to a UE in a bursty manner every 10 ms, the end-time difference will be 10 ms. The bursty scheduling pattern is determined by which of the various BS parameters has a significant impact, such as uplink SR period, uplink-downlink switch time in case of time division duplexing (TDD), and vendor-specific scheduling rules [25]. In Fig. 5, uplink SR period (2.5 ms) has a significant impact, so bursty pattern (a set) occurs every 2.5 ms. But, if the BS employs TDD and the uplink-downlink switch time is 5 ms, scheduling follows a pattern of 5 ms. Or if the vendor has configured the BS to avoid allocating resources to the recently allocated UE for the next 10 ms for fairness, bursty scheduling will occur every 10 ms. The crucial point is that, regardless of the cause, the inherent nature of cellular systems inevitably leads to bursty scheduling patterns.

The end-time differences between *sets* can vary. For example, downlink resources may not be allocated depending on UE channel conditions or BS status as previously mentioned in §III-A. In Fig. 5, if the downlink resources in slots 5 and 6 are not assigned due to poor UE channel conditions, the time difference will be doubled, skipping one *set*. Therefore, we estimate the minimum time difference to deduce the bursty scheduling characteristic.

C. Scheduling Unit Estimation and Packet Selection

We identified that *scheduling unit* correlates with the intervals between consecutive TCP ACKs at the sender. The sender

[3]We choose BBR because it is known to generally have good performance in wired networks [30].



(a) SKT LTE network **(b)** KT LTE network **(c)** LGU+ LTE network

(d) SKT 5G network **(e)** KT 5G network **(f)** LGU+ 5G network

Fig. 6: Interval samples (blue dots) between consecutive TCP ACKs and estimated *scheduling unit* (green line) at the sender (end-host) over commercial LTE and 5G networks provided by three mobile carriers.

receives ACKs in batches because packets are usually processed by the BS in *sets* in a bursty manner. Therefore, the end-time difference between *sets* can be estimated by comparing the arrival time intervals between consecutive ACKs. Fig. 6 are example plots of these measured intervals. Many intervals are distributed near 0 ms due to bursty ACKs. As shown in Fig. 6a and Fig. 6d, if the BS has a 5 ms scheduling pattern (due to 5 ms SR period), the interval patterns are multiples of 5 ms. Intervals outside 5 ms patterns occur possibly due to changes in propagation delays, delays in non-bottleneck links, or the UE/BS processing, which are relatively less dominant. By gathering ACK interval samples, *César* estimates *scheduling unit* as the smallest least common multiple of the ACK interval pattern that frequently appears, not including those near 0 ms. If the ACK pattern does not appear in bursts (not multiple lines), *César* determines that the bottleneck is not the cellular link. In this way, *César* can identify the bottleneck without additional information. *Scheduling unit* may vary over time due to vendor's scheduling rules (e.g., dependent on number of users). Therefore, *César* measures ACK intervals and estimates *scheduling unit* continuously. Fig. 6e shows that *César* effectively adapts to the changing pattern. Further details will be discussed in the evaluation section (§V-B).

Based on the estimated *scheduling unit*, *César* can determine when each *set* starts and ends. If the ACK interval exceeds *scheduling unit* or is almost the same, it indicates that all bursty ACKs from the current *set* have been received, and ACKs from the next *set* have started to arrive. On the other hand, if the ACK interval is near 0 ms, it implies that ACKs from the current *set* are yet to be received. For example, in Fig. 7 where *scheduling unit* is 5 ms, we infer that the next *set* has begun at the point where the ACK timestamp increases by 5 ms. To distinguish *sets* more accurately, *César* determines that a *set* ends if more than *scheduling unit* time has elapsed since the arrival of the first ACK in a group of bursty ACKs.

César selects the last packet in each *set* as the *anchor packet* for congestion and pacing rate control because it is the least affected by scheduling. RTT may include waiting time for uplink scheduling which contaminates the RTT from

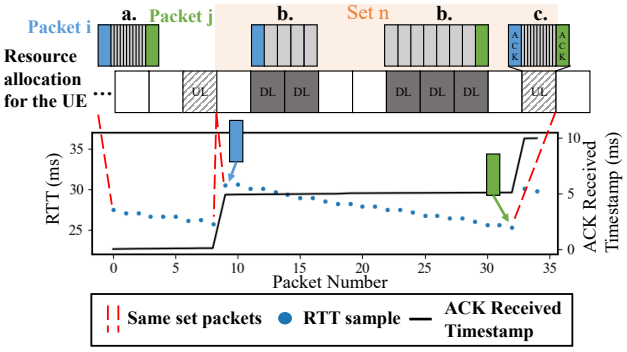


Fig. 7: Collected RTT samples and received timestamps at the server using a commercial cellular network, along with an example of their resource allocation (same network topology and legend as Fig. 5). Packet i and j share the same *set* n, and SR period is 5 ms.

a congestion control perspective. In this situation, the last packet of each *set* experiences the shortest wait time for uplink among the packets in each *set*. In Fig. 7, packet i is the first packet in *set n*, and packet j is the last which is the *anchor packet* of *set n*. Packet i is served in the first downlink and waits the longest for uplink, whereas packet j is served in the final downlink slot and waits the shortest time for uplink. Also, the departure times of TCP packets from the sender (end-host) are delayed because *César* uses pacing. However, the arrival times of ACKs are aligned by the uplink. Thus, listing the RTTs of packets in *set n* shows a gradually decreasing trend as shown in Fig. 7. This decreasing trend occurs purely due to scheduling, and does not indicate an improvement in network congestion. Thus, the *anchor packet* prevents misinterpretation of temporary RTT decreases caused by scheduling as improvements in network conditions.

D. Pacing Rate Control

Pacing can be used to improve performance in cellular networks. If the sender transmits packets in a burst that exceeds the capacity of a *set*, it will cause large delays as the packets wait for the next *set*. By using pacing appropriately, packets can arrive at the BS in accordance with the downlink resources, reducing *scheduling delays*.

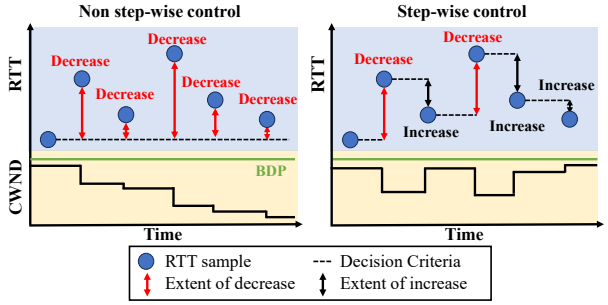


Fig. 8: Difference between *step-wise* and *non step-wise* control when RTT increase is purely caused by *scheduling delay*.

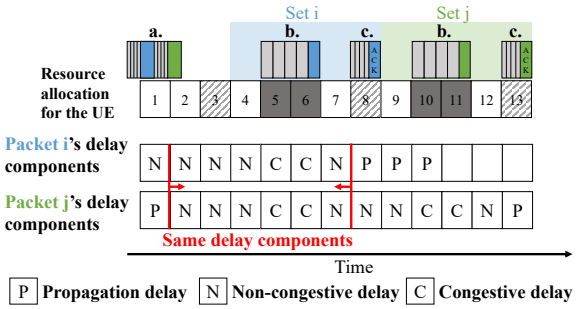


Fig. 9: An example of resource allocation for one UE and delay components according to §II-B (same network topology and legend as Fig. 5). The RTT of packet j is larger than that of packet i.

César estimates bandwidth based on *anchor packets*. The estimated bandwidth for packet i (BW_i) is calculated as,

$$BW_i = \frac{Delivered_i}{RTT_i}. \tag{1}$$

where RTT_i denotes RTT of packet i, and $Delivered_i$ is the total number of packets delivered during RTT_i. To smooth the short-term fluctuation, *César* utilizes the exponentially weighted moving average (ewma) of BW_i to estimate cellular bandwidth as follows,

$$BW_i^{ewma} = (1-\gamma) \times BW_{i-1}^{ewma} + \gamma \times BW_i. \tag{2}$$

where BW_i^{ewma} is the ewma bandwidth and $0 < \gamma < 1$.

However, using this ewma bandwidth directly as the pacing rate is insufficient because we must not only operate effectively according to the current bandwidth but also harness the available high bandwidth. Therefore, *César* sets the pacing rate to be greater than the ewma bandwidth while considering RTT, as shown in the following equations.

$$Pacing_i = \alpha \times \rho_i \times BW_i^{ewma}. \tag{3}$$

$$\rho_i = 1 + \frac{1-\alpha}{\alpha} \times \frac{RTT_i - RTT_{min}}{RTT_i}. \tag{4}$$

$Pacing_i$ denotes the calculated pacing rate using packet i, and the unit is packets per time (e.g., packet/ms). The spacing between packets is the inverse of the pacing rate. α is a parameter that determines how aggressively to increase the pacing rate to fully utilize the bandwidth ($\alpha > 1$), and RTT_{min} denotes minimum RTT. ρ_i is a penalty factor against RTT increase. When the RTT of packet i gets closer to RTT_{min}, ρ_i approaches its maximum of 1. Thus, as the pacing rate increases, packets are sent with shorter intervals, increasing bandwidth utilization. On the other hand, as RTT increases, ρ_i approaches its minimum of $1/\alpha$, causing packets to be sent with longer intervals. This results in draining queue or responding to *scheduling delay* increase. The reason for using RTT value itself, which includes *scheduling delay*, is that the pacing rate can mitigate burstiness in packet transmission, thereby alleviating *scheduling delays*.

E. Step-wise Congestion Window Control

By *step-wise control*, we mean changing the decision criteria for adjusting `cwnd` (e.g., previous RTT) at every step (see Fig. 8). *Non step-wise control*, commonly used in existing CCAs, decides to decrease the `cwnd` when the RTT exceeds fixed criteria (e.g., RTT_{min}). This leads to improper congestion control as described in §III-A, failing to set the `cwnd` closer to the BDP. On the other hand, *step-wise control* utilizes the time-varying nature of *scheduling delay* by reducing `cwnd` when *scheduling delay* increases, aligning decision criteria with *scheduling delay*, and compensating `cwnd` when *scheduling delay* decreases. By changing the decision criteria in a step-wise manner, the impact of RTT fluctuations caused by scheduling can be minimized. However, this method is more susceptible to queuing delay compared to *non-step-wise* approach. *César* effectively utilizes the RTT value itself and bandwidth estimation to suppress the increase in queuing delay while leveraging the advantages of *step-wise control*.

The control equation of *César* is based on that of FAST [31]. *César* decreases `cwnd` when the current RTT exceeds the previous RTT (of *anchor packets*) as follows:

$$cwnd_i = (1-\beta) \times cwnd_{i-1} + \beta \times cwnd_{i-1} \times \left(\frac{RTT_{min}}{RTT_{min} + CD_i + BR_i} \right) \tag{5}$$

$$CD_i = (RTT_i - RTT_{i-1}) \times \frac{BW_i^{set}}{BW_i} \tag{6}$$

$$BR_i = (RTT_i - RTT_{min}) \times \frac{max(0,\ BW_i^{ewma} - BW_i)}{BW_i^{ewma}} \tag{7}$$

$cwnd_i$, CD_i, and BR_i denote `cwnd` when receiving packet i, inferred congestive delay, and penalty term for bandwidth reduction. BW_i^{set} is the estimated *set* bandwidth, calculated as the total number of delivered packets in *set* i divided by the duration of *set* i. β is a parameter that determines how sensitively `cwnd` responds to the RTT increase ($0 < \beta < 1$).

CD_i grows with larger RTT increases and larger BW_i^{set} compared to BW_i. BW_i^{set}/ BW_i term provides a hint about the cause of the RTT increase. As shown in Fig. 9, when there is an RTT increase between packets i and j, ACKs of packet i and j arrive at the sender through different *set*s. In this situation, both packets i and j are in the bottleneck queue after both packets arrive at the BS (slot 2) and share the same process until packet i receives its uplink resource (slot 8). So, they share the same delay components in slots 2-7 as shown in Fig. 9. Then, the cause of packet j's RTT increase compared to packet i is determined by the type of the delay component in

set j[4]. If BW_j^{set} is close to or larger than the recently estimated short-term bandwidth (BW_j), it indicates that the BS allocated sufficient downlink resources to the UE, resulting in the delay component in set j being predominantly congestive delay rather than *scheduling delay* as Fig. 9. Conversely, if BW_j^{set} is smaller than BW_j, it can be divided into two scenarios: the first scenario where the BS allocates fewer downlink resources to the UE, leading to *scheduling delay predominance*; and the second scenario where the bottleneck bandwidth decreases. In the first scenario, as the CD_j decreases, it results in a correct operation that reduces the impact on *scheduling delays*. To address the second scenario, *César* adds the penalty term for bandwidth reduction (*BR*). BR_j increases when BW_j decreases below the long-term estimated bandwidth BW_j^{ewma}, indicating a reduction in bottleneck bandwidth, and increases further as RTT_j becomes larger compared to RTT_{min} to quickly respond to bandwidth reduction.

If the current RTT is smaller than or equal to the previous RTT, *César* increases the `cwnd` using the following equation:

$$u_i = cwnd_{i-1} + BW_i^{ewma} \times SU \times \rho_i \times \left(\frac{RTT_{i-1} - RTT_i}{RTT_{i-1} - RTT_{min}} \right) \quad (8)$$

$$r_i = (1-\beta) \times u_i + \beta \times u_i \times \left(\frac{RTT_{min}}{RTT_{min} + BR_i} \right) \quad (9)$$

$$cwnd_i = max(cwnd_{i-1},\ r_i) \quad (10)$$

where u_i, r_i and *SU* denote increased `cwnd`, adjusted `cwnd` by bandwidth reduction, and *scheduling unit*, respectively. The final `cwnd` for packet i is $cwnd_i$. In Eq. (8), $BW_i^{ewma} \times SU$ means the number of packets that can be delivered during *scheduling unit* time with the current bandwidth. This term adjusts the amount of increase considering the current bandwidth, and if *scheduling unit* is large, the time between *sets* becomes longer, decreasing the frequency of `cwnd` increase decisions, so the amount of increase needs to be adjusted according to *scheduling unit* for high bandwidth utilization. ρ_i is used as a penalty against RTT increase. Also, if the current RTT has decreased significantly compared to the previous RTT, *César* increases `cwnd` more. In Eq. (9), *César* adjusts the increase amount to rapidly address the bandwidth reduction, which incorporates only the BR_i term from Eq. (5). Lastly, since *César* has decided to increase the `cwnd` when RTT decreases, Eq. (10) is used to prevent the final `cwnd` from decreasing due to the reduction amount specified in Eq. (9).

F. Fairness of César

César adopts a variant of FAST's control algorithm, thereby inheriting its fairness characteristics to a certain extent. Additionally, the cellular system guarantees a certain level of fairness in resource allocation. Also, since *César* adjusts the amount of `cwnd` increase and decrease based on bandwidth changes, *César* can quickly share bandwidth fairly. When a new flow joins the network and injects packets into the network, the existing flows detect a reduction in current bandwidth compared to the ewma bandwidth according to Eq. (7), causing the BR term to increase. The increase in the BR term raises the decrease amount of `cwnd` in Eq. (5) and reduces the increase amount in Eq. (9), causing existing flows to reduce their `cwnd`. On the other hand, the new flow experiences an increase in bandwidth, causing the BR term to become 0. Unlike existing flows, this leads to an increase in the `cwnd`, allowing the new flow to rapidly achieve its fair share. Also, *César* leverages the difference between the current and previous RTT, as well as the increase in RTT compared to RTT_{min}, allowing flows with significant propagation delays (large RTT_{min}) to fairly use their share.

V. Evaluation

We evaluate *César* by comparing it against state-of-the-art end-to-end CCAs on multiple commercial cellular networks.

A. Implementation and Experiment Setup

We implement *César* in 5.4.0 Linux kernel on an AWS cloud server[5], using a `t3.small` EC2 instance that offers up to 5 Gbps network bandwidth. To ensure that memory is not the bottleneck, the `tcp_wmem` and `tcp_rmem` parameters for a TCP socket are set sufficiently large [32]. Additionally, `tcp_no_metrics_save` is set to ensure that TCP parameters are not reused in repeated experiments. To measure maximum throughput, `Hystart` operation of Cubic is disabled. The cloud server sends downlink traffic to UEs using iperf3 [33] over commercial LTE or 5G networks provided by three mobile carriers (SKT, KT, LGU+). For UEs, we use two types of smartphones, Samsung Galaxy S22 and S23. We obtain RTT and received bytes data using the `pkts_acked` function of the `tcp_congestion_ops` on the server. We set *César*'s parameters to α=2, β=0.05 and γ=0.125[6].

B. Scheduling Unit of Different Carriers (LTE and 5G)

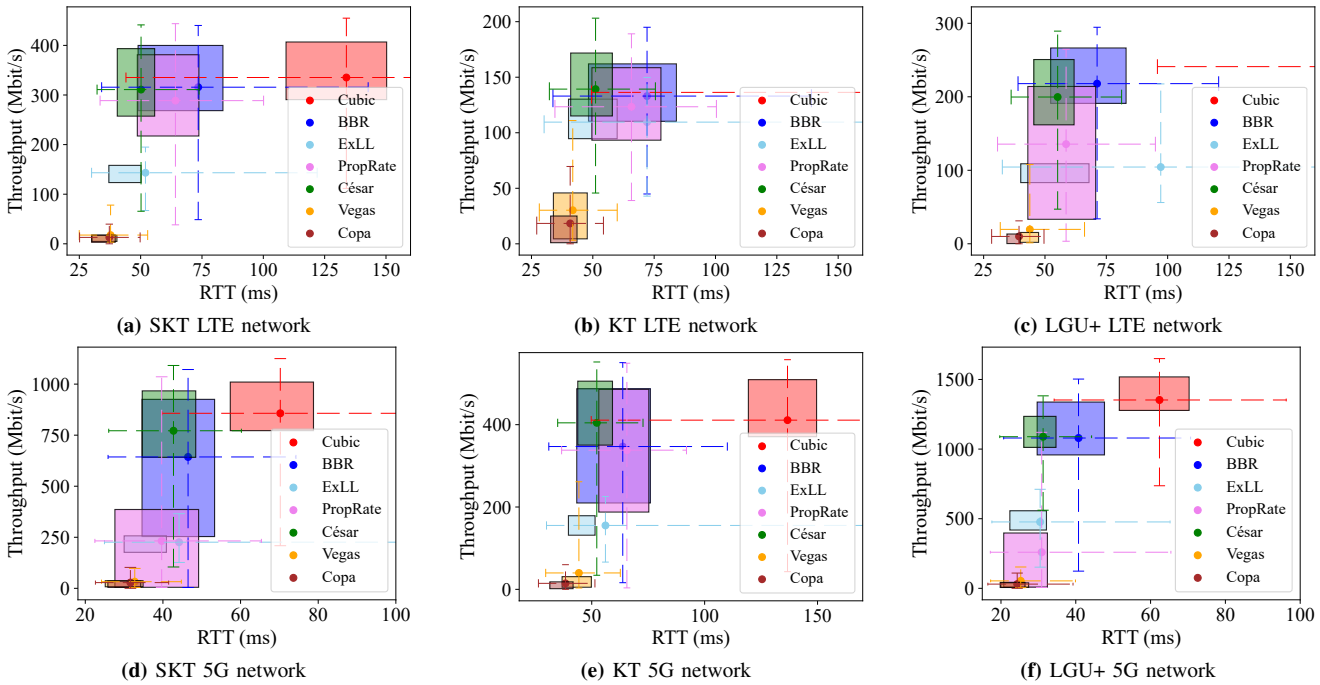
Each cellular network may have different *scheduling unit* depending on the technology, carrier, scheduling rules, and parameter settings for their BS. Unfortunately, cellular BS scheduling rules are classified information, preventing us from knowing the ground truth. Therefore, we utilize XCAL [34], a tool enabling analysis of control messages received at the UE, to infer BS parameters and verify the ACK interval pattern.

As mentioned in §IV-C, Fig. 6 plots the ACK intervals on LTE and 5G networks of three mobile carriers. For LTE, we verify that SKT and KT utilize a 5 ms SR period, and *César* correctly estimates *scheduling unit* as 5 ms. LGU+ employs a 20 ms SR period, visible as a line occurring at 20 ms in Fig. 6c. We deduce that LGU+ assigns uplink resources every 20 ms according to SR reports based on the specific scheduling rule, resulting in an 8 ms pattern. *César* accurately estimates

[4]The delay component between the arrival times at the BS of packets i and j (slot 1) are not considered because it is not a factor to increase the RTT of packet j compared to i.

[5]Code available at: https://github.com/Juhun1329/Cesar-TCP-CCA.git.

[6]As β increases, *César* prioritizes latency over throughput, and with higher α and γ, *César* becomes more sensitive to changes in bandwidth and RTT. 0.2, 0.05, and 0.125 are empirically selected for balancing high throughput and low latency simultaneously across all tested networks.



(a) SKT LTE network **(b)** KT LTE network **(c)** LGU+ LTE network

(d) SKT 5G network **(e)** KT 5G network **(f)** LGU+ 5G network

Fig. 10: Throughput and latency performance of different CCAs. The left and bottom edges of the box indicate the 25^{th} percentile for RTT and throughput, respectively, while the right and top edges indicate the 75^{th} percentiles. The two error bars represent the 5^{th} and 95^{th} percentiles, and the point where the two error bars overlap indicates the mean of the throughput and RTT. Upper-left is better performance.

scheduling unit as 8 ms. For 5G, all three mobile carriers set the BS parameters as 0.5 ms slot length, 5 ms SR period, and 2.5 ms uplink-downlink switch time. SKT shows a 5 ms pattern because the SR period has the dominant impact, while LGU+ shows a 2.5 ms pattern where the uplink-downlink switch time is dominant. We infer that KT's pattern changes from 2.5 ms to 5 ms depending on the variations of BS's scheduling rule (e.g., number of users), which in turn alters the influence of SR period and uplink-downlink switch time. *César* accurately estimates *scheduling unit* as 5 ms for both SKT and LGU+, and effectively adapts to the changing pattern for KT by accurately estimating it as both 2.5 ms and 5 ms.

C. Throughput and Latency Performance

We conduct 10-second TCP experiments 50 times for each CCA at different times of the day over 3 weeks, for each mobile carrier and network type (50×6=300 experiments for each CCA). Fig. 10 plots the throughput and latency of Cubic, BBR, ExLL, PropRate, *César*, Vegas, and Copa tested on commercial LTE and 5G networks across three different mobile carriers. To quantitatively assess their effectiveness in achieving high throughput and low latency simultaneously, we define an *efficiency* metric as *Throughput (Mbit/s)/Latency (ms)*. Having twice the *efficiency* means achieving the same throughput with half the RTT or doubling the throughput with the same RTT. In Fig. 10, as *efficiency* increases, the data point will move towards the upper-left. Average *efficiencies* are presented in Table I. To assess how high the *efficiency* is maintained, Table I also presents the 25^{th} percentile of *efficiency* in parentheses. Additionally, the best (highest) values for the respective networks are highlighted in bold.

César shows outstanding performance in achieving high throughput and low latency simultaneously across all mobile carriers' 5G and LTE networks, positioning it at the most upper-left for all cases in Fig. 10. In Table I, *César* achieves the highest average *efficiency* compared to all other CCAs in all networks. Additionally, even the 25^{th} percentile *efficiency* exceeds the average *efficiency* of all other CCAs in all networks except for BBR on LGU+ LTE network. This confirms *César*'s ability to maintain consistently high and stable *efficiency*. Also, *César* achieves the lowest 95% tail latency among the CCAs with high throughput in all networks. These results demonstrate that *César* does not depend on any specific mobile carrier nor network, and exhibits outstanding performance across various bandwidth and RTT_{min} conditions.

While *César* has a slightly higher latency compared to CCAs with the lowest throughput and RTT (e.g., Copa in Fig. 10d), CCAs that focus primarily on latency are not effective for achieving high throughput (*César* achieves 20.78× average *efficiency* compared to Copa in Fig. 10d). Despite ExLL showing low 25^{th} and 75^{th} percentile latency, the tail latency is too high, and it fails to fully utilize the bandwidth. BBR can achieve higher throughput compared to other CCAs because it uses the estimated maximum bandwidth within a certain window, which can be high when bursty traffic is processed all at once within a window. However, as mentioned in §III-B, this requires setting the proper window size and depends on scheduling luck.

In the SKT LTE network, *César* achieves throughput similar

TABLE I: Mean and 25th percentile (in parentheses) of *Throughput/Latency Efficiency* for Fig. 10.

Network	CCA													
	Cubic		BBR		ExLL		PropRate		*César*		Vegas		Copa	
SKT LTE	2.57	(1.8)	4.56	(2.92)	3.12	(2.68)	4.35	(3.13)	**6.12**	(**4.88**)	0.46	(0.14)	0.34	(0.09)
KT LTE	0.48	(0.29)	1.99	(1.32)	2.1	(1.36)	1.84	(1.34)	**2.71**	(**2.26**)	0.7	(0.13)	0.45	(0.01)
LGU+ LTE	0.89	(0.59)	3.17	(2.47)	1.74	(1.43)	2.19	(0.84)	**3.58**	(**2.84**)	0.42	(0.05)	0.25	(0.01)
SKT 5G	12.67	(9.34)	13.39	(5.71)	6.11	(4.89)	4.87	(0.18)	**17.66**	(**14.76**)	1.03	(0.25)	0.85	(0.14)
KT 5G	3.25	(2.51)	5.89	(3.47)	3.32	(2.73)	5.09	(3.02)	**7.67**	(**6.87**)	0.88	(0.13)	0.36	(0.03)
LGU+ 5G	21.86	(18.68)	27.06	(21.13)	17.68	(15.11)	8.08	(0.42)	**34.81**	(**31.93**)	2.13	(0.29)	1.27	(0.2)

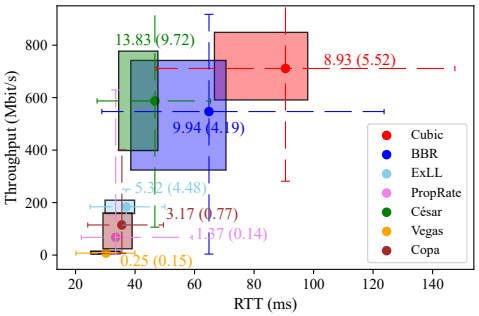


Fig. 11: Throughput, latency, and *efficiency* performance of different CCAs under mobility.

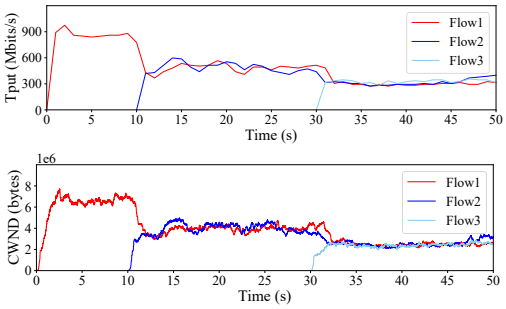


Fig. 12: Throughput and *cwnd* of *César* in a 3-flow scenario.

# flows	Cubic	BBR	*César*
2	0.992	0.998	0.999
3	0.995	0.987	0.994
4	0.996	0.990	0.991

TABLE II: Jain's fairness index when multiple flows start together.

to Cubic while attaining the lowest latency compared to BBR and PropRate which also achieve high throughput. *César* achieves 1.34× and 1.40× higher average *efficiency* compared to BBR and PropRate, reducing RTT by 25% and 29% for the same throughput. Similar improvements can also be observed in the KT LTE network. In the LGU+ LTE network, *César* exhibits slightly lower throughput compared to Cubic and BBR but shows significantly superior latency performance.

In SKT and KT 5G networks, while all other CCAs fail to reach the throughput of Cubic or show large variations, only *César* achieves high throughput with small variations and simultaneously maintains low latency. In the LGU+ 5G network, *César* exhibits slightly lower throughput than BBR but demonstrates outstanding latency performance, with its average latency being close to BBR's 25th percentile latency.

D. Performance under Mobility

In a cellular network, UEs experience channel variations due to mobility. Therefore, we conduct experiments under mobility conditions as well. A UE starts from a point with an RSRP of -85 dBm, considered a good channel, and moves along a predefined path to a region with an RSRP of -105 dBm, considered a poor channel, over the first 60 seconds. Then, it returns to the starting point over the next 60 seconds at the same speed. We conduct this 120-second experiment 15 times for each CCA over 2 weeks on the SKT 5G network.

Fig. 11 plots the throughput, latency, and their *efficiency* of various CCAs. *César*'s position at the upper-left of the plot indicates its effectiveness in achieving both high throughput and low latency simultaneously even under mobility. In contrast, BBR fails to adapt well to channel variations, leading to a significant increase in latency. As a result *César* achieves 1.39× average efficiency compared to BBR (was 1.32× in the stationary scenario). In general, *César* shows higher *efficiency* compared to all compared CCAs, effectively achieving both high throughput and low latency even under mobility.

E. Fairness in Multiple Flows

Fig. 12 plots the `cwnd` and throughput of *César* when flow 1 initially occupies the bottleneck bandwidth, and flow 2 and 3 join the network at t=10s and 30s, respectively. When flow 2 joins, flow 1 and 2 share the bandwidth fairly, and when flow 3 joins, all three flows share the bandwidth equally. Table II presents the Jain's fairness index of average throughput when multiple flows start together and coexist for 30 seconds. *César* achieves a fairness score of 0.99, similar to Cubic and BBR, demonstrating its ability to share the bottleneck capacity.

VI. Conclusion

No matter how high the bandwidth and low the latency provided by cellular networks are, users cannot fully utilize these benefits without proper operation at the transport layer. Current delay-based CCAs fail to account for the scheduling characteristics and the corresponding non-congestive delays of modern cellular systems. To address this problem, we proposed *César*, a cellular resource scheduling-aware congestion control with only sender-side modification. *César* identifies the scheduling characteristics of the current cellular link through the ACK intervals, and adjusts the `cwnd` in a *step-wise* manner to minimize the effects of *scheduling delays*. We have demonstrated *César*'s ability to achieve high throughput and low latency simultaneously in multiple commercial networks.

While *César* effectively reduces the impact of *scheduling delays* on `cwnd` adjustment, *César* does not distinguish queuing delays from occasional non-congestive delays such as handover. However, applying our insights in Open-RAN which facilitates cross-layer approaches can make this possible and practical. We leave this as our future work.

REFERENCES

[1] M. I. Rochman, V. Sathya, D. Fernandez, N. Nunez, A. S. Ibrahim, W. Payne, and M. Ghosh, "A comprehensive analysis of the coverage and performance of 4G and 5G deployments," *Computer Networks*, vol. 237, p. 110060, 2023.

[2] J. Kim and S. Bahk, "Blockage-Aware Flow Control in E-UTRA-NR Dual Connectivity for QoS Enhancement," *IEEE Access*, vol. 10, pp. 68 834–68 845, 2022.

[3] S. Jung and S. Bahk, "Online Control of Traffic Split and Distributed Cell Group State Decisions for Multi-Connectivity in 5G and Beyond," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 2843–2858, 2022.

[4] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Operating Systems Review*, vol. 42, no. 5, p. 64–74, jul 2008.

[5] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet: Networks without effective AQM may again be vulnerable to congestion collapse," *Queue*, vol. 9, no. 11, p. 40–54, nov 2011.

[6] H. Jiang, Z. Liu, Y. Wang, K. Lee, and I. Rhee, "Understanding bufferbloat in cellular networks," in *Proceedings of the ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design (CellNet'12)*, 2012, p. 1–6.

[7] Y. Guo, F. Qian, Q. A. Chen, Z. M. Mao, and S. Sen, "Understanding On-device Bufferbloat for Cellular Upload," in *Proceedings of the Internet Measurement Conference (IMC'16)*, 2016, p. 303–317.

[8] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of LTE: effect of network protocol and application behavior on performance," in *Proceedings of the ACM SIGCOMM*, 2013, p. 363–374.

[9] V. Arun and H. Balakrishnan, "Copa: Practical Delay-Based Congestion Control for the Internet," in *Proceedings of the Applied Networking Research Workshop (ANRW)*, 2018, p. 19.

[10] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Congestion Control for Web Real-Time Communication," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2629–2642, 2017.

[11] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, p. 20–53, Oct 2016.

[12] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," *SIGCOMM Computer Communication Review*, vol. 24, no. 4, p. 24–35, Oct 1994.

[13] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, and K. Lee, "ExLL: an extremely low-latency congestion control for mobile cellular networks," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2018, p. 307–319.

[14] W. K. Leong, Z. Wang, and B. Leong, "TCP Congestion Control Beyond Bandwidth-Delay Product for Mobile Cellular Networks," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2017, p. 167–179.

[15] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive Congestion Control for Unpredictable Cellular Networks," in *Proceedings of the ACM SIGCOMM*, 2015, p. 509–522.

[16] S. Abbasloo, Y. Xu, and H. J. Chao, "C2TCP: A Flexible Cellular TCP to Meet Stringent Delay Requirements," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 918–932, 2019.

[17] X. Xie, X. Zhang, and S. Zhu, "Accelerating Mobile Web Loading Using Cellular Link Information," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'17)*, 2017, p. 427–439.

[18] Y. Xie, F. Yi, and K. Jamieson, "PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements," in *Proceedings of the ACM SIGCOMM*, 2020, p. 451–464.

[19] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, and H. Balakrishnan, "ABC: A Simple Explicit Congestion Controller for Wireless Networks," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, Santa Clara, CA, Feb. 2020, pp. 353–372.

[20] 3rd Generation Partnership Project (3GPP), "NR; Physical channels and modulation," 3GPP, TS 38.211, 2024, version 18.3.0.

[21] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu, "Supporting Mobile VR in LTE Networks: How Close Are We?" *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, March 2018.

[22] 3rd Generation Partnership Project (3GPP), "NR; Physical layer procedures for data," 3GPP, TS 38.214, 2024, version 18.3.0.

[23] F. Capozzi, G. Piro, L. Grieco, G. Boggia, and P. Camarda, "Downlink Packet Scheduling in LTE Cellular Networks: Key Design Issues and a Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 678–700, 2013.

[24] V. Arun, M. Alizadeh, and H. Balakrishnan, "Starvation in end-to-end congestion control," in *ACM SIGCOMM*, 2022, p. 177–192.

[25] A. Balasingam, M. Bansal, R. Misra, K. Nagaraj, R. Tandra, S. Katti, and A. Schulman, "Detecting if LTE is the Bottleneck with BurstTracker," in *The 25th International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2019.

[26] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proceedings IEEE International Conference on Computer Communications (INFOCOM)*, vol. 3, 2000, pp. 1157–1165.

[27] R. V. Bhat, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Network- and application-aware adaptive congestion control algorithm," *Journal of Communications and Networks*, vol. 26, no. 3, pp. 344–355, 2024.

[28] M. Park and J. Paek, "TAiM: TCP Assistant-in-the-Middle for Multihop Low-power and Lossy Networks in IoT," *Journal of Communications and Networks*, vol. 21, no. 2, pp. 188–195, 2019.

[29] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'02)*. ACM, 2002, p. 89–102.

[30] E. F. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 Alpha for wired broadband," *Computer Communications*, vol. 161, pp. 212–224, 2020.

[31] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, 2006.

[32] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, F. Qian, and Z.-L. Zhang, "A variegated look at 5G in the wild: performance, power, and QoE implications," in *Proceedings of the ACM SIGCOMM*, 2021, p. 610–625.

[33] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu., "iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool." 2019, https://iperf.fr.

[34] Innowireless., "XCAL." 2024, https://www.innowireless.co.kr, https://www.accuver.com.