The Quest for the Best: Evaluating Congestion Control in 5G

ROHAIL ASIM, New York University Abu Dhabi, UAE
LAKSHMI SUBRAMANIAN, New York University, USA
YASIR ZAKI, New York University Abu Dhabi, UAE

The rapid evolution of the next generation of mobile networks has paved the path for the development of a wide array of exciting new applications. Instead of gradual and incremental enhancements, each new generation of mobile networks improves the capabilities of cellular networks tenfold, with 5G typically being 10 times faster than 4G, and 6G is anticipated to improve upon 5G by an even higher factor. Given these rapid exponential improvements, it is important to ensure that all layers of the network stack are able to keep pace and support the performance improvements that are possible with these new technologies. Unfortunately, at the transport layer, we still lack a clear understanding of the performance characteristics of current state-of-the-art Congestion Control Algorithms (CCAs) in 5G environments with high channel fluctuations over short timescales. In this paper, we present *Zeus*, a novel framework that enables unified and repeatable evaluation of CCAs in 5G environments. Secondly, we conduct the most comprehensive cross-protocol benchmarking study to date, covering 10 CCAs across real and synthetic 5G traces, buffer regimes, and application scenarios. Finally, we condense the plethora of raw results that we generate and contextualize them in terms of the performance of CCA in real world applications using a scenario-aware scoring system. Our analysis reveals surprising performance inversions and highlights the important considerations that must guide the design and evaluation of future CCA.

1 INTRODUCTION

5G networks exhibit several characteristics that make them unique compared to previous generations, such as significantly higher network variability over short time scales [46]. Despite the vast array of CCAs proposed over the past two decades and the large number of studies conducted to measure their performance in 5G [27, 34, 40, 41], we still lack a detailed understanding of the performance characteristics of these CCAs across diverse 5G channels [11, 17, 18, 24, 49, 50]. As a result, a careful study across different dimensions, environments, and buffer sizes is required to shed light on CCAs intricacies, and identify their potential strengths, weaknesses, and trade-offs. Due to the nature of 5G, it is important to verify that the behavior of these CCAs conforms to the expectations set by previous evaluations on non-5G networks. As we find in our evaluation (Section 6), this is not always the case. Through this study, we identify the CCAs that are currently best positioned to rise to the 5G challenge and we observe the protocol designs that require modifications to efficiently leverage the performance gains provided by 5G and support the exciting new applications enabled by these improvements.

This paper aims to understand the dynamics of different CCAs in 5G network environments and provide a uniform framework for the evaluation of CCA which can be used to conduct a broad array of CCA evaluations while maintaining the ability to directly compare the results to existing evaluations using the same metrics. The main contributions of this paper are as follows:

- A comprehensive analysis of the strengths and weaknesses of prominent CCA across various 5G network environments
- A framework for unified real-world and simulation-based experimentation for the analysis of the CCA performance in any network environment coupled with a dataset of network traces collected for these experiments encompassing a diverse range of network environments[1]

[1]This evaluation framework and the dataset of collected network traces will be made open-source upon publication

Authors' Contact Information: Rohail Asim, rohail.asim@nyu.edu, New York University Abu Dhabi, UAE; Lakshmi Subramanian, New York University, USA, lakshmi@nyu.edu; Yasir Zaki, New York University Abu Dhabi, UAE, yasir.zaki@nyu.edu.

Type of network	Optimal CCA from throughput	Optimal CCA from delay	Throughput/ delay trade-off	Closest to Oracle	Closest to Optimal	Oracle dist. from optimal
City drive (real 5G)	Cubic, Reno	Copa	Vivace (1 BDP)	BBR	Vivace (1 BDP)	88ms
Beachfront walk (real 5G)	Cubic, Reno	Copa, Vivace, Ledbat	BBR	BBR	BBR (1 BDP)	8ms
Rural Macro (NYUSIM)	Cubic, Reno	Ledbat	BBR (1 BDP)	Cubic (10 BDP)	Reno (5 BDP)	137ms
Urban Macro (NYUSIM)	Cubic, Reno	Ledbat	Verus (5 BDP)	Cubic (10 BDP)	Verus (5 BDP)	137ms
Street Canyon (NYUSIM)	Cubic, Reno	Copa	BBR (10 BDP)	Reno (inf BDP)	BBR (10 BDP)	214ms

Table 1. 5G channels summary of results. Distance between the performance of two protocols is measured in terms of throughput/delay trade-off performance.

Metric	Reno	Cubic	BBR	Allegro	Verus	Proteus-P	Vivace	Copa	Proteus-S	Ledbat
Util. (%)	99.7 ± 0.3	99.1 ± 0.4	95.9 ± 0.6	87.6 ± 5.7	84.6 ± 4.7	80.7 ± 3.4	72.9 ± 6.9	63.5 ± 8.2	54.2 ± 7.8	4.9 ± 0.9
Delay	$15.6x \pm 2.8$	$12.2x \pm 1.5$	$4.3x \pm 0.6$	$11.4x \pm 2.1$	$67x \pm 30$	$57.4x \pm 12.4$	$3.4x \pm 0.5$	$1.5x \pm 0.2$	$25.8x \pm 10.2$	$1.7x \pm 0.3$
Intra-fairness	0.88	0.86	0.87	0.84	0.85	0.92	0.87	0.98	0.91	0.91
Inter-fairness	0.81	0.86	0.63	0.64	0.64	0.55	0.66	0.76	0.54	0.58

Table 2. CCAs results summary. Utilization is the CCA's channel utilization, delay is the queueing delay, and fairness is the Jain-Fairness-Index (we are measuring for two flows, so the index is lower-bounded by 0.5).

For our experimental setup, we perform real-world experiments in the wild using a commercial 5G deployment as the client and a server with different possible configurations of the active CCA. However, real-world experiments present many challenges due to factors including high costs and the high variability in the network environment that renders it difficult to distinguish between changes in protocol performance caused by the environment variability in real time as opposed to protocol performance. We complement these experiments with a wide array of simulation-based experiments that emulate the real-world environments, provide full control over the environment, and are immune to the challenges that hinder repeatability and reproducibility of real world experiments. For the simulation-based experiments, we gather a diverse collection of 5G channel traces, including real-world traces from commercial 5G deployments and trace-driven models built in the NYUSIM model [42]. Finally, we assess 10 prominent CCAs using the enhanced evaluation framework and the collected 5G traces. The chosen CCAs include legacy Internet CCAs like TCP Cubic [16] and TCP Reno, as well as newer state-of-the-art (SotA) algorithms including BBR at Google [8], Ledbat at BitTorrent [39], Copa at Facebook [5], Vivace [13], Allegro [12], Proteus-P [31], Proteus-S [31], and Verus [48]. We compare these CCAs with two reference benchmarks: (i) a *Delayed Oracle Model* where the base station can compute the best congestion response based on the channel state at a given time, but the sender receives this feedback after a propagation delay by which time the channel state may have changed; (ii) an optimal offline throughput and delay computation based on complete knowledge of the channel variability ahead of time. Finally, we introduce a scenario-aware scoring system to compile the results into a simplified metric that informs users how these protocols will perform in real world applications.

Table 1 provides a summary of our results showing the best-performing CCAs on different 5G channels, as examined in this paper. We evaluate CCAs with different buffer sizes, expressed as multiples of the bandwidth-delay product (BDP), based on three distinct metrics: throughput utilization, maintaining low network delays, and achieving a favorable balance between throughput and delay. Table 2 presents the average CCA results across the 5G traces, focusing on channel utilization, delay increase relative to the baseRTT (that only accounts for propagation delay), and intra/inter-fairness when competing with another flow over the same link. The CCAs are listed according to their channel utilization, with Reno and Cubic dominating at over 99% utilization. BBR ranks third with approximately 96% average channel utilization, followed by Allegro and Verus. Ledbat exhibits the lowest channel utilization at a mere 5%.

Our results reveal key insights to inform the future of CCA design and reinforce the need of a unified evaluation framework. No single CCA dominates across all conditions. Legacy protocols such as Cubic and Reno continue to outperform many newer designs in throughput-intensive

scenarios, while delay-sensitive algorithms like Copa and Vivace excel in real-time and fair-sharing environments. Learning-based CCAs often exhibit instability and poor coexistence behavior in volatile 5G settings. We further find that protocol behavior consistently maps onto a throughput-delay trade-off frontier, where improvements in one dimension come at the cost of the other. This frontier is shaped by buffer size and protocol tuning but cannot be fundamentally circumvented. To make sense of the large number of data points, we introduce a scenario-based QoE scoring system that maps CCA behavior to practical deployment needs, offering protocol recommendations for streaming, real-time, and mixed-traffic environments. This paper attempts to answer the question: what is the best congestion control algorithm for 5G for different applications and network contexts? Zeus provides a principled and extensible platform to make that answer reproducible, quantitative, and scenario-aware.

2 RELATED WORK

Evaluating congestion control algorithms (CCAs) has long been a central focus of networking research. New CCAs are typically coupled with an analysis and comparison against a small number of existing protocols under selected conditions. For instance, Copa [5], BBR [8], PCC-Vivace [13], and Proteus [31] each present comparative evaluations, but these tend to be limited to low-bandwidth environments, limited buffer configurations, and a small set of network environments. Controlled network environments often do not reflect the burstiness, queuing dynamics, or RTT variability found in commercial 5G networks. As a result, most prior studies, while sufficient to demonstrate their performance for certain scenarios, these evaluations are often narrow in scope, use inconsistent methodologies, and yield results that are difficult to compare or generalize across protocols and deployment scenarios.

A wide range of simulation and emulation tools have emerged to support congestion control research over the years. Simulators such as NS-2/NS-3 [21, 37] and OMNeT++ [43] are widely used due to their flexibility in defining topologies, transport models, and link-layer behavior. However, they lack execution-time realism and are not designed to run real protocol stacks or real applications. Real-time emulators such as Dummynet [38], NetEm [20], Mininet [19], Hercules [35], and Mahimahi [33] provide more realistic testing by operating with live applications and transport stacks. These tools allow traffic to be shaped at runtime, enabling more representative evaluations of protocol performance. However, they are inherently limited by hardware capacity, making it difficult to scale to multi-Gigabit rates without introducing bottlenecks in replay fidelity or timing accuracy. In particular, Mahimahi supports simple delay and bandwidth variability, but it the original implementation of Mahimahi suffers from bandwidth limitations discussed in 4.2. Pantheon [47] was another framework for CCA evaluation through automation and reproducibility. It included a curated set of CCAs and a test harness for benchmarking across a controlled testbed. However, it was primarily designed for legacy TCP protocols and does not support high-throughput mobile environments, modern transport protocols, or cellular traces with realistic 5G dynamics. Moreover, the project is no longer actively maintained and lacks support for extensibility with newer CCAs or application-specific workloads.

Beyond software tools, several 5G-specific testbeds have been deployed in recent years to provide experimental infrastructure for mobile networking research. Examples include 5TONIC [2], 5GIC [25], FOKUS [14], COSMOS [36], POWDER [7], and AERPAW [30]. While these platforms offer access to real hardware, radio resources, and edge-cloud setups, they are often geographically constrained, require application approval, and are generally tailored to specific experiment types (e.g., wireless PHY testing, MEC deployments). As such, they are not designed for reproducible, trace-driven protocol evaluation at scale, and lack the flexibility and accessibility needed for rapid iteration across a large set of CCAs.

In contrast, Zeus is a unified, extensible framework that addresses these tooling and methodological limitations. It enables reproducible, high-throughput evaluation of CCAs using real-world 5G traces. Zeus extends Mahimahi to support the capabilities critical for capturing the high bandwidth capacities, short-timescale variability, and burstiness of 5G. Unlike prior tools, Zeus standardizes the evaluation process, enabling fair, side-by-side comparisons across a diverse set of protocols. This paper utilizes these capabilities conduct a comprehensive benchmarking study of 10 CCAs across different traces and buffer configurations, and multiple metrics including fairness, delay inflation, and harm to Cubic highlighting the trade-offs relevant for real-world deployment.

3 RESEARCH METHODOLOGY

Assessing these CCAs in 5G environments remains a complex challenge. Factors contributing to the difficulty of this task include the 5G environment's variability due to uncontrollable elements such as competing network traffic, signal fluctuations, and the large propagation losses caused by the high-frequency nature of mmWave signals. Due to such factors, running the exact same 5G real-world experiment at the same location can yield highly variable results. Cost is another significant factor; a single one-minute experiment can consume up to 7 GB of data at a rate of 1 Gbps. This makes the analysis of real networks extremely difficult because one cannot attribute the observed effects only to CCA change, given these factors. As a result, researchers often resort to simulators or emulators. However, when it comes to 5G, access to high-quality prototype frameworks is limited, restricting the ability to test novel CCAs in realistic 5G settings. Given these challenges, it is vital to evaluate and compare various CCAs across diverse 5G channels in a consistent manner. Repeatability and reproducibility are crucial to drawing accurate and meaningful conclusions, as the variability of 5G makes evaluating CCAs in the real world challenging and expensive.

3.1 Metrics for Evaluating CCAs

(i) End-to-end throughput and delay: It is reasonable to question whether existing CCAs are compatible with the characteristics of 5G networks and whether their performance is impacted by the unique challenges presented in 5G environments. Specifically, we focus on evaluating the end-to-end throughput and delay in the context of the diverse 5G networks.
(ii) Fairness and Harm: In today's literature, it has become customary to evaluate the fairness of new CCA with respect to legacy solutions when sharing a bottleneck. To determine this, several papers use Jain's fairness index. However, in our analysis, in addition to Jain's fairness index, we also evaluate a more practical harm-based approach, defined in [44]. This approach analyzes the harm that a new CCA causes to an existing legacy CCA when sharing a bottleneck to ensure that new CCA does not cause more harm than existing solutions. We chose Cubic as the legacy CCA, since it is the default TCP flavor on many of today's platforms, such as Linux [45] and Android OS [3]. We measure the harm caused to Cubic flows by other competing CCA flows to determine if the new CCA is suitable for coexistence with Cubic in the wild.
(iii) Impact of Buffer Sizes: A critical aspect of evaluating a CCA in cellular networks is examining the impact of the bottleneck buffer size. Many previous studies have either overlooked this vital parameter or concentrated on relatively small buffer sizes. Therefore, we assess four buffer size settings based on the bottleneck buffer size: 1 BDP, 5 BDPs, 10 BDPs, and an infinite buffer.

3.2 Reference Benchmarks

To better understand the challenges 5G environments pose for CCAs, we introduce two reference models to compare against: the *Optimal reference model*, and the *Delayed feedback Oracle model*. The optimal reference model represents the ideal operating point with a fully saturated channel and baseRTT assuming complete knowledge of the channel variability ahead of time. The Delayed

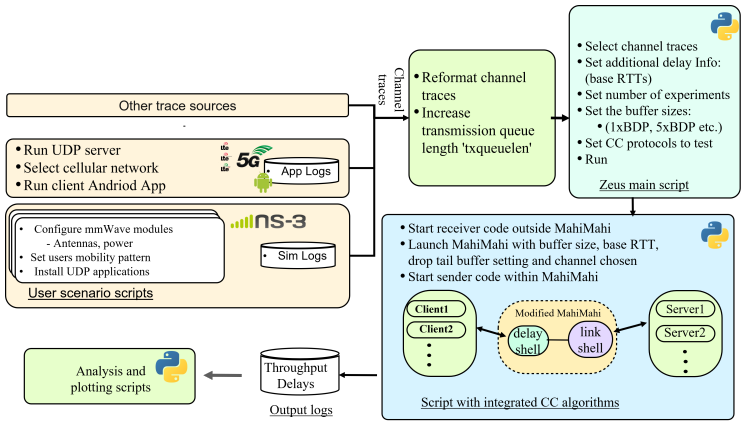


Fig. 1. Architecture of the *Zeus* framework

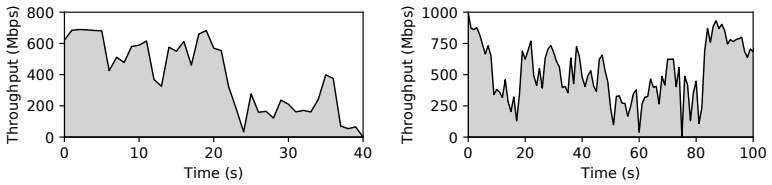
Feedback Oracle model represents a hypothetical algorithm situated at the cellular base station with perfect knowledge of the current 5G channel state only at a given time without any future knowledge about the state of the channel. If the sender could access this information and the primary bottleneck was the cellular link, it could fully utilize the link without incurring additional delays. However, since this information is only available at the base station, it must be shared with the sender, introducing one-way delay, followed by another one-way delay for the data to reach the receiver. In theory, this is the best information a CCA can use, but most lack access to it and instead infer it through signals like delays, inter-arrival times, etc.

4 THE *ZEUS* FRAMEWORK

This section describes *Zeus*, a framework designed and developed to conduct performance analysis of CC algorithms over 5G channels. The analysis methodology described in this section is general and technology-agnostic so that it can be extended to study the performance of CC algorithms over other types of channels. The overall workflow is detailed in Figure 1. As for the channel traces, different sources can be used as long as they follow the appropriate format detailed in Section 4.2. Furthermore, the framework includes a set of traces obtained from two different sources: a real 5G cellular network of a commercial operator (Non-StandAlone (NSA)) and mmWave traces generated with an ns-3 network simulator. *Zeus* embeds a tailored version of ns-3 [37] to generate traces for mmWave scenarios, using the mmWave module [32] developed by NYU wireless. Figure 2 shows 2 of the recorded channels. The link emulator can be tailored with additional end-to-end delay, loss rates, and/or modifying the buffer size of the link. The framework sets up sender/receiver pairs using a CC algorithm and connects them through Mahimahi. The results allow us to compare the behavior of different CCAs in the exact same environment in a systematic manner that is repeatable and reproducible. *Zeus* also allows the analysis of bottleneck link-sharing among traffic flows. It leverages the best out of the real-life and emulation realms without the reproducible limitations.

4.1 Real 5G network traces

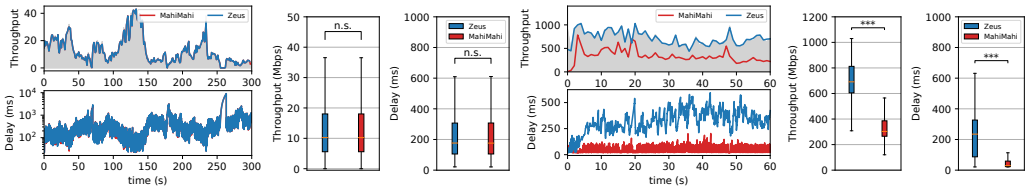
For the generation of real 5G network channel traces, *Zeus* offers a client and server-side implementation that can record real-time 5G channels in a cellular environment. For the trace generation, the server must have sufficient upload capability exceeding the maximum download speeds of the 5G client. The client consists of an Android application that communicates with the server. Upon connecting, the Linux-based server begins sending UDP packets with a maximum transmission



(a) 5G city drive (b) Street canyon

Fig. 2. 5G channel traces

unit (MTU) size of 1500 bytes at a constant data rate to the Android mobile client. The Android application serves as a sink and logs the inter-arrival times of the received UDP packets. These logs are then used to create the trace files, which are converted to the proper channel trace format compatible with the modified Mahimahi emulator.



(a) Verizon-LTE (4G) (b) Stat. signif. (4G) (c) Indoor InHM (5G) (d) Stat. signif. (5G)

Fig. 3. Benchmarking *Zeus* against Mahimahi on 4G and 5G channels

4.2 Modified Mahimahi

The original Mahimahi implementation has limitations that affect the emulation of multi-Gigabit capacity channels, such as those based on mmWave. The original implementation has certain bottlenecks that result in a limit of around 400 Mbps on the throughput that the network can handle efficiently. Any trace exceeding the threshold of ≈ 400 Mbps causes Mahimahi to drop packets randomly, thus capping the channel throughput and utilization. To resolve this, we modify Mahimahi to be able to overcome this limitation. The most critical modification involved drastically reducing the required number of inter-process events for emulating high-bandwidth 5G traces.

Figure 3 shows a performance comparison of the modified emulator to the original Mahimahi implementation. The results are obtained by emulating a network flow with consistent full-buffer transmission. Figures 3a and 3b show the results obtained when a 4G Verizon-LTE [45] trace was used, with a maximum capacity around 40Mbps. Figures 3c and 3d show the performance obtained over a synthetic 5G channel generated in ns-3 using the Indoor Hotspot Model (InHM), with capacity reaching 1Gbps. Both channel capacities are shown in Figures 3a and 3c with the shaded background. It is evident that for the 4G Verizon-LTE channel, both the original Mahimahi and *Zeus* versions offer the same performance, with no statistical significance (n.s.) observed in terms of throughput and delay between the two, completely saturating the channel capacity. However, for the InHM 5G channel, the original Mahimahi version fails to handle the number of sending events, leading to the under-utilization of the channel capacity and eventual packet losses due to buffer overflow. In contrast, *Zeus*'s Mahimahi manages to support all the generated traffic, enforcing the correct emulation of the 5G channel, thus mimicking the correct behavior of the CCA in use. A considerable statistical difference for both throughput and delay is observed in Figure 3d where the achieved throughput for Zeus is significantly higher leading to bufferbloat and higher queuing

delays, whereas the exact same experiment with the original Mahimahi implementation is not capable of saturating the available bandwidth.

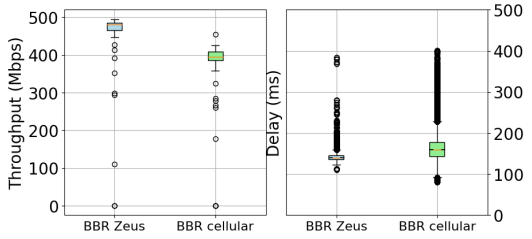


Fig. 4. *BBR* in a real 5G cellular connection vs. *Zeus*

4.3 *Zeus*' operation validation

We validate the correct operation of *Zeus* by comparing its performance with real-world tests. Figure 4 shows the results obtained from three real 5G cellular connections, each lasting 60 seconds, using a server running *BBR* and a client device using a real 5G connection in the wild, and results obtained using *Zeus* to simulate a 60 seconds connection using *BBR* over similar network traces recorded at the exact location where the real 5G experiments were conducted. The figure shows that the results obtained with *Zeus* are similar to those obtained from the actual 5G connections, both in terms of average value and sparsity. In particular, the throughput obtained in both cases, *Zeus* and cellular, reach comparable average values and tight distribution. In the case of the delay, the average values are again alike, while the sparsity observed for the *BBR* cellular connection is slightly larger. Although the results are not identical, they serve to ensure that the CC protocols experience similar and comparable conditions with *Zeus* to a real cellular connection. Obtaining identical results is not possible, since the wireless channel realizations are different for the 5G connection running *BBR* and those using UDP to generate the traces used by *Zeus*.

5 EXPERIMENTAL METHODOLOGY

5.1 Real World Experiments

Evaluating the performance of congestion control algorithms (CCAs) in real-world mobile networks poses significant challenges due to the variability and complexity of such environments. While trace-driven simulations offer a controlled means to address many of these challenges, they may not fully capture the intricacies of real-world network behavior. To complement simulation-based evaluations, we conducted a subset of experiments in live mobile network conditions for further testing and validation. The setup involved using a 5G-enabled smartphone as the client device and an AWS EC2 instance in the same geographic region as the server. The server was configured to support five different CCAs via the Linux pluggable congestion control interface.

Using the *iperf3* tool, we performed network experiments in which data was sent from the server to the client using the selected transport protocols. For each protocol, six independent trials were conducted. These experiments allowed us to observe key performance metrics, including throughput and variations in the congestion window size for each protocol. By combining simulation and real-world testing, this approach provides a more comprehensive evaluation of CCA performance, highlighting their behavior under controlled and dynamic network conditions.

5.2 Emulation Environment

To create a consistent testing ground for checking the effectiveness of CCAs in a way that is realistic, repeatable, and reproducible, we improved the Mahimahi link emulator [33], which is commonly used to test CCAs in a range of network conditions [4, 5, 13, 15, 47]. This enhancement aims to improve Mahimahi's support for multi-gigabit capacity traces, allowing for the assessment of CCAs without altering their original implementations. The most notable modification involves reducing the number of internal inter-process I/O system calls. In the original implementation of Mahimahi, if multiple packets were to be scheduled simultaneously, Mahimahi would allocate a separate I/O system call for each. In 5G, where capacity can reach multiple Gbps, this can generate hundreds or thousands of system calls at once. We optimize this by introducing a new channel trace format, where multiple packets scheduled at the same time are combined into a single system call. We intend to make this extension along with our evaluation framework open-source upon publication. Further details and validation of our framework are provided in Section 4.

5.3 Generating Realistic Network Traces

We employed two distinct methods for generating the channel traces: i) a real 5G cellular network, and ii) mmWave traces generated with the NS-3 simulator. For the real 5G traces, we used a similar trace collection methodology as [45, 48], with a commercial 5G connection under different mobility scenarios. The details of our approach are mentioned in Section 4. Two such traces are used in this paper, namely: City drive, and Beachfront walking. Finally, the mmWave traces were generated using the mmWave module [32] built atop NS-3 [37], developed by the New York University wireless group (NYU Wireless). We collected two different groups of traces: with buildings and predefined users' motion, and without buildings and with random users' motion. For the first group, we deployed a grid of 3×3 buildings and defined different users' tracks. The Street Canyon comprised two basestations, and users moved following a straight line, getting close or going far to/from each of the base stations. This trace used the urban micro (UMi) 3GPP model. The second group of traces represents open areas. In particular, we exploit the Urban Macro (UMa), and Rural Macro (RMa) 3GPP models to generate the last two traces [51].

5.4 CCAs Evaluation Setup

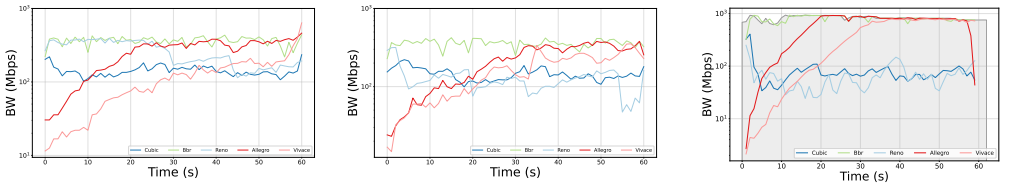
All experiments were conducted on a customized server with an Intel Xeon Bronze 3204 CPU @ 1.90GHz × 12, 15 GiB memory, and Ubuntu 20.04.3 LTS operating system. To ensure accuracy, we consulted with some of the authors of these CCAs to verify their configuration and behavior. For each CCA-channel pair, we performed 5 experiments with four bottleneck buffer sizes namely 1 BDP, 5 BDP, 10 BDP, and infinite buffer.

6 RESULTS

6.1 Real World Experiments

The throughput trends of the real-world experiments, a subset of which are shown in Figure 5, provide valuable insights into the performance of congestion control algorithms (CCAs) under live mobile network conditions. Using a 5G-enabled smartphone as the client and an AWS EC2 instance as the server, we observed significant variations in key performance metrics across the five CCAs tested. We compare the real world experiment to the results of simulation experiments configured to emulate a similar high-bandwidth real 5G network environment.

The throughput results highlight the impact of each CCA's design on its ability to utilize available bandwidth efficiently. We observe that BBR is able to adapt quickly to the volatile high-bandwidth network environment in both cases and consistently achieve high throughput. Legacy CCA are



(a) Real World Experiment 1 (b) Real World Experiment 2 (c) Zeus Emulation Experiment

Fig. 5. Comparison between simulation and real experiments

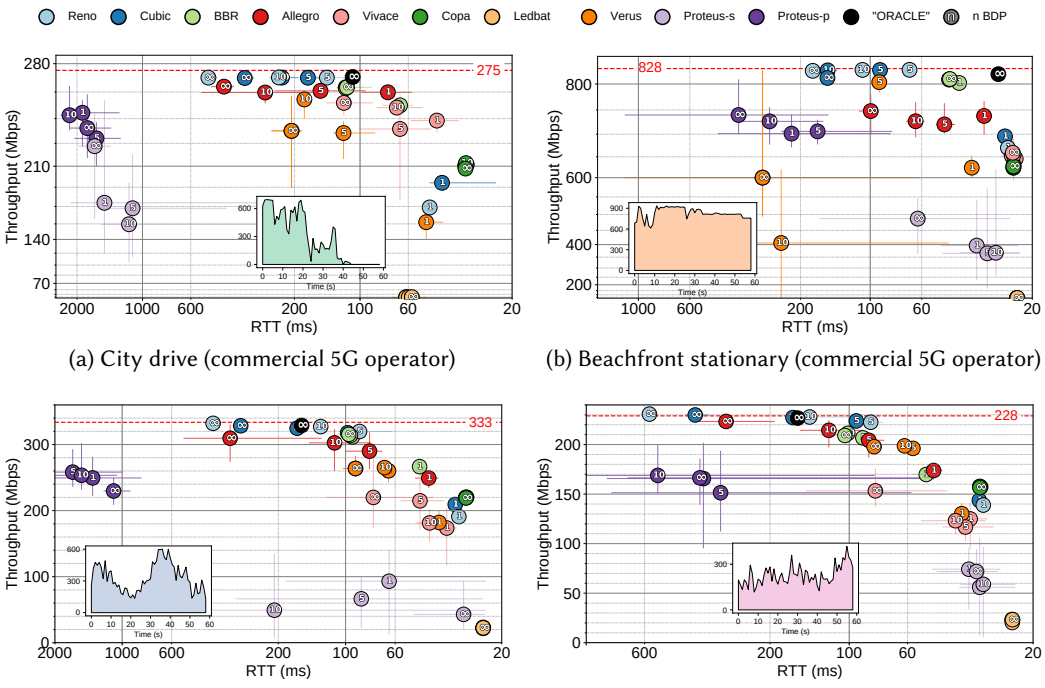
able to maintain high throughput initially but incur bufferbloat which rapidly deteriorates their performance. Allegro and Vivace both gradually converge to their standard operating points in the network environment, which prioritizes high throughput for Allegro, and lower delays for Vivace. Protocols optimized for high-speed environments demonstrated consistently higher throughput, while others exhibited occasional under-utilization due to their more conservative congestion control mechanisms. Packet retransmissions were also observed to vary between protocols, with some CCAs showing resilience in maintaining performance despite sporadic packet drops inherent to mobile networks. Legacy protocols were more sensitive to loss, resulting in a noticeable decline in throughput during challenging network conditions. The replication of the same trends in protocol behavior in the simulation experiments further validate the accuracy of the experimental setup using the simulation framework.

Due to the challenges in real cellular environments, discussed in Section 3, it is not sustainable to conduct a comprehensive analysis of CCA at scale relying solely on real world experiments. Thus we introduce the Zeus framework capable of conducting a comprehensive analysis in a repeatable and reproducible manner.

6.2 End-to-end throughput/delay analysis

Figure 6 illustrates scatter plots that display the performance of ten SotA CCAs over different types of 5G channel traces. The traces include stable channels as well as highly volatile high-bandwidth channels. Each circular data point represents the average throughput (y-axis) and delay (x-axis) performance of a CCA. We conducted 5 separate runs per channel trace for each CCA to ensure the stability of the algorithms' performance and their ability to efficiently use the 5G capacity. Each CCA is indicated by a different color with a number representing one of the four buffer sizes, from as low as 1BDP to an infinite size. The figures also show the capacity over time in an inset subplot.

Legacy CCA' performance (Cubic & Reno): Both Cubic and Reno are known for building large sending windows, resulting in a well-known phenomenon called "bufferbloat" in cellular networks which has been well investigated in the context of 3G/4G cellular networks [23]. From our observations across all 5G traces, we found that TCP Cubic and Reno were able to saturate the channel capacity for 5 BDP, 10 BDP, and infinite buffer sizes (the red dotted horizontal line represents the average channel capacity). However, it was observed that a buffer size of 1 BDP was insufficient for loss-based algorithms like TCP Cubic and Reno to saturate the link capacity, as it would lead to packet losses at a faster rate. Our findings also reveal that increasing the buffer size to more than 5 BDP does not necessarily improve channel utilization and mostly results in higher delay. Compared to other CCAs, TCP Cubic and Reno were associated with the highest delays in almost all cases (except for Proteus-P and Proteus-S). Despite legacy TCP being highly criticized



(a) City drive (commercial 5G operator)

(b) Beachfront stationary (commercial 5G operator)

(c) Rural Macro (NS-3)

(d) Urban Macro (NS-3)

Fig. 6. CCAs under fluctuating 5G channels

given their bufferbloat issues, we show surprising results that they, with reasonable buffer sizes (5-10 BDPs), still outperform many SotA CCAs in some cases, such as the Urban Macro.

Google's BBR performance: BBR's slight under-utilization is due to its operation in a series of states, i.e., the startup phase, a PROBE_BW phase every 8 RTTs to estimate bandwidth, and a PROBE_RTT phase every `10 sec` to re-estimate the minimum RTT. The Probe RTT phase is when BBR briefly reduces its packets in-flight to just four packets, draining any queue build up, which appears to cause the slight under-utilization of the available channel. In our analysis, this design allows BBR to exhibit remarkably consistent performance across all traces, reaching a relatively high channel capacity utilization while slightly underperforming legacy alternatives in terms of utilization. Particularly, its throughput remains slightly lower than the average channel capacity, with 74%-98% utilization for the 1BDP, and 90%-98% utilization for the 5BDPs, 10BDPs, and infinite buffer sizes. Despite BBR's small loss in throughput utilization, it makes up for it in the delay performance, yielding a remarkable reduction in the delays' performance to approximately 50-73% of that of both Cubic and Reno for the larger buffer sizes.

Facebook's Copa performance: Copa also relies on queuing delays for CC and strives to minimize the packet delays by periodically draining the buffers. Copa exhibits stable behavior across all 5G traces, although its efforts to maintain a minimum queuing delay results in the throughput performance being somewhat diminished compared to other alternatives. Copa successfully maintains minimal delays consistently in our analysis. Previous evaluations, conducted under links of up to 100 Mbps [5], show that Copa achieves significantly lower queuing delays, with only a small throughput reduction, outperforming Cubic, BBR, and Allegro. However, we observed that these claims do not necessarily hold in certain 5G environments, where there is a significant throughput penalty associated with Copa. It achieves average utilization between 66% and 76% for all 5G traces.

Based on these findings, Copa can be considered an excellent CCA for applications demanding extremely low delays while preserving decent throughput utilization. The results indicate that Copa prioritizes delay over exploiting the capacity.

The PCC Family performance: Allegro operates in a fixed-step increment or decrement when adjusting the sending rate. In 5G environments, these may be too small. This fixed-step sending rate limits Allegro's ability to quickly adapt to changes in the channel resulting in lower channel utilization and increased delays. In our analysis, Allegro attains marginally lower delays than Reno and Cubic, but generally experiences higher delays compared to BBR, particularly in cases with larger buffer sizes like 10BDP and infinite ones. Moreover, Allegro's throughput is generally lower than BBR's but remains higher than Copa's. An intriguing observation from Allegro's results is that increasing the buffer size negatively impacts its performance, especially in terms of end-to-end delay. This is evident in the high-bandwidth stable 5G channels (see Figure 6), where expanding the buffer size beyond 1BDP does not result in increased throughput but instead significantly raises the average delay. In fact, for many 5G channels, Allegro performs better with a shallow buffer (1BDP). Another new observation is that Allegro achieves 83% link utilization over a rapidly changing network in previous evaluations [12] but in our 5G traces, this property often does not hold.

Vivace is a learning-based CCA using online greedy optimization methods to control its sending rate via a utility function. However, optimization methods are susceptible to getting trapped in a local optima [28], which is why we see variable throughput performance with repeated experiments. Moreover, Vivace desires to approximate the lowest achievable RTT (RTT_{min}) to decide its sending rate, which explains the steady delays. However, in 5G networks, operating at RTT_{min} is not necessarily optimal. In our observations, Vivace consistently maintains lower delays across many of the 5G traces. Previous evaluations show that Vivace significantly outperforms legacy TCP flavors, Allegro, and BBR in throughput, latency, and friendliness towards TCP [13]. We observe that in our 5G traces, the throughput utilization of Vivace drops significantly. Our fairness analysis shows that Vivace's TCP friendliness in 5G is similar to Allegro and BBR. In terms of throughput, Vivace's performance varies depending on the channel being examined. For instance, in the high-bandwidth stable channels displayed in Figure 6, Vivace achieves higher throughput than Copa with almost the same delay. In such situations, Vivace's performance appears to be unaffected by the configured buffer size. However, when assessed over the volatile 5G channels shown in Figure 6, it experiences higher delay compared to Copa and benefits from the increased buffer size.

Proteus-S and Proteus-P are built atop an online learning CC framework [12]. Proteus-S, with its scavenger utility, controls the sending rate and leverages delay variation as a sensitive early indication of flow competition. Proteus-S performance varies significantly across channels, either inducing extremely high delays of approximately 600ms and even more for some traces (see Figure 6a) or exhibiting very low delays for others (see Figures 6d and 6b). Likewise, Proteus-S's throughput is also highly variable, generally leaning towards the lower end. Despite having a strategy to compensate for misleading delay variation in a non-congested channel, it exhibits inconsistent performance in 5G due to the uncertainty of the learning-based components responsible for identifying the changes in equilibrium points. On the other hand, Adhering to a scavenger strategy, Proteus-S struggles to adapt to the variability of the analyzed 5G channels. In contrast, Proteus-P's utilization is higher than Proteus-S's, but it too faces difficulties adapting to highly variable channels, resulting in extremely high delays. Proteus-P controls its sending rate using a modified version of the utility function of Vivace (with negative RTT gradients ignored). These modifications allows it to outperform Vivace in terms of throughput but at the cost of significantly higher delays in 5G. In previous evaluations [31], Proteus achieves 90% utilization when running alone and limits 95th percentile inflation ratio for latency below 10%. We observe that in 5G environments, the performance changes significantly compared to relatively low-bandwidth network environments.

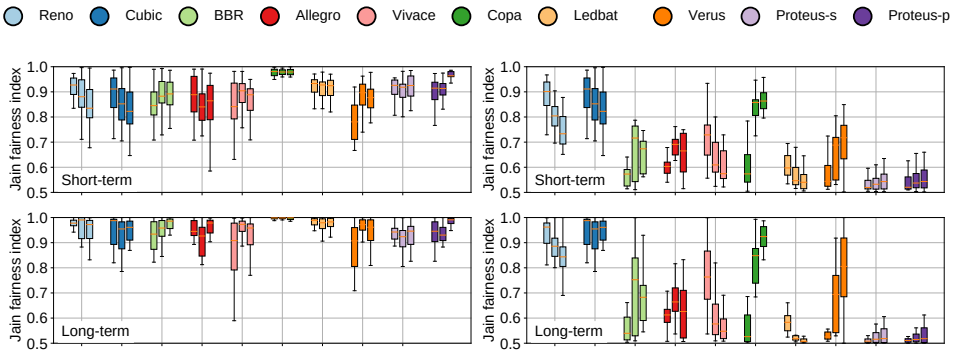
Others: Examining Verus results, it displays varied behavior across all traces, making it unpredictable in 5G environments. The configured buffer size appears to have a significant impact on Verus performance, where increasing the buffer occupancy to larger settings causes the algorithm to create a more substantial bufferbloat. We have excluded the infinite buffer results of Verus in several instances since they were multiple seconds and would have distorted the results visualization. Verus achieves variable performance in 5G. We attribute this to Verus' delay profile curve, which is the basis for adjusting the CWND, struggling to adapt to certain 5G environments. Similarly, contrary to previous non-5G evaluations [39] where Ledbat was able to saturate the link when no other traffic is present, Ledbat is unable to saturate the link capacity in 5G and achieves the lowest throughput across all traces. This is because Ledbat tries to restrict the delays below a predefined target (`default=25ms`), and maintain that value to form a steady-state and never incur delays higher than the target delay. Ledbat's congestion signal is RTT exceeding a target threshold. Due to the variability in 5G environments, this causes Ledbat to achieve low throughput and delay across all channel traces.
The Delayed Feedback Oracle: The hypothetical Oracle algorithm, having a perfect knowledge of the network conditions, is able to maintain high throughput utilization, saturating the channel capacity across all traces. It is also able to maintain low delays in stable channels. However, we see that in highly fluctuating channels the end-to-end delay suffers a significant negative impact causing it to lose to some of the actual CCAs. This is because when there is a steep drop in the channel capacity, the one-way delay causes the algorithm to be slow to react, thus leading to bufferbloat. This exemplifies the difficulty of creating an optimal CCA for 5G.

6.3 Fairness analysis

We evaluate the Jain-fairness-index [22] for all 10 CCAs from two different angles: intra-fairness, where a CCA shares a link with itself, and inter-fairness, where a CCA shares a link with TCP Cubic. Starting with intra-fairness in Figures 7a, the boxplots are divided into short-term fairness (upper subplot) and long-term fairness (lower subplot). These values are computed using a rolling window in steps of $k \times \text{RTT}$, where k is 1, 2, 5, 10, and 20 for short-term and 50, 100, 500, 1000, and 3000 for long-term fairness. For each CCA, three different boxplots are displayed: left for 1BDP, center for 5BDPs, and right for 10BDPs. Nearly all CCAs exhibit decent short-term intra-fairness of 0.8 or higher and even greater long-term fairness of 0.9 or more. In contrast, the inter-fairness results are shown in Figures 7b. Apart from Reno, Cubic, and Copa (for 5 and 10BDPs), all other CCAs demonstrate lower fairness when competing with TCP Cubic, with both Proteus versions and Ledbat having the lowest possible Jain-fairness-index, close to the minimum of 0.5. The primary limitation of the Jain-fairness index is that it measures fairness by assigning equal score in both cases, whether a new CCA utilizes a larger share of the bandwidth than Cubic or vice versa.

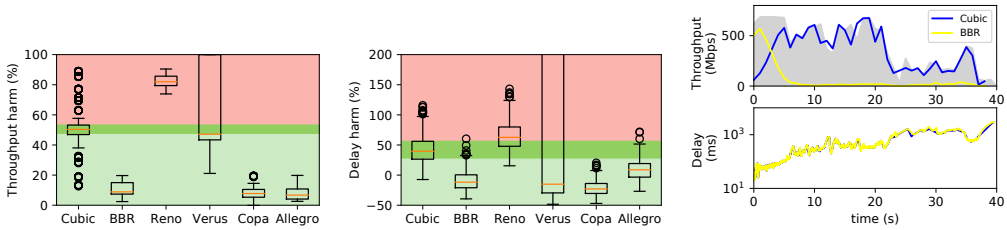
6.4 Harm Analysis

A common requirement for any new CCA is its fairness in sharing the bottleneck bandwidth with existing TCP (e.g., Cubic). However, this goal is too idealistic to execute in practice. It is believed that being unfair to Cubic is acceptable because Cubic is not even fair to itself [5]. Inspired by [44], we follow a harm-based approach in this analysis to quantify the harm the CCAs do to Cubic when they co-exist over the 5G channels. We chose BBR, Reno, and Copa for our analysis because of their wide deployment along with Allegro and Verus based on the results of our end-to-end throughput and delays analysis. Following the harm definition in [44], we define x = solo performance; and y = performance after introduction of a competitor connection. Then for metrics where 'more is



(a) Intra fairness (against itself) (b) Inter fairness (against TCP Cubic)

Fig. 7. CCAs' fairness, each CCA with three boxplots (letf: 1BDP, center: 5BDPs, and right: 10BDPs).



(a) Throughput harm to Cubic (b) Delay harm to Cubic (c) Instantaneous (Cubic/BBR)

Fig. 8. Comparison of CCAs' performance over the 5G city drive channel trace with infinite buffer.

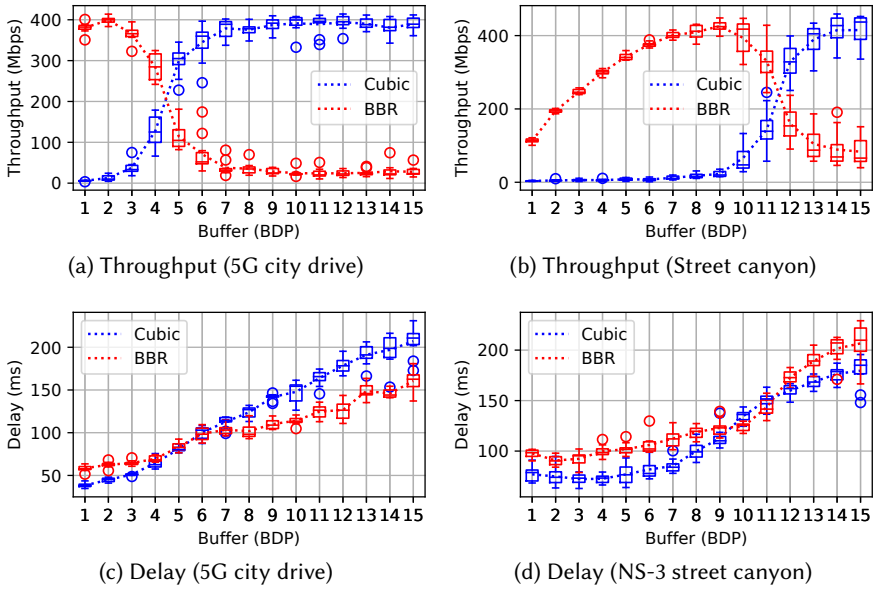
better' (e.g., throughput) the harm = $\frac{x-y}{x} \cdot 100$. On the other hand, for metrics where 'less is better' (e.g., delay) harm = $\frac{y-x}{x} \cdot 100$.

We conducted 20 experiments with an "infinite buffer" for each trace with two Cubic flows co-existing on a 5G channel to calculate Cubic's throughput and delay self-harm as the reference threshold (i.e., the baseline). This threshold provides a firm definition for when a CCA can be deployed alongside Cubic and when it is not. We represent this threshold in the horizontal dark green areas shown in Figures 8a and 8b. The green area represents the safe zone reflecting the acceptability of the algorithm alongside Cubic when sharing a bottleneck. In contrast, the red zone is where the algorithm is not suitable for deployment. We observe that Cubic causes 50% throughput-harm to itself, reflecting an equal share when competing with itself[2].

In all traces, we observe that BBR, Allegro and Copa fall in the green zone not causing throughput or delay harm to Cubic. However, Reno is the only CC that always falls in the red zone and harms Cubic in terms of both throughput and delay. Reno does more than 80% throughput-harm with 60% delay-harm to Cubic in the 5G city drive trace, above 60% throughput-harm, and nearly 90% delay-harm (similar trends have been observed for other traces). This could be why Netflix adheres to Reno due to its aggressive behavior in dominating other CCAs. However, Google recently announced that Netflix is currently experimenting with BBR [9]. Although BBR is expected to make a major departure from traditional congestion-window-based CC; however, it does not harm Cubic's performance and allows Cubic to take most of the channel capacity across all traces. Figure 8a

[2]Note that negative delay harm indicates a reduction in Cubic delays.

shows that BBR does not have a strong impact on Cubic's throughput. With respect to delay, Figure 8b shows that BBR's impact is slightly higher, but is still below Cubic's self-inflicted harm. We further analyze the interaction between Cubic and BBR in Figure 8c, , where we observe that Cubic dominates the channel capacity, unfairly killing BBR flows in an "infinite buffer" setting.



(a) Throughput (5G city drive)

(b) Throughput (Street canyon)

(c) Delay (5G city drive)

(d) Delay (NS-3 street canyon)

Fig. 9. Effect of buffer size on Cubic and BBR inter-fairness

6.5 Impact of buffer sizes

To understand BBR's behavior when sharing a bottleneck link with Cubic and assess the impact of the buffer size on its performance, we experimented with different buffer sizes, varying from 1 to 15BDPs. Figures 9a and 9c show the throughput and delay inter-change between BBR and Cubic according to the bottleneck buffer size in the 5G city drive trace. The figures show that at approximately 4.5BDPs both algorithms reach a fair share of the capacity. On the other hand, Cubic claims more capacity upon increasing the buffer size, almost to the point of full dominance around 7BDPs and above. As for values below 4.5BDPs size, BBR dominates the performance leaving almost no share for Cubic to explore.

Similarly, for the street canyon results shown in Figures 9b and 9d, both BBR and Cubic reach a fair share at around 11.5BDPs before Cubic dominating the channel capacity with growing buffer sizes. We find two logical explanations for this behavior. First, BBR restricts the packets in-flight at a maximum of 2BDPs (the extra BDP deals with delayed/aggregated ACKs). As a result, this extra BDP of data in shallow buffers causes huge packet re-transmissions due to losses. BBR neglects loss as a congestion signal and maintains high re-transmissions over time, in turn worsening things. On the other hand, Cubic adjusts its `CWND` upon a loss. Therefore, BBR causes more packet transmission/re-transmissions than Cubic. This implies that BBR delivers high throughput in shallow buffers but at the expense of high packet re-transmissions. Second, BBR finds the max `target_CWND` as `CWND_gain x BtlBw x RTprop` and increases the window each time an ACK is received until the window reaches the `target_CWND`. Where BtlBw and RTprop are the estimated bandwidth and RTT, respectively. Every 10 secs, BBR probes for `RTprop` by reducing its in-flight

packets to just 4 packets to drain the queue. When BBR has an accurate estimate of `BtlBw` and `RTprop`, it caps its packets in-flight at 2BDPs, i.e., BBR allows just 1BDP worth of packets to queue at the buffer for 8 RTTs. Meanwhile, Cubic expands its `CWND` to fill the buffer before encountering loss. Since a flow's throughput is proportional to the buffer share, Cubic gets more packets queued. BBR observes a lower throughput and further decreases its `CWND`. This creates a positive feedback loop allowing Cubic to increase its `CWND` in response to BBR's `CWND` decrease.

6.6 Real World Performance

Under the hood, the primary metrics measuring network performance include throughput and latency. However, given the throughput and delay performance of a CCA, it is difficult to extrapolate and compare from these data points to the Quality of Experience (QoE) observed in real applications using these CCA in various network environments. To bridge this gap, we devise a simplified scenario-aware scoring framework that evaluates CCA performance in alignment with practical application demands. Each CCA instance is assessed using a set of normalized metrics: throughput utilization, delay inflation, intra-fairness, and inter-fairness. We define composite utility scores for three representative application scenarios by assigning weights to each metric based on its relative importance. These weights reflect how different classes of applications trade off between responsiveness, utilization, and fairness. They are motivated by a combination of industry standards [1], QoE modeling literature [10], and empirical studies of traffic behavior for these scenarios [6, 26, 29]).

Let x be a CCA instance with normalized metrics Thpt, Delay, IntraFair, InterFair. The following composite score functions are used:

- **Real-Time**: $Score_{RT}(x) = 0.4 \cdot Delay^{-1} + 0.3 \cdot InterFair + 0.2 \cdot Thpt + 0.1 \cdot IntraFair$ Real-time applications such as cloud gaming, AR/VR, and video conferencing are highly sensitive to queuing delay and jitter. To ensure responsiveness, delay inflation and inter-flow fairness are prioritized, with throughput weighted lower to reflect its secondary importance once basic video/audio quality thresholds are met.
- **Large Object Transfer**: $Score_{Bulk}(x) = 0.75 \cdot Thpt + 0.05 \cdot Delay^{-1} + 0.15 \cdot IntraFair + 0.05 \cdot InterFair$ For applications involving cloud backups, and file transfers, sustained throughput is the dominant concern, as buffers and retries can often absorb variable delays. Fairness remains important for consistency across flows, while latency is given moderate weight to discourage extreme delay inflation.
- **Mixed Traffic**: $Score_{Mixed}(x) = 0.20 \cdot Thpt + 0.20 \cdot Delay^{-1} + 0.30 \cdot IntraFair + 0.30 \cdot InterFair$ In enterprise or hotspot environments with diverse traffic types, balanced behavior is critical. Equal weighting reflects the need for protocols that are fair, stable, responsive, and capable of maintaining reasonable throughput under shared conditions.

Scenario	Allegro	BBR	Copa	Cubic	Ledbat	Proteus-P	Proteus-S	Reno	Verus	Vivace
Real-Time	0.510	0.594	0.707	0.588	0.457	0.451	0.377	0.575	0.473	0.555
Large Object Transfer	0.819	0.893	0.695	0.919	0.232	0.772	0.572	0.923	0.795	0.725
Mixed Traffic	0.637	0.688	0.782	0.731	0.574	0.606	0.551	0.719	0.619	0.664

Table 3. Scenario-based composite scores for each CCA

This scoring system computes average scores for each CCA based on the analysis results. These composite scores provide a scenario-aware ranking of protocol suitability, allowing us to link observed performance trends to real-world deployment implications. Table 3 presents the comparative results across the three application profiles. We observe that Copa is expected to perform well in

low-delay and high-fairness environments, aligning with the needs of interactive and real-time applications. In contrast, Cubic and Reno dominate in high-throughput scenarios due to their aggressive sending behavior, though their delay performance is poor. BBR emerges as a strong all-around performer, balancing throughput and delay effectively. Ledbat underperforms in high throughput scenarios due to conservative bandwidth usage. Proteus-S and Vivace also perform well in balanced settings, maintaining fairness and delay control without severely compromising throughput.

7 DISCUSSION AND KEY TAKEAWAYS

Our comprehensive evaluation presents an overwhelming set of data points across a diverse set of 5G conditions, CCAs, and buffer regimes. We distill these findings into several core insights that inform both practical deployment and future research directions.

Cubic and Reno, though dated, continue to perform well in terms of raw throughput, especially under moderate-to-large buffer conditions making them effective choices for throughput-intensive applications such as video streaming, cloud backups, and software updates. Notably, Reno sometimes outperforms modern algorithms in some 5G traces, particularly those with moderate BDP and relatively stable variation. Copa and Vivace are strong candidates for realtime, interactive applications such as AR/VR, cloud gaming, and video conferencing. However, while suitable for latency-critical workloads, they may be unsuitable for bandwidth-intensive tasks unless paired with adaptive tuning. Learning-based CCAs like Proteus and Verus are designed to adapt dynamically to network conditions, but their real world performance under 5G variability is inconsistent. Verus displays unstable delay and low throughput in some scenarios, and Proteus variants show erratic responsiveness that leads to both underutilization and high coexistence harm. These findings suggest that while learning-based designs hold promise, current approaches may require more robust mechanisms to handle the abrupt dynamics of mobile 5G networks. BBR and Allegro offer a balanced approach, making them suitable for mixed traffic environments such as enterprise WLANs, shared mobile hotspots, or urban 5G deployments. BBR maintains competitive throughput while moderating delay and limiting coexistence harm, making it a reasonable default for general purpose use. Allegro performs similarly well across most conditions, but like Copa, it occasionally underutilizes links in highly dynamic environments where capacity shifts faster than its reaction time. Both protocols stand out for their ability to navigate multiple performance trade-offs without leaning too heavily toward a single objective.

Across all scenarios, we consistently observe that protocol behavior falls along a clear throughput-delay trade-off curve. Each CCA implicitly chooses a point along this frontier: some favoring utilization at the cost of queuing delay, others sacrificing bandwidth for responsiveness. Tuning protocol parameters or varying buffer size shifts a CCA's position along this curve, suggesting that the set of achievable trade-offs is bounded. This insight reinforces the importance of designing CCAs that can dynamically reposition themselves along the frontier in response to shifting network or application constraints.

In conclusion, our detailed evaluation of ten different CCAs in 5G environments unearthed several limitations of individual protocols without a clear winner across all environments. Even the best performing protocol in specific 5G settings exhibits limitations across other 5G environments including serious fairness concerns when competing with other flows in 5G. Additionally, we present an evaluation framework for CCA to enable various types of CCA evaluations in a unified setting with access to standardized benchmarking that is directly comparable to other CCA evaluations using the standard framework. We believe that there still exists an essential gap in congestion control research in 5G and designing an optimal CCA to address the 5G challenge remains an open research question. Finally, these results highlight the broader value of Zeus as a benchmarking

platform. By enabling standardized and reproducible evaluations across diverse 5G scenarios, it could serve as a foundational tool for the design, testing, and selection of future congestion control algorithms that meet the demands of next-generation networks.

REFERENCES

[1] 3GPP. [n. d.]. TS 23.203: Policy and Charging Control Architecture. 3GPP Technical Specification. ..

[2] 5TONIC. 2022. *An Open Research and Innovation Laboratory Focusing on 5G Technologies.* Accessed: 19-05-2022.

[3] Soheil Abbasloo, Yang Xu, and H Jonathan Chao. 2019. C2TCP: A flexible cellular TCP to meet stringent delay requirements. *IEEE Journal on Selected Areas in Communications* 37, 4 (2019), 918–932.

[4] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. 2020. Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet. In *Proc. ACM SIGCOMM* (Virtual Event, USA). New York, NY, USA, 632–647. https://doi.org/10.1145/3387514.3405892

[5] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical delay-based congestion control for the internet. In *Proc. of NSDI*. 329–342.

[6] Y. Birk and D. Crupnicoff. 2003. A multicast transmission schedule for scalable multirate distribution of bulk data using nonscalable erasure-correcting codes. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, Vol. 2. 1033–1043 vol.2. https://doi.org/10.1109/INFCOM.2003.1208940

[7] Joe Breen, Andrew Buffmire, Jonathon Duerig, Kevin Dutt, Eric Eide, Mike Hibler, David Johnson, Sneha Kumar Kasera, Earl Lewis, Dustin Maas, Alex Orange, Neal Patwari, Daniel Reading, Robert Ricci, David Schurig, Leigh B. Stoller, Jacobus Van der Merwe, Kirk Webb, and Gary Wong. 2020. POWDER: Platform for Open Wireless Data-Driven Experimental Research. In *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization* (London, United Kingdom) *(WiNTECH'20)*. Association for Computing Machinery, New York, NY, USA, 17–24. https://doi.org/10.1145/3411276.3412204

[8] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: congestion-based congestion control. *Commun. ACM* 60, 2 (2017), 58–66.

[9] Neal Cardwell, Yuchung Cheng, S Hassas Yeganeh, Ian Swett, Victor Vasiliev, Priyaranjan Jha, Yousuk Seung, Matt Mathis, and Van Jacobson. 2019. Bbrv2: A model-based congestion control. In *Presentation in ICCRG at IETF 104th meeting*.

[10] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. 2018. Controlling queuing delays for real-time communication: the interplay of E2E and AQM algorithms. *SIGCOMM Comput. Commun. Rev.* 46, 3, Article 1 (July 2018), 7 pages. https://doi.org/10.1145/3243157.3243158

[11] Luis Diez, Alfonso Fernández, Muhammad Khan, Yasir Zaki, and Ramón Agüero. 2020. Can We Exploit Machine Learning to Predict Congestion over mmWave 5G Channels? *Applied Sciences* 10, 18 (2020), 6164.

[12] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira. 2015. PCC: Re-architecting congestion control for consistent high performance. In *Proc. of NSDI*. 395–408.

[13] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. PCC vivace: Online-learning congestion control. In *Proc. of NSDI*. 343–356.

[14] Fraunhofer FOKUS. 2022. *5G Playground.* Accessed: 19-05-2022.

[15] Prateesh Goyal, Anup Agarwal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. 2020. ABC: A Simple Explicit Congestion Controller for Wireless Networks. In *Proc. of NSDI)*. Santa Clara, CA, 353–372. https://www.usenix.org/conference/nsdi20/presentation/goyal

[16] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[17] Habtegebreil Haile, Karl-Johan Grinnemo, Simone Ferlin, Per Hurtig, and Anna Brunstrom. 2021. End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks. *Computer Networks* 186 (2021), 107692.

[18] Habtegebreil Haile, Karl-Johan Grinnemo, Simone Ferlin, Per Hurtig, and Anna Brunstrom. 2022. Performance of QUIC congestion control algorithms in 5G networks. In *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*. 15–21.

[19] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. 2012. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 253–264.

[20] Stephen Hemminger et al. 2005. Network emulation with NetEm. In *Linux conf au*, Vol. 844. Citeseer.

[21] Teerawat Issariyakul and Ekram Hossain. 2009. Introduction to network simulator 2 (NS2). In *Introduction to network simulator NS2*. Springer, 1–18.

[22] Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. 1984. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* 21 (1984).

[23] Haiqing Jiang, Zeyu Liu, Yaogong Wang, Kyunghan Lee, and Injong Rhee. 2012. Understanding bufferbloat in cellular networks. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*. 1–6.

[24] Tomoaki Kanaya, Nobuo Tabata, and Saneyasu Yamaguchi. 2020. A study on performance of CUBIC TCP and TCP BBR in 5G environment. In *2020 IEEE 3rd 5G World Forum (5GWF)*. IEEE, 508–513.

[25] K. Kondepu, F. Giannone, S. Vural, B. Riemer, P. Castoldi, and L. Valcarenghi. 2019. Experimental Demonstration of 5G Virtual EPC Recovery in Federated Testbeds. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 712–713.

[26] A. Kuzmanovic and E.W. Knightly. 2003. TCP-LP: a distributed algorithm for low priority data transfer. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, Vol. 3. 1691–1701 vol.3. https://doi.org/10.1109/INFCOM.2003.1209192

[27] Chih-Ping Li, Jing Jiang, Wanshi Chen, Tingfang Ji, and John Smee. 2017. 5G ultra-reliable and low-latency systems design. In *2017 European Conference on Networks and Communications (EuCNC)*. IEEE, 1–5.

[28] Yiqing Ma, Han Tian, Xudong Liao, Junxue Zhang, Weiyan Wang, Kai Chen, and Xin Jin. 2022. Multi-Objective Congestion Control. In *Proc. 17th EuroSys Conf.* (Rennes, France). Association for Computing Machinery, 218–235. https://doi.org/10.1145/3492321.3519593

[29] Jonathan Mace, Peter Bodik, Rodrigo Fonseca, and Madanlal Musuvathi. 2015. Retro: Targeted resource management in multi-tenant distributed systems. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation* (Oakland, CA) *(NSDI'15)*. USENIX Association, USA, 589–603.

[30] Vuk Marojevic, Ismail Guvenc, Rudra Dutta, Mihail Sichitiu, and Brian Floyd. 2020. Advanced Wireless for Unmanned Aerial Systems: 5G Standardization, Research Challenges, and AERPAW Experimentation Platform. *IEEE Vehicular Technology Magazine* PP (04 2020). https://doi.org/10.1109/MVT.2020.2979494

[31] Tong Meng, Neta Rozen Schiff, P Brighten Godfrey, and Michael Schapira. 2020. PCC proteus: Scavenger transport and beyond. In *Proc. ACM SIGCOMM*. 615–631.

[32] Marco Mezzavilla, Menglei Zhang, Michele Polese, Russell Ford, Sourjya Dutta, Sundeep Rangan, and Michele Zorzi. 2018. End-to-End Simulation of 5G mmWave Networks. *IEEE Communications Surveys Tutorials* 20, 3 (2018), 2237–2263. https://doi.org/10.1109/COMST.2018.2828880

[33] Ravi Netravali, Anirudh Sivaraman, Keith Winstein, Somak Das, Ameesh Goyal, and Hari Balakrishnan. 2014. Mahimahi: A Lightweight Toolkit for Reproducible Web Measurement. *ACM SIGCOMM* 44, 4 (aug 2014), 129–130. https://doi.org/10.1145/2740070.2631455

[34] Yong Niu, Yong Li, Depeng Jin, Li Su, and Athanasios V Vasilakos. 2015. A survey of millimeter wave communications (mmWave) for 5G: opportunities and challenges. *Wireless networks* 21, 8 (2015), 2657–2676.

[35] Andrea Pinto, Andrew Ashdown, Tanzil Bin Hassan, Hai Cheng, Flavio Esposito, Leonardo Bonati, Salvatore D'Oro, Tommaso Melodia, and Francesco Restuccia. 2023. Hercules: An Emulation-Based Framework for Transport Layer Measurements over 5G Wireless Networks. In *Proceedings of the 17th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*. 72–79.

[36] Dipankar Raychaudhuri, Ivan Seskar, Gil Zussman, Thanasis Korakis, Dan Kilper, Tingjun Chen, Jakub Kolodziejski, Michael Sherman, Zoran Kostic, Xiaoxiong Gu, et al. 2020. Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–13.

[37] George F. Riley and Thomas R. Henderson. 2010. The ns-3 Network Simulator. In *Modeling and Tools for Network Simulation*, Klaus Wehrle, Mesut Günes, and James Gross (Eds.). Springer, 15–34. http://dblp.uni-trier.de/db/books/collections/Wehrle2010.html#RileyH10

[38] Luigi Rizzo. 1997. Dummynet: a simple approach to the evaluation of network protocols. *ACM SIGCOMM Computer Communication Review* 27, 1 (1997), 31–41.

[39] Dario Rossi, Claudio Testa, Silvio Valenti, and Luca Muscariello. 2010. LEDBAT: the new BitTorrent congestion control protocol. In *2010 Proceedings of 19th International Conference on Computer Communications and Networks*. IEEE, 1–6.

[40] Chiranjib Saha and Harpreet S Dhillon. 2019. Millimeter wave integrated access and backhaul in 5G: Performance analysis and design insights. *IEEE Journal on Selected Areas in Communications* 37, 12 (2019), 2669–2684.

[41] Christopher Slezak, Menglei Zhang, Marco Mezzavilla, and Sundeep Rangan. 2018. Understanding end-to-end effects of channel dynamics in millimeter wave 5G new radio. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 1–5.

[42] Shu Sun, George R. MacCartney, and Theodore S. Rappaport. 2017. A novel millimeter-wave channel simulator and applications for 5G wireless communications. In *2017 IEEE Int. Conf. Commun. (ICC)*. 1–7. https://doi.org/10.1109/ICC.2017.7996792

[43] Andras Varga. 2010. OMNeT++. In *Modeling and tools for network simulation*. Springer, 35–59.
[44] Ranysha Ware, Matthew K Mukerjee, Srinivasan Seshan, and Justine Sherry. 2019. Beyond Jain's Fairness Index: Setting the Bar For The Deployment of Congestion Control Algorithms. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. 17–24.
[45] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Proc. of NSDI*. Lombard, IL, 459–471. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/winstein
[46] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. 2020. Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption. In *Proc. ACM SIGCOMM*. 479–494.
[47] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: the training ground for Internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 731–743.
[48] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive congestion control for unpredictable cellular networks. In *Proc. ACM SIGCOMM*. 509–522.
[49] Menglei Zhang, Marco Mezzavilla, Russell Ford, Sundeep Rangan, Shivendra Panwar, Evangelos Mellios, Di Kong, Andrew Nix, and Michele Zorzi. 2016. Transport layer performance in 5G mmWave cellular. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 730–735.
[50] Menglei Zhang, Michele Polese, Marco Mezzavilla, Jing Zhu, Sundeep Rangan, Shivendra Panwar, and Michele Zorzi. 2019. Will TCP work in mmWave 5G cellular networks? *IEEE Communications Magazine* 57, 1 (2019), 65–71.
[51] Tommaso Zugno, Michele Polese, Natale Patriciello, Biljana Bojović, Sandra Lagen, and Michele Zorzi. 2020. Implementation of a Spatial Channel Model for ns-3. In *Proceedings of the 2020 Workshop on ns-3*. 49–56.

A ETHICS

This work does not raise any ethical issues.