Paper under submission. Please do not distribute

Zeus: Can Current Congestion Control Algorithms Rise to the 5G Challenge?

Rohail Asim, *NYU Abu Dhabi*
Muhammad Khan, *NYU Abu Dhabi*
Luis Diez, *University of Cantabria*
Shiva Iyer, *NYU*
Ramon Aguero, *University of Cantabria*
Lakshmi Subramanian, *NYU*
Yasir Zaki, *NYU Abu Dhabi*

Abstract

As global cellular networks converge to 5G, one question lingers: Are we ready for the 5G challenge? A growing concern surrounds how well existing congestion control algorithms perform in diverse 5G networks. Given that 5G networks are not yet widely deployed, assessing the performance of existing congestion control algorithms in realistic 5G settings presents several challenges. Moreover, existing network simulation and emulation environments are also not ideally suited to address the unique challenges of 5G network environments. Therefore, building a simple and easily accessible platform becomes crucial to allow testing and comparison of congestion control algorithms under different testing conditions. This paper makes two main contributions. First, we present *Zeus*, an open-source testbed that emulates 5G channels to evaluate congestion control algorithms in a repeatable and reproducible manner. Second, using Zeus, we characterize the performance of ten state-of-the-art congestion control algorithms and demonstrate the strengths and limitations of these protocols under diverse 5G environments. In addition, using the recently proposed "harm metric", we perform a detailed characterization of these algorithms when sharing the network with other TCP flows.

1 Introduction

It seems impossible to evade the hype around 5G, which promises to create unprecedented user experiences, transform industries, and enrich lives [20]. However, such a transformational technology will require time to evolve and mature. Today, the 5G roll-outs by major cellular operators [10, 44] follow the pragmatic Non-StandAlone (NSA) mode, reusing legacy 4G infrastructure to reduce cost. That is, relying on the same underlying 4G radio technology to deliver higher data rates similar to those targeted by 5G, by using higher bandwidths rather than relying on newer access radio technology such as millimeter wave (mmWave) [39]. With 5G deployments, operators seek to achieve multi-Gbps wireless bit-rate for bandwidth-hungry applications such as 4K/8K Ultra High Definition (UHD) video and Virtual Reality (VR) transmission, ultra-Reliable and Low Latency Communication (uRLLC) for auto-driving, or telesurgery [13]. However, it is still too early to judge how long it will take for 5G to meet its full potential. A question that remains unanswered is *"How far is 5G from its prospects, and what are the missing pieces of the 5G puzzle?"*. For instance, the 5G New Radio (NR) has advanced swiftly between the 3GPP standardized Release 15 [32] and Release 16 (finalized) [31], enabling an extension to higher carrier frequencies to meet the continuous need for more traffic and higher data rates. However, the transport layer has not evolved at the same pace to address the challenges accompanied by 5G NR. This is the case for 5G millimeter-wave (mmWave) access technologies, which refer to higher spectrum bands in the range of 30 GHz and 300 GHz and exhibit substantial variations in their transmission capacity over short time scales. Such high fluctuations have led to several MAC/PHY layer solutions [33, 37, 41, 52]. However, lower-layer network innovations result in different forms of packet delay and rate variations over short time scales, introducing unexpected interactions with higher-layer Congestion Control (CC) algorithms. Moreover, today's Internet hosts an expanding oeuvre of CC algorithms, such as New Reno at Netflix [42], Copa at Facebook [3], and BBR [5] at Google. Hence, evaluating these state-of-the-art CC algorithms over the 5G environments is still an open and challenging task.

There are many factors that make the evaluation of such protocols in the wild a challenging task. These factors include the variability of the 5G environment due to external uncontrollable factors such as background competing network traffic, signal fluctuations due to noise, and the high-frequency nature of mmWave signals that leads to high propagation losses. Another important factor to consider is cost; a single minute experiment of a CC algorithm on 5G in the wild can consume up to 7 GB of data assuming a data rate of 1 Gbps. In addition, the variability of the 5G channel also means that it is crucial to repeat the experiment many times for a fair evaluation of the protocol that is statistically significant. These requirements

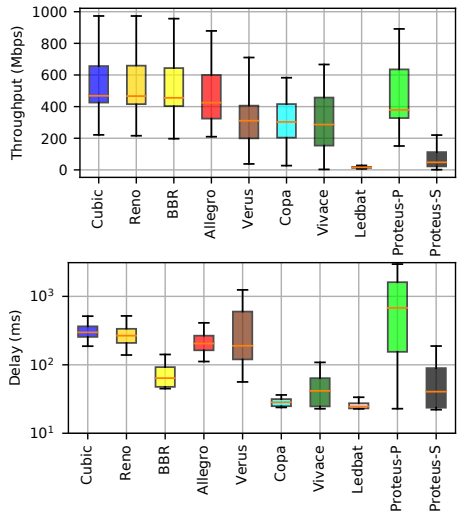


Figure 1: Summary of the CC algorithms performance on 5G

quickly drive up the cost and compromise the feasibility of such an experiment. Hence, researchers rely on alternative evaluation methods such as simulators or emulators to perform these evaluations. Unfortunately when it comes to 5G, there is a lack of widespread access to high-quality prototype environments, which limits the ability of researchers to test their protocols' designs in natural 5G environments. Even when such access is available [9], it suffers from reproducibility and control issues. In the context of 5G networks, the research community lacks an end-to-end framework that enables effective testing of a variety of state-of-the-art protocols, in a repeatable and reproducible manner, due to multiple factors [28]. First, the simulation environment needs to model realistic fine-grained bandwidth, loss, and buffer variations. Second, many newly proposed CC algorithms have tailored real-world implementations that lag in developing their corresponding simulation counterparts. Third, simulations and real-world evaluations may not always match due to protocol over-simplifications or experimental variations. This is due to the complexity of the lower-layer simulation models (especially PHY and MAC layers), whose behavior is modeled in the order of milli—and micro—seconds, while much longer simulations are required to evaluate CC algorithms (i.e., order of tens of seconds). Considering the challenges above, it is still necessary to understand the expected behavior of different CC algorithms over the potential scenarios brought by the 5G technology and analyze whether the current CC mechanisms can rise to the 5G challenge.

This paper presents *Zeus*[1], a framework that allows the emulation of realistic 5G channel traces in a repeatable and reproducible manner, allowing the evaluation of native implementations of CC algorithms over the *Zeus* 5G emulation environment. *Zeus* supports multi-Gigabit traffic rates using an enhanced implementation of the Mahimahi [29] link emulator. Repeatability and reproducibility of results in 5G evaluations is essential in order to derive accurate and meaningful conclusions as the variability of 5G render evaluating/comparing CC algorithms in the wild a challenging and expensive proposition.

Additionally, this paper also provides a detailed performance analysis of ten prominent CC algorithms over 5G scenarios, carried out using the *Zeus* framework. The above analysis includes both the single CC algorithm evaluations, and the interplay between different CC alternatives. Figure 1 shows an overall summary of the results—combined over 15 different 5G channel traces. The figure shows that TCP Cubic, Reno, and BBR [5] achieve the highest throughput compared to the rest of the protocols (top part of the figure); while the TCP variants experience very high end-to-end delays, BBR achieves a relatively lower end-to-end delay. The detailed analysis of the individual 5G channels is discussed later in Section 4. We summarize several interesting behavioral characteristics of these protocols that we observed: (i) despite the better performance of BBR, it is highly impacted when co-existing with Cubic, where its performance is dependent on the bottleneck's buffer size; (ii) Vivace [8] and Verus [49] exhibit high-performance variability over multiple runs under different 5G conditions; (iii) the throughput of delay conscious CC protocols, like Copa and Ledbat, is significantly affected in the process of holding delays at a specific threshold; (iv) Allegro [7] avoids Cubic's delay deficiencies and appears to provide higher throughput than competing solutions; however, it is still unable to achieve the full capacity of the 5G channels; (v) Proteus-P achieves high throughput but suffers from high end-to-end delays (similar to Cubic); (vi) online learning algorithms like Vivace and Proteus-S exhibit convergence issues due to the difficulties of identifying a better operating point in highly variable 5G environments; and (vii) Most CC algorithms do not easily co-exist in 5G environments with varying harm metric levels.

2 Challenges in 5G networks

This section briefly outlines specific challenges to developing an experimental environment for a realistic evaluation of CC protocols in 5G and beyond-5G networks.

Why is 5G different from 3G and 4G?
In order to get a better picture of the defining characteristics of 5G that make it unique from its predecessors thus posing new challenges, we have computed and compared multiple statistical metrics for each network technology over a wide variety of channel traces that we have measured in the wild. We used 16 different channel traces for each cellular technology. Each channel capacity was divided into several samples of one RTT size (i.e., 10ms)—given that the CC loop is usu-

[1]The framework will be open-sourced and released to the public once the work has been accepted to ensure the anonymity of the authors.

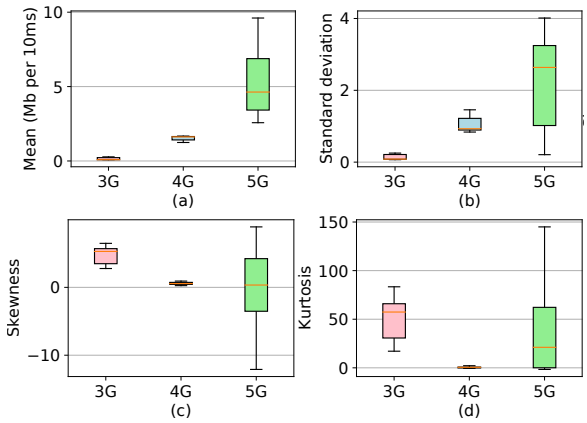


Figure 2: Cellular channels variability analysis

ally done over one RTT. As shown in Figure 2, we chose to compute the following metrics over the 10ms samples for each channel, which were: a) mean, b) standard deviation, c) skewness, and d) Kurtosis. Figure 2a shows the boxplots of the mean channels' capacity in Mega bits per 10ms time interval. It can be seen that 5G (depicted in green) has an order of magnitude higher mean compared to 4G and 3G. This is also the case when observing the standard deviation of the channels capacity shown (Figure 2b). As for the skewness, it can be observed from Figure 2c that the 3G channels are highly positively skewed given the higher skew values of greater than 1 (depicted in red). On the other hand, for 4G channels the results show that the channel capacity are moderately skewed to fairly symmetrical (depicted in blue). As for 5G, we can see high variability in skewness between the channels, where for some its highly positively skewed, while highly negatively skewed in others. In fact, negative skewness is only present in 5G compared to 3G and 4G. Finally, Figure 2d shows the kurtosis results, where again it can be seen that 4G has a low kurtosis indicating the lack of outliers in the channel capacity samples. On the other hand, 3G shows consistently high Kurtosis—which reflects the presence of a heavy tails or outliers. Finally, 5G again shows a wide variety of kurtosis values ranging from low values all the way to extremely high heavy tail. This analysis illustrates how the sheer magnitude of the average channel capacity and the bandwidth variation of the channel for 5G dwarfs the corresponding metrics for 3G/4G, which highlights interesting insights into how the variation of the channel predominates other metrics in terms of the effects of these characteristics on the different CC algorithms.

High bandwidth variations and loss rates: 5G networks can exhibit high bandwidth variations over short time scales, especially for high-frequency bands such as mmWave. Transmission rates reaching 8 `Gbps` [39] and 176 `Gbps` [11, 50] are observed within a 60 `GHz` frequency band in several scenarios. Moreover, mmWave is envisioned as a low-cost substitute for expensive fiber optic backhaul in cellular networks [23, 43], without compromising the multi-gigabit transmission rates [30]. mmWave signals suffer notable propagation losses when penetrating buildings or solid materials like walls. Besides the signal attenuation, the limited propagation distance can cause transmission outages or abrupt connectivity terminations when shifting from Line-of-Sight (LoS) to Non LoS (NLoS) conditions and vice versa [40] [17].

Short-term link interruptions: Repeated LoS-NLoS state switching in 5G networks can trigger link disruptions over short time scales. In 4G and 5G protocol stacks, the Hybrid Automatic Repeat reQuest (HARQ) [38] configured at the MAC layer, and the acknowledged and unacknowledged modes (AM/UM) [24] configured at the Radio Link Control (RLC), use re-transmissions to compensate for packet losses. However, they cannot compensate for link interruptions triggered by sudden NLoS transitions that occur over time scales larger than a few round-trip time (RTT) periods.

Buffer sizes and bloats: Configuring appropriate buffer sizes in 5G is not easy. A typical approach to compensate for packet losses is to over-provision the network buffers. Buffer sizes above one bandwidth-delay product (BDP) in 3G/4G are considered to mitigate packet losses [22]. Such oversized buffers also create buffer bloats, leading to increased delays [47].

3 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 3. The framework permits sending traffic using up to 10 different CC solutions (kernel-based or application-based) at high data rates under different network conditions. To emulate these scenarios and support such high data rates, the framework embeds a modified version of the Mahimahi [29] link emulator, which has been adopted by many to measure the performance of CC algorithms for various network conditions [2, 3, 8, 12, 48].

As for the channel traces, different sources can be used—as long as they follow the appropriate format—which is detailed in Section 3.2. Furthermore, the framework includes a set of traces obtained from three different sources: a real 5G cellular network of a commercial operator (Non-StandAlone (NSA)), a WiGig router connection (mmWave), and mmWave traces generated with an ns-3 network simulator. *Zeus* embeds a tailored version of ns-3 [35] to generate traces for mmWave scenarios, using the mmWave module [28] developed by NYU wireless. Figure 4 shows a sample recorded channel for each of the three recorded 5G channels. *Zeus* chooses the traces that are used to model the channel dynamics and correspondingly configures the Mahimahi link. The link emulator can be

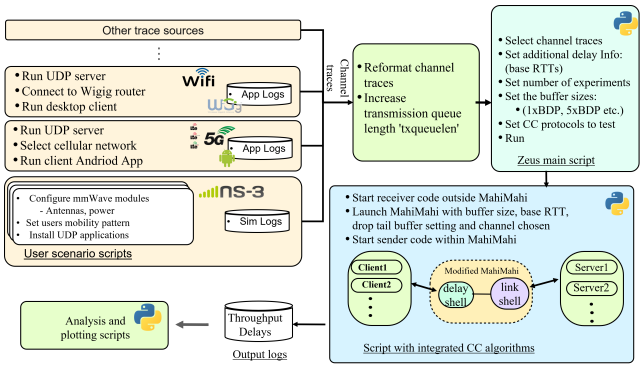
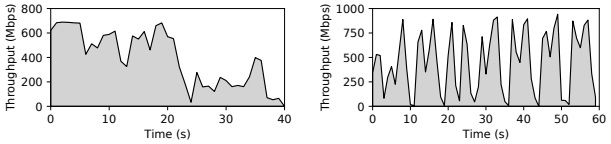
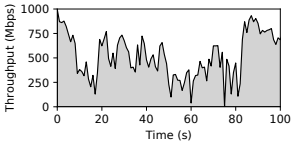


Figure 3: Architecture of the *Zeus* framework



(a) `5G city drive` (b) `WiGig walking`



(c) `Street canyon`

Figure 4: Evaluated 5G channel traces

tailored with additional end-to-end delay and/or modifying the buffer size of the link. Regardless of the chosen channel trace, Mahimahi is used by the framework to establish the link between end-to-end connections. The framework sets up sender/receiver pairs using a CC algorithm and connects them through Mahimahi. The sender/receiver pairs then start sending traffic, and the framework generates the result files, including: packet capture (PCAP) files, logs of sending and receiving events, and the evolution of buffer occupancy. Altogether, the output files allow us to compare the behavior of different CC algorithms over the same circumstances in a systematic manner that is repeatable and reproducible. Besides, *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.

In the following we detail the three most relevant aspects of the *Zeus* framework: 1) we describe the methodology of generating the real 5G traces[2], 2) we detail the changes that were required in Mahimahi to enable the multi-Gigabit transmission, and 3) we compare the performance of *Zeus* with that obtained over a real 5G network under similar circumstances.

[2]The reader may refer to Appendix A for a detailed explanation of the methodology for generating the Wigig and ns-3 traces.

3.1 Real 5G celullar network traces

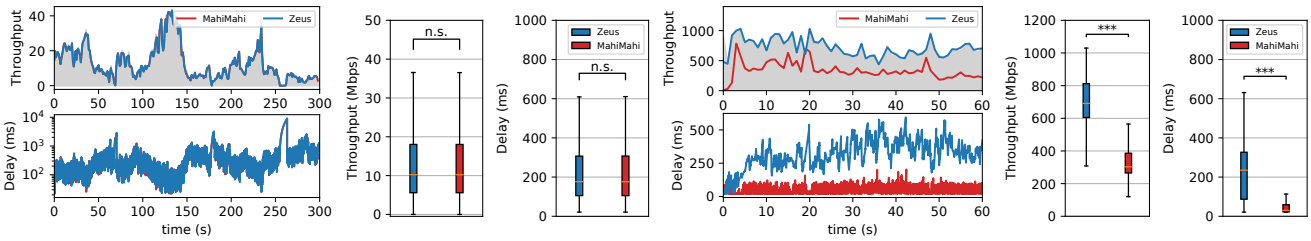
For the generation of real 5G network channel traces, *Zeus* offers a client and server-side implementation that can record real-time 5G channels in a cellular environment. However, for the generation to be successful, the server must have a Gigabit upload capability. Usually, Internet Server Providers (ISPs) advertise Internet plans, with download speeds reaching up to Gigabits per second (Gb/sec). However, the upload data rate is often capped at a much lower threshold. This threshold varies with regions and ISPs that are available in the area. Connecting the server to the ISP's backbone network via a fiber link with Gb/sec upload and download capacity is recommended. The client consists of an Android application that communicates with the server using the server's IP and port number. Upon connecting, the Linux-based server begins sending UDP packets with a maximum transmission unit (MTU) size of 1500 bytes at a constant data rate ($\geq$ Gb/sec) to the IP address of 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.

We recorded several real 5G network channel traces from a commercial cellular operator by considering multiple diverse scenarios including: an indoor and outdoor car park, a slow drive in the city, a walk along the beach, and sitting in a public park. In this paper, we refer to these collected 5G cellular traces as: `Indoor car park`, `Outdoor car park`, `City drive`, `Beachfront walking`, and `Public park`, respectively. On average, each trace was recorded for 60 seconds and consumed up to 6 GigaBytes (GB) of cellular data.

3.2 Modified Mahimahi Emulator

Among the different functionalities of Mahimahi, we mainly use its capability to replay channel traces (i.e., `mm-link` tool). However, the original Mahimahi implementation has certain 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. The developers acknowledge that Mahimahi is not tested beyond 100 `Mbps`. Any trace exceeding the threshold of $\approx$ 400 `Mbps` causes Mahimahi to drop packets randomly, thus capping the channel throughput and utilization.

To address this issue, we made several modifications, some after discussions with the original MahiMahi developers. The most significant modification was reducing the number of `read` and `write` system calls. Mahimahi's `mm-link` program simulates a channel based on a trace file, which contains one number per line. Each number indicates the millisecond from the channel's start time to accommodate a packet transmission. Thus, the trace file shows the channel's capacity regarding



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

Figure 5: Benchmarking *Zeus* against Mahimahi on 4G and 5G channels

the number of packets accommodated every millisecond. The `mm-link` program simulates the channel by holding each incoming packet and releasing them at the specified millisecond mark in the trace file. The original Mahimahi makes a call to `read` and `write` for each packet. We speed this up separately for the `read` and `write` calls. For the `write`, all packets are read in every millisecond from the trace file, and we make a single call to the `write` when writing to the internal socket. For the `read`, we make the socket non-blocking and read in as many available packets in a single instance before buffering them.These modifications significantly increased the maximum throughput in Mahimahi, allowing *Zeus* to run experiments over multi-Gigabit capacity traces.

Figure 5 shows a performance comparison of *Zeus*'s Mahimahi to the original Mahimahi implementation. The results are obtained by having a utilizing a Cubic flow with bulk full-buffer transmission during the whole experiment duration. Figures 5a and 5b show the results obtained when a 4G Verizon-LTE [47] trace was used, with a maximum capacity around `40Mbps`. Figures 5c and 5d show the performance obtained over a synthetic 5G channel generated in ns-3 using Indoor Hotspot Model `(InHM)`, with capacity reaching `1Gbps`. Both channel capacities are shown in Figures 5a and 5c 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 (please refer to Section 4(iv) for how the statistical significance was computed). 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 CC protocol in use. This is done by significantly lowering the required inter-process events for such high-bandwidth 5G traces. A considerable statistical difference for both throughput and delay is observed as can be seen in Figure 5d.

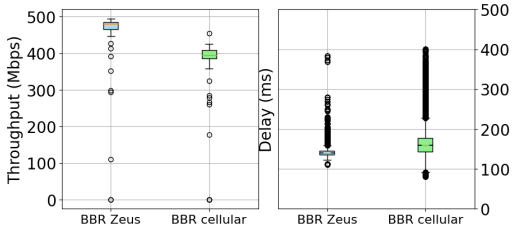


Figure 6: *BBR* in a real 5G cellular connection vs. *Zeus*

3.3 *Zeus*' operation validation

We validate the correct operation of *Zeus* by comparing its performance with that obtained in a real 5G environment. In particular, in Figure 6 we represent the statistical distribution of the delay and throughput with whisker plots when using *BBR* as a CC algorithm. The delay is measured as the time elapsed since a packet leaves the sender until an Acknowledgement (Ack) is received back at the sender, confirming the packet delivery. In contrast, the throughput is periodically measured every second, taking the number of bits received within the last second. Figure 6 shows the results obtained from three 5G cellular connections, each lasting 60 seconds, using *BBR* during the connections, and results obtained by *Zeus* for *BBR* over similar channel traces recorded at the exact location where the three 5G cellular connections are created.

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*.

4 Evaluation

Our evaluation aims to explore the performance of state-of-the-art CC protocols from the following perspectives:

(i) Diverse 5G networks: Initial 5G roll-outs heed an NSA flavor of 5G built on existing 4G-LTE Evolved Packet Core (EPC). It is an upgrade to quickly introduce partial 5G services while maximizing the reuse of existing 4G networks, helping Mobile Network Operators (MNOs) from suffering economic impediments. Moreover, the adoption of mmWave into 5G systems is an essential component currently missing in the 5G puzzle. Although 3GPP has already finalized 5G NR Release 16, the actual deployment of mmWave technology making its first public debut in 5G is still unknown. Therefore, we include three broad types of networks in our evaluation: a) an existing NSA 5G cellular network, b) a physical 5G WiGig router for a 5G mmWave network, and c) an ns-3 mmWave network. The *Zeus* toolkit is packed with various channel traces that we recorded over diverse 5G scenarios. Due to space limitations, we selected three distinct channel traces obtained from a real 5G cellular network (5G city drive), a physical 5G WiGig router (WiGig walking), and a simulated mmWave ns-3 scenario (street canyon) for an in-depth performance evaluation. Figure 4 illustrates the selected 5G traces. However, we have also evaluated the CC algorithms on four additional diverse channels for each category (i.e., total of 15), encompassing a wide range of user mobility (stationary, walking, city and highway driving, etc.) and locations (indoor, outdoor, malls, beach, highway, etc.). The results of these traces can be found in Appendix B.

(ii) End-to-end throughput and delay: It is sensible to question the compatibility of existing CC algorithms with the characteristics of 5G networks, hence their performances may encounter the attrition factors highlighted earlier in Section 2.

(iii) Harm analysis of CC protocols: It has become a norm in the existing literature to assess the fairness of new CC algorithms towards existing legacy CC solutions while sharing a bottleneck bandwidth. Several papers studying CC exploit Jain's fairness index to ascertain this aspect. However, unlike fairness or friendliness to legacy TCP, in our analysis, we adopt a more practical harm-based approach (defined in [46]), which aims at identifying whether a new CC algorithm causes more harm to the performance of an existing legacy CC protocol when sharing a 5G bottleneck than the legacy protocol causes to itself. This rationale is to ensure that new techniques do not cause more harm than existing solutions. Taking Cubic as our default TCP flavor[3], we measure the harm caused to Cubic flows by other competing flows of SoTA CC protocols. To identify if a new protocol is eligible to co-exist with Cubic in the wild, it must not damage Cubic flows beyond the damage Cubic causes to itself. In simple words, the maximum harm allowed for a new CC protocol to be deployed alongside Cubic should not surpass Cubic's self-harm.

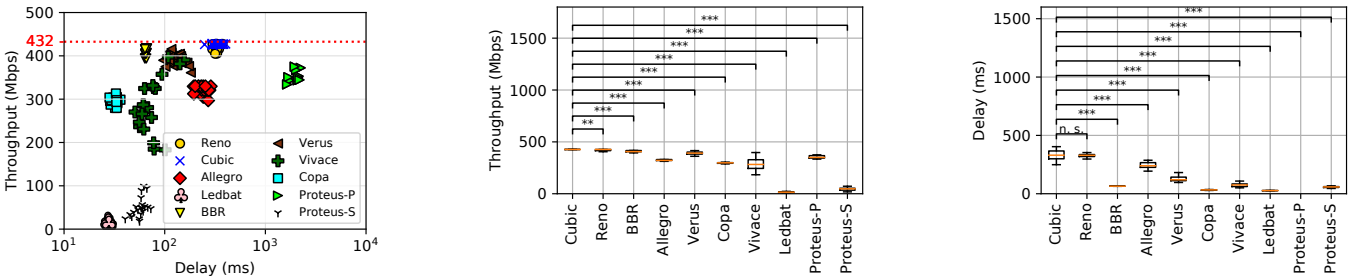
(iv) Statistical Significance: Statistical significance establishes how confident we are that the difference or relationship between two random variables does not exist by chance. When a result is identified as statistically significant, there is an absolute difference or relationship between two variables, and it is unlikely that it is a one-off occurrence. In our case, we are interested in measuring how different CC algorithms are. More specifically, two variables are statistically significant when their p-value falls below a certain threshold, called the *significance level*. We represent the level of significance with stars. If the p-value is less than 0.05, the level of significance is flagged with one star (*). If a p-value is less than 0.01, it is flagged with two stars (**). If a p-value is less than 0.001, it is flagged with three stars (***). Finally, if the p-value is larger than 0.05, then there is no statistical significance between the two variables, and it is flagged with (n.s.). Our evaluation identifies the statistical significance in terms of throughput and delay of CC algorithms compared to Cubic.

Setup: All experiments are set with a propagation delay of `10ms`. The experiments are conducted on a customized server with an Intel Xeon Bronze 3204 CPU @ 1.90GHz × 12, a 15 GiB memory, and an Ubuntu 20.04.3 LTS Operating system. The experiments are performed with a bftpd server and an FTP client, requesting a large 20 GB file from the server using a given CC algorithm. For each CC algorithm-channel tuple, we ran 20 independent experiments. We also had a discussion with some of the authors of these CC algorithms to verify the configuration and behavior of their protocols in order to ensure the accuracy of the evaluation.

4.1 End-to-end throughput and delay analysis

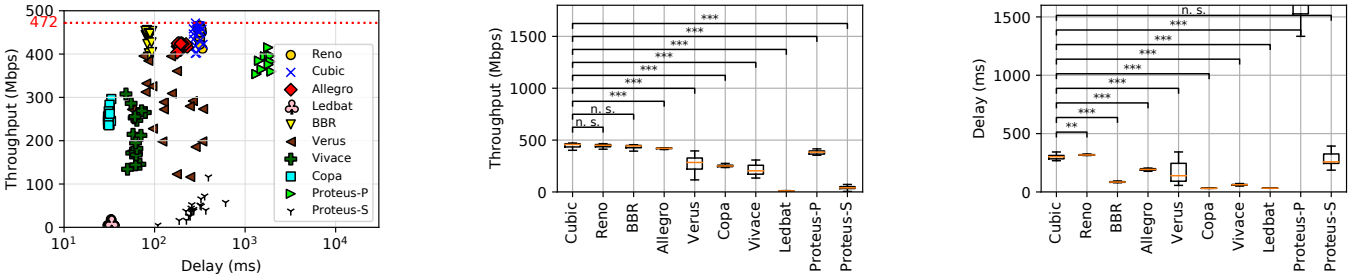
Legacy CC algorithms' performance (Cubic & Reno): Figures 7a, 8a, and 9a present scatter plots (throughput vs. delay) of ten SotA CC algorithms over a real 5G cellular trace, a physical WiGig router trace, and the ns-3 street canyon trace, respectively. Each marker point represents the average throughput (y-axis) and delay (x-axis) performance for a CC algorithm of a single run. For each CC algorithm, we conducted 20 separate runs per channel trace with an infinite buffer setting in *Zeus* to ascertain the stability of protocols' performance and how efficiently they can utilize the 5G capacity. We observed that, across all three 5G traces, Cubic and Reno managed to saturate the channel capacity (the average channel capacity is shown with a red dotted horizontal line). However, in comparison, both Cubic and Reno account for the highest delays among almost all other CC algorithms (apart from Proteus-P). We observe no statistical significance (n.s) in terms of throughput between Cubic and Reno for the WiGig walking and street canyon traces (Figures 8b and 9b, respectively). For the 5G city drive trace, the throughput

[3]We choose Cubic as our baseline protocol since it is the default TCP flavor in many of today's platforms such as Linux [47], or Android OS [1]



(a) Scatter comparison (b) Throughput stat. significance (c) Delay stat. significance

Figure 7: 5G city drive CC protocols comparison



(a) Scatter comparison (b) Throughput stat. significance (c) Delay stat. significance

Figure 8: WiGig walking CC protocols comparison

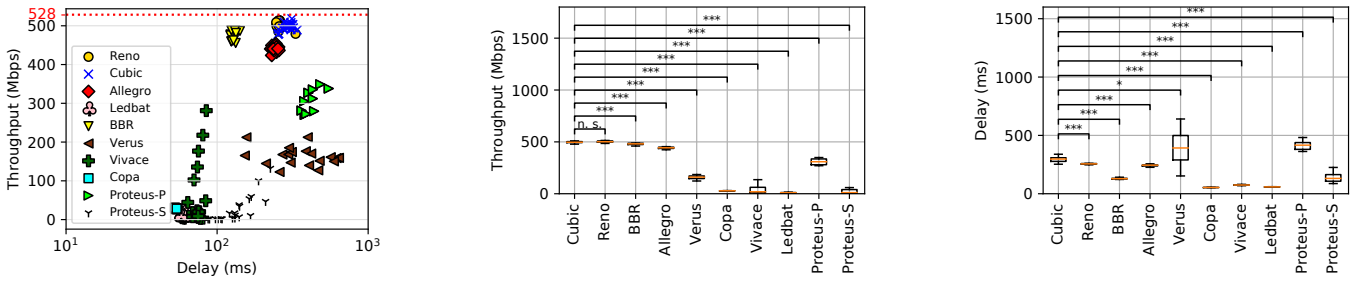
comparison is flagged with two stars, highlighting a medium statistical significance in the results. In contrast, there is no statistical significance in delay for the 5G city drive trace (Figure 7c). For delay comparison of the WiGig walking and the street canyon traces, we observed a statistical significance in the results of the two protocols, highlighted by the two and three stars, respectively (Figures 8c and 9c).

Google's BBR performance: BBR shows very stable performance across all traces, almost fully saturating the channel capacity. Particularly, its throughput remains slightly lower than the average channel throughput, with 94% utilization over the 5G city drive trace and 92% utilization for both the WiGig walking and street canyon traces. 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 by one-half, one-fifth, and one-third times, compared to both Cubic and Reno for the WiGig walking, 5G city drive, and street canyon traces, respectively. Looking at the statistical significance, it can be seen that for both the 5G city drive and the street canyon traces, there is a statistical significance (***) in the results compared to Cubic (Figures 7b, and 9b). However, for the WiGig walking trace, no statistical significance of throughput is observed between BBR and Cubic (Figure 8b). On the other hand, BBR's delays are of statistical significance (***) when compared to Cubic, across all traces (Figures 7c, 8c, and 9c), which highlights their different expected behavior in terms of delay in all scenarios.

Facebook's Copa performance: Copa maintains a stable behavior for all traces, although its performance is degraded in terms of throughput compared with the aforementioned alternatives. Copa's throughput varies across the different channels despite keeping the delays at a minimum. As seen in Figure 9a in the street canyon trace, Copa explores only 5% of the available channel capacity, whereas it achieves a 68% and 54% utilization for the 5G city drive and WiGig walking trace, respectively. Copa's performance is statistically significant compared to Cubic, for both delay and throughput.

The PCC Family performance: Allegro achieves lower delays than Reno and Cubic but higher delays than BBR. Apart from the 5G city drive trace, Allegro's throughput performance is higher than all CC protocols apart from BBR, Cubic, and Reno. In all cases, Allegro maintains its delay within the `200-300ms` range . As for Vivace, it shows steady results in terms of delay, although it exhibits a highly variable throughput in independent experiments over the same channel. In Figure 7, we observe that Vivace achieves a high throughput, whereas it's not able to explore the capacity in other scenarios.

Following a scavenger strategy, Proteus-S is not able to adapt to the variability of the analyzed channels, reaching only 3% of the available capacity. In contrast, the bandwidth utilization of Proteus-P tends to be high (comparable to Cubic and Reno), but it also suffers from the same problem of having difficulty dynamically adapting to the highly variable channels which leads to very high delays. Allegro and Vivace



(a) `Scatter comparison` (b) `Throughput stat. significance` (c) `Delay stat. significance`

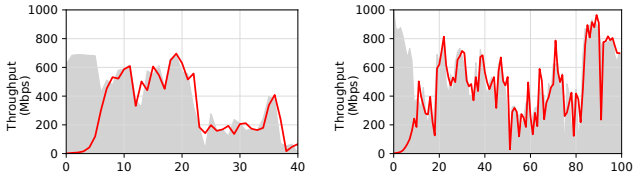
Figure 9: Street canyon CC protocols comparison

have a high statistical significance for delay compared to Cubic across all traces, given their substantially lower delays. However, this comes at the expense of throughput, with Cubic results being statistically significant. For Proteus-S, there is no statistical significance in the delays for the WiGig trace.

Others: Observing the Verus results, it exhibits scattered throughput and delay values across all traces, making it an unpredictable protocol in 5G environments (specifically in the WiGig and street canyon traces). Similarly, Ledbat, which adopts a scavenger strategy like Proteus-S, fails to saturate the link capacity, achieving a low throughput in all traces.

4.2 Why are the algorithms behaving this way?

Both Cubic and Reno are known for building large sending windows, resulting in unnecessary delays due to over buffering packets within the networks. Such behavior often leads to a well-known phenomenon called "bufferbloat" in cellular networks and has been well investigated in the context of 3G/4G cellular networks [22]. We observe the same behavior with an infinite buffer in the 5G network, where both Cubic and Reno rev the sending rate saturating the available channel capacity resulting in higher delays. Moreover, 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, and it appears to be the cause of the slight under-utilization of the available channel. Copa also relies on queuing delays for CC and strives to minimize the packet delays by periodically draining the buffers. However, Copa's effort to maintain a minimum queuing delay results in a significant throughput degradation in 5G networks. Allegro's operates in a fixed-step increment or decrement when adjusting the sending rate. We observed that these steps may be too small in a 5G network setting. As observed in the instantaneous throughput plots given in Figures 10a, and 10b, Allegro seeks to saturate the link capacity in steps at the startup phase. However, when the sending rate becomes significantly higher than the channel capacity, Allegro drops its sending rate in fixed increments until it reaches the bandwidth threshold again. This fixed-step sending rate limits Allegro's ability to achieve higher channel utilization and comes at the cost of increased delays. As alluded to by the authors of [8], Vivace does not perform well in a highly dynamic network such as cellular networks. Vivace is a learning-based CC algorithm using online greedy optimization methods to control its sending rate via a utility function. However, optimization methods are easy to trap in a local optima [26], which is why we see a different 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 due to drastic changes in available bandwidth.



(a) `5G city Drive` (b) `street canyon`

Figure 10: Allegro's inst. throughput across sample traces

Proteus-S and Proteus-P are built atop an online learning CC framework [7]. Proteus-S, with its scavenger utility, controls the sending rate and leverages delay variation as a sensitive early indication of flow competition. Despite having a strategy to compensate for misleading delay variation in a non-congested channel (which is a highly likely case in 5G), it still exhibits low performance. We think this is due to the uncertainty of the learning-based components and the difficulties of identifying the change of the equilibrium points. On the other hand, Proteus-P controls its sending rate using a modified version of the utility function of Vivace (with negative RTT gradients ignored). These modifications allow Proteus-P to outperform Vivace in terms of throughput but at the cost of significantly higher delays in 5G.

Ledbat tries to keep delays at a predefined target value (default=25ms), and stays at the value to form a steady-state and never add more than a target delay in queuing. Ledbat's congestion signal is RTT exceeding a target threshold. This explains why Ledbat achieves a low throughput and delay across all channel traces. Whereas Verus achieves variable performance in the 5G scenarios (even on repeated experiment on the same channel). We attribute this to Verus's delay profile curve, which is the basis for adjusting the CWND. From the results, it seems that Verus learns a different delay profile curve in each experiment. Despite the fact that Verus adjust the delay curve over time, it seems that it's not fast enough to catch up with the highly variable 5G channels.

4.3 Harm Analysis

A common requirement for any new CC protocol 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 [3]. In addition, Jain's Fairness Index [21] used to measure fairness assigns an equal score in both cases, i.e., whether a new CC protocol utilizes a larger share of the bottleneck bandwidth than the legacy Cubic or vise-versa. Therefore, inspired by [46], we follow a harm-based approach in this analysis to identify the damage/harm the CC algorithms do to Cubic when they co-exist over the 5G channels. We chose BBR, Reno, and Copa for our analysis because of their deployment in today's most used and popular services: Google, Netflix, and Facebook. We also considered selecting Allegro and Verus based on our end-to-end throughput and delays analysis. Following the harm definition in [46]), we define x= solo performance; and y = performance after introduction of a competitor connection. Then for metrics where 'more is better' (e.g., throughput) the harm = $\frac{x-y}{x} * 100$. On the other hand, for metrics where 'less is better' (e.g., delay) harm = $\frac{y-x}{x} * 100$.

We conducted 20 experiments with an "infinite buffer" setting 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., a baseline). This threshold shall provide a firm definition for when a CC protocol can be deployed alongside Cubic (i.e. it should not cause more harm than Cubic itself) and when it is not. We represent this threshold in the horizontal dark green areas shown in Figures 11a, 11b, 12a, 12b, 13a, and 13b. The green area represents the safe zone reflecting the acceptability of the protocol alongside Cubic when sharing a bottleneck. In contrast, the red part reflects the zone where the protocol is not suitable for deployment alongside Cubic. We observe that Cubic causes 50% throughput-harm to itself, reflecting an equal sharing behavior when competing with another Cubic flow[4].

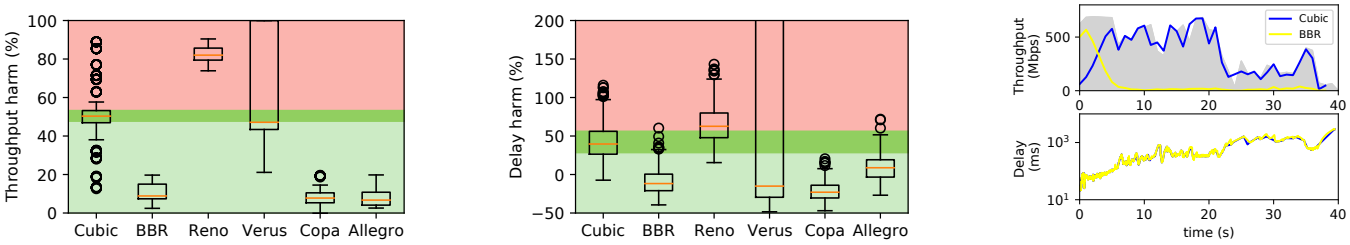
[4]Note that negative delay harm indicates a reduction in Cubic delays, whereas a positive delay-harm shows an increase in delay.

In all traces, we observe that BBR, Allegro and Copa fall in the green zone not doing much throughput or delay harm to Cubic. However, Reno is the only CC that always falls into 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 in the WiGig walking trace, and 55% throughput-harm and nearly 100% delay-harm in the street canyon trace. This could also be why Netflix adheres to Reno due to its aggressive behavior in dominating other CC algorithms. However, Google recently announced that Netflix is currently experimenting with BBR [6]. 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. Figures 11a, and 12a show that BBR does not have a strong impact on Cubic's throughput. In the case of the mmWave NS-3 channel, Figure 13a shows a slight impact of BBR over Cubic, but again below Cubic's self-harm threshold. With respect to delay, Figures 11b, 12b and 13b, shows that BBR's impact is slightly higher, but in all cases it is still below the Cubic's self-inflicted harm. We further analyze the interaction between Cubic and BBR in Figures 11c, 12c and 13c, where we observe that Cubic dominates the channel capacity, unfairly killing BBR flows in an "infinite buffer" setting.

4.4 Impact of buffer sizes

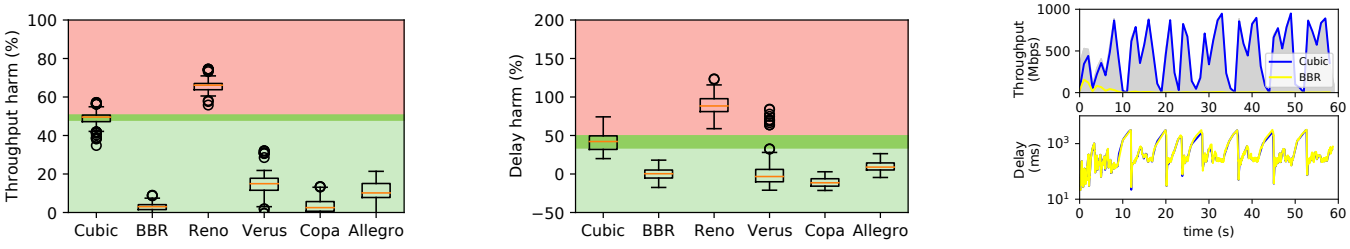
To understand BBR's behavior when sharing a bottleneck link with Cubic and assess the impact of the buffer size on its performance, we experimented with different buffer sizes, varying from 1 to 15 BDPs. Figures 14a and 14d show the throughput and delay inter-change between BBR and Cubic according to the bottleneck buffer size in the 5G city drive trace. The figures show that at approximately 4.5xBDP both protocols 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 7xBDP and above. As for values below the 4.5xBDP size, BBR dominates the performance leaving almost no share for Cubic to explore.

Similarly, for the street canyon trace results shown in Figures 14c and 14f, both BBR and Cubic reach a fair share at around 11.5xBDP 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 2xBDP (the extra BDP deals with delayed/aggregated ACKs). As a result, the 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 detecting a loss. Therefore, BBR causes more packet transmission/re-transmissions than Cubic. This also implies that BBR delivers high throughput in shallow buffers but at the expense of



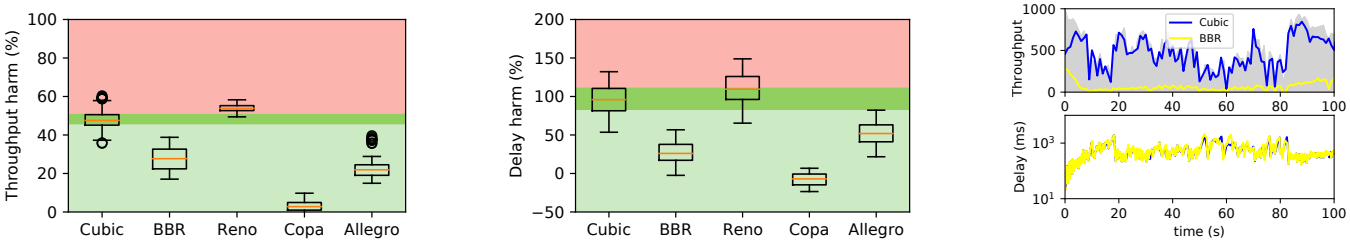
(a) Throughput harm to Cubic (b) Delay harm to Cubic (c) Instantaneous (Cubic/BBR)

Figure 11: Comparison of CC protocols' performance over real 5G city driving channel trace with infinite buffer.



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

Figure 12: Comparison of CC protocols' performance over WiGig walking trace with infinite buffer.

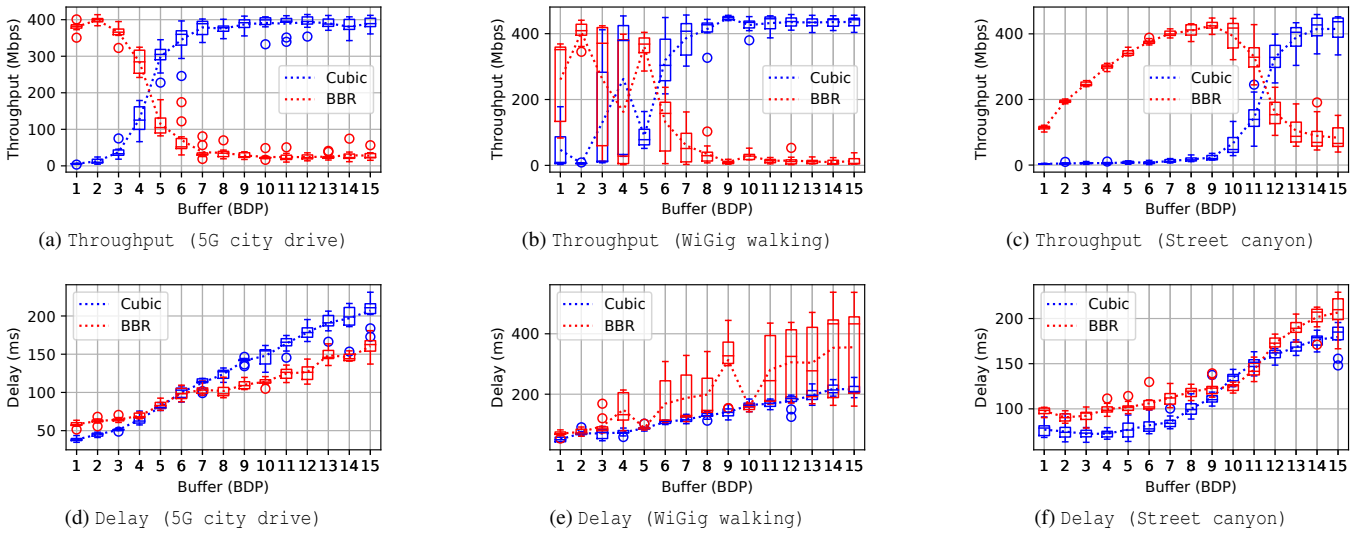


(a) Throughput harm to Cubic (b) Delay harm to Cubic (c) Instantaneous (Cubic/BBR)

Figure 13: Comparison of CC protocols' performance over 5G street canyon trace with infinite buffer.

high packet re-transmissions. Second, BBR finds the maximum `target_CWND` as `CWND_gain x BtlBw x RTprop` and increases the congestion window each time an ACK is received until the window reaches the `target_CWND`. Where BtlBw and RTprop are the estimated bottleneck bandwidth and round-trip propagation delay, respectively. Every 10 secs, BBR probes for `RTprop` by reducing its in-flight packets to just four packets to drain the queue. When BBR has an accurate estimate for the `BtlBw` and `RTprop`, it caps its in-flight packets at `2xBDP`. This means that BBR allows just 1 BDP of packets to queue at the bottleneck 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 thus further decreases its `CWND`. This creates a positive feedback loop allowing Cubic to continue its `CWND` increase in response to BBR's decrease in `CWND`.

Figures 14b and 14e, show a different behavior over WiGig trace. Given that Cubic is a loss-based CC, it tends to saturate the bottleneck buffer up to exhaustion, regardless of its size, whereas BBR restricts the in-flight data to `2xBDP`. This means that the larger the bottleneck buffer, the bigger share for Cubic. However, since Cubic builds long-lasting standing queues, a competing BBR flow may not acquire the true RTT_{min} during its `PROBE_RTT` phase. During the `PROBE_RTT` phase, BBR significantly reduces its in-flight packets to drain the queue; however, Cubic does not. Thus the bottleneck buffer is not drained completely, which makes BBR assume a higher RTT_{min} and thus, it increases its in-flight cap to a larger value. Occasionally, BBR gets a nearly fair share of the bottleneck bandwidth. At other times, most of the bottleneck bandwidth is occupied by Cubic. This highlights that BBR might not always be an appropriate choice for 5G. Additionally, from the above results, there is no clear pattern on what exact buffer size should be maintained for BBR to be harmless to Cubic since we have a different cut-off size for each trace.



(a) Throughput (5G city drive) (b) Throughput (WiGig walking) (c) Throughput (Street canyon)

(d) Delay (5G city drive) (e) Delay (WiGig walking) (f) Delay (Street canyon)

Figure 14: Effect of buffer size on Cubic and BBR sharing a bottleneck across the 5G channel traces.

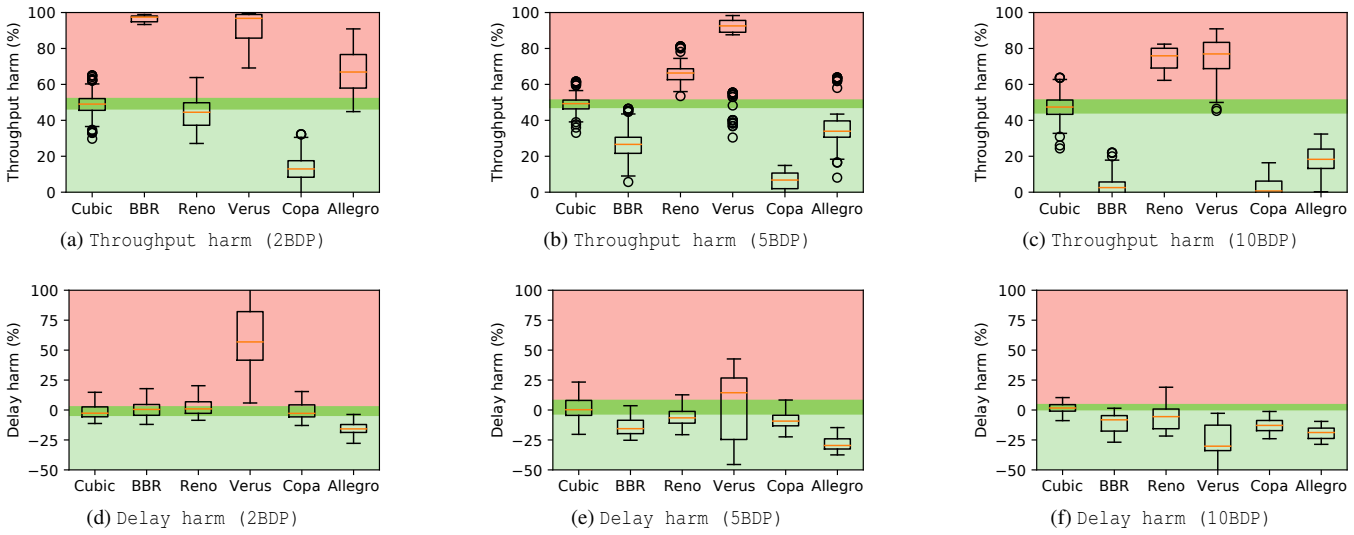
4.5 Harm analysis with different buffer sizes

Figure 15 shows the harm analysis of the CC algorithms chosen earlier with respect to Cubic in a shallow, medium and large buffer setting i.e., 2xBDP,5xBDP, and 10xBDP, over the 5G city driving trace. In the interest of saving space, we omitted the results of the other two traces given their similar trends (see Appendix C). We observe that in a shallow buffer 2xBDP, BBR, Verus, and Allegro fall in the red zone harming Cubic in terms of throughput. We have noticed how BBR and Cubic co-exist in different buffer settings in Figure 14. In a shallow buffer of 2xBDP and 5G city drive trace, Reno equally shares the bandwidth with Cubic with no harm done to Cubic's throughput and delay. However, with 5xBDP and 10xBDP buffers Reno harm to Cubic increases to 65% and 75% in terms of throughput while Cubic delays are reduced by 10% based on the observed (-ve 10%) delay-harm. Verus, with its uncertain behavior, harms Cubic above 95% in a shallow buffer and 80% in a 5xBDP setting in terms of throughput. Copa remains in the green zone for all traces, and buffer sizes regarding throughput and delays harm due to its low throughput performance. Allegro appears to be aggressive to Cubic in shallow buffers with nearly 75% throughput harm.

5 Related work

Network Simulators and Emulators: There is a wide variety of literature available on different simulation and emulation frameworks for cellular networks. Research on congestion control has long used network simulators like Network Simulator version-2, version-3 (NS-2, NS-3) [19, 35] and OMNeT++ [45], which can be used to virtually model any network. Moreover, real-time emulators like Dummynet [36], NetEm [18], Mininet [16], and MahiMahi [29] are broadly utilized for congestion control research. These run in real-time, limiting the scale of experiments to whatever the underlying hardware provides, while they exhibit adequate parameters and mechanisms to recreate complex network responses.However, setting up the emulator with realistic parameters to match the settings used in commercial networks is an open problem. The Pantheon project [48] is the most similar to *Zeus*. However, this project focuses more on live-testing and is less flexible for customized scenarios, specifically for 5G mmWave, hindering repeatability. Although Pantheon also provides MahiMahi as a tool for emulation, it does not support capacities above a specific limit ($\approx$ 500 Mbps). Moreover, the authors announced to no longer maintain Pantheon.

Congestion Control: Many CC protocol designs are based on assumptions about cellular bottlenecks and cross-traffic, and target specific scenarios that often restrict their deployment. Traditional TCP variants (e.g. [4, 14, 25]) fail to deliver consistent high performances [3, 5, 49] in wireless networks because they consider packet loss as a sign of congestion, which may not be accurate in wireless environments. Numerous CC protocols are proposed to address TCP's issues, which can be categorized as follows. The performance-oriented congestion control (PCC) family has contributed several CC protocols, leveraging a sending rate control logic based on an empirical utility function, such as PCC-Allegro [7] and PCC-Vivace [8]. One of the most recent proposals is PCC-Proteus [27], which has a primary mode that prioritizes throughput (Proteus-P), a scavenger mode (Proteus-S), and a hybrid mode (Proteus-H) with a piecewise utility function that switches between the primary and scavenger modes using an adaptive threshold based on the application requirements. Recently, Google presented their BBR algorithm [5, 6] showing good performance over



(a) Throughput harm (2BDP) (b) Throughput harm (5BDP) (c) Throughput harm (10BDP)

(d) Delay harm (2BDP) (e) Delay harm (5BDP) (f) Delay harm (10BDP)

Figure 15: Effect of buffer size on the CC protocols' harm for the 5G city driving channel trace.

wireless channels with low latency. BBR uses an estimate of the available bottleneck link bandwidth and RTT to govern its pacing rate. Copa [3] leverages the observed minimum RTT to achieve a target rate that optimizes a natural throughput function and delay under a Markov packet arrival model. More on CC protocols and their performance analysis can be found in recent surveys such as [15] and [34].

6 Discussion and Conclusion

While the development of 5G technologies has kicked into high gear, it is crucial to enable CC developers to test their implementations on top of realistic 5G mmWave-based environments. Therefore, we aim to provide a benchmarking platform *Zeus* to fill this gap by allowing evaluation and analysis of multiple CC algorithms under different 5G channels, and aid in designing new CC algorithms.

Our analysis of an extensive array of existing CC algorithms evinces a significant gap between most protocol performances and channel capacity in 5G. We additionally experimented with creating a hypothetical algorithm, *a Delayed Oracle Model*, that has perfect knowledge of the current 5G channel state but incurs a delay in communicating the state to the sender. Even in the presence of such knowledge, we observed in our 5G traces that with delayed feedback from the Oracle, the sender is not able to fully saturate the 5G link in the face of sudden up-spikes or avoid increased end-to-end delays in the face of sudden down-spikes. This simple experiment illustrates the difficulty of designing efficient congestion control algorithms that can achieve link saturations at low delays over such 5G channels.

Our evaluation of ten different protocols in 5G environments unearthed several limitations of individual protocols without any clear winners. While classic TCP variants inherently could not provide low delays, BBR had acceptable delay/throughput tradeoffs under high buffers; however, BBR did not exhibit compatible behavior alongside TCP Cubic under different queue sizes. Additionally, choosing an appropriate buffer sizes for their fair operation proved tricky across different 5G environments. Copa sacrifices throughput at the expense of maintain strict delay guarantees. Allegro performed relatively better than many of its delay-based CC protocol variants achieving a reasonable delay-throughput tradeoff while Proteus-P achieved higher throughput at the expense of potentially much higher delays; however, neither of them could saturate the underlying channels without dramatic increases in buffering delays. Verus and Vivace exhibit high performance variability over similar 5G network conditions, and do not co-existing well with Cubic over such channels.

In summary, an essential gap in congestion control research in the 5G domain still exists. Even with perfect knowledge of the 5G channel it is still fairly difficult to maintain high utilization without compromising the end-to-end delay. Designing a CC algorithm that can address the 5G challenge is still an open research question. Additionally, the research community must evaluate CC schemes for 5G over broader network conditions and from several aspects, such as: protocols' stability over drastically changing 5G network conditions, efficient behavior over a range of buffer sizes, and consistent behavior alongside existing protocols, particularly from the perspective of the harms they cause to each other.

References

[1] Soheil Abbasloo, Yang Xu, and H Jonathan Chao. C2tcp: A flexible cellular tcp to meet stringent delay require-

ments. *IEEE Journal on Selected Areas in Communications*, 37(4):918–932, 2019.

[2] Soheil Abbasloo, Chen-Yu Yen, and H. Jonathan Chao. Classic meets modern: A pragmatic learning-based congestion control for the internet. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 632–647, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Venkat Arun and Hari Balakrishnan. Copa: Practical delay-based congestion control for the internet. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 329–342, 2018.

[4] Lawrence S Brakmo, Sean W O'Malley, and Larry L Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications*, pages 24–35, 1994.

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

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

[7] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira. {PCC}: Re-architecting congestion control for consistent high performance. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 395–408, 2015.

[8] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. {PCC} vivace: Online-learning congestion control. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 343–356, 2018.

[9] Xenofon Foukas, Fragkiskos Sardis, Fox Foster, Mahesh K. Marina, Maria A. Lema, and Mischa Dohler. Experience building a prototype 5g testbed. In *Proceedings of the Workshop on Experimentation and Measurements in 5G*, EM-5G'18, page 13–18, New York, NY, USA, 2018. Association for Computing Machinery.

[10] Jill C Gallagher and Michael E DeVine. Fifth-generation (5G) telecommunications technologies: Issues for congress. *Congressional Research Service*, 1(30):1–39, 2019.

[11] Yasaman Ghasempour, Claudio RCM da Silva, Carlos Cordeiro, and Edward W Knightly. IEEE 802.11 ay: Next-generation 60 GHz communication for 100 Gb/s Wi-Fi. *IEEE Communications Magazine*, 55(12):186–192, 2017.

[12] Prateesh Goyal, Anup Agarwal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. ABC: A simple explicit congestion controller for wireless networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 353–372, Santa Clara, CA, February 2020. USENIX Association.

[13] Rajesh Gupta, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. Tactile-internet-based telesurgery system for healthcare 4.0: An architecture, research challenges, and future directions. *IEEE Network*, 33(6):22–29, 2019.

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

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

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

[17] Danping He, Bo Ai, Cesar Briso-Rodriguez, and Zhangdui Zhong. Train-to-infrastructure channel modeling and simulation in mmWave band. *IEEE Communications Magazine*, 57(9):44–49, 2019.

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

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

[20] Nura Jabagi, Andrew Park, and Jan Kietzmann. The 5G Revolution: expectations versus reality. *IT Professional*, 22(6):8–15, 2020.

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

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

[23] George Kalfas, Christos Vagionas, Angelos Antonopoulos, Elli Kartsakli, Agapi Mesodiakaki, Sotirios Papaioannou, Pavlos Maniotis, John S Vardakas, Christos Verikoukis, and Nikos Pleros. Next generation fiber-wireless fronthaul for 5G mmWave networks. *IEEE Communications Magazine*, 57(3):138–144, 2019.

[24] Brett Levasseur, Mark Claypool, and Robert Kinicki. Impact of acknowledgments on application performance in 4G LTE networks. *Wireless Personal Communications*, 85(4):2367–2392, 2015.

[25] Shao Liu, Tamer Başar, and R. Srikant. Tcp-illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *Performance Evaluation*, 65(6):417–440, 2008. Innovative Performance Evaluation Methodologies and Tools: Selected Papers from ValueTools 2006.

[26] Yiqing Ma, Han Tian, Xudong Liao, Junxue Zhang, Weiyan Wang, Kai Chen, and Xin Jin. Multi-objective congestion control. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 218–235, 2022.

[27] Tong Meng, Neta Rozen Schiff, P Brighten Godfrey, and Michael Schapira. PCC proteus: Scavenger transport and beyond. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 615–631, 2020.

[28] Marco Mezzavilla, Menglei Zhang, Michele Polese, Russell Ford, Sourjya Dutta, Sundeep Rangan, and Michele Zorzi. End-to-end simulation of 5g mmwave networks. *IEEE Communications Surveys Tutorials*, 20(3):2237–2263, 2018.

[29] Ravi Netravali, Anirudh Sivaraman, Keith Winstein, Somak Das, Ameesh Goyal, and Hari Balakrishnan. Mahimahi: A lightweight toolkit for reproducible web measurement. *SIGCOMM Comput. Commun. Rev.*, 44(4):129–130, aug 2014.

[30] Yong Niu, Chuhan Gao, Yong Li, Li Su, Depeng Jin, and Athanasios V Vasilakos. Exploiting device-to-device communications in joint scheduling of access and backhaul for mmWave small cells. *IEEE Journal on Selected Areas in Communications*, 33(10):2052–2069, 2015.

[31] Stefan Parkvall, Yufei Blankenship, Ricardo Blasco, Erik Dahlman, Gabor Fodor, Stephen Grant, Erik Stare, and Magnus Stattin. 5G NR release 16: Start of the 5G evolution. *IEEE Communications Standards Magazine*, 4(4):56–63, 2020.

[32] Stefan Parkvall, Erik Dahlman, and Johan Sköld. *5G NR: the next generation wireless access technology*. Academic Press, 2018.

[33] Michele Polese, Marco Giordani, Arnab Roy, Sanjay Goyal, Douglas Castor, and Michele Zorzi. End-to-end simulation of integrated access and backhaul at mmWaves. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–7. IEEE, 2018.

[34] Yongmao Ren, Wanghong Yang, Xu Zhou, Huan Chen, and Bing Liu. A survey on TCP over mmWave. *Computer Communications*, 2021.

[35] George F. Riley and Thomas R. Henderson. The ns-3 network simulator. In Klaus Wehrle, Mesut Günes, and James Gross, editors, *Modeling and Tools for Network Simulation*, pages 15–34. Springer, 2010.

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

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

[38] Min Sheng, Juan Wen, Jiandong Li, Ben Liang, and Xijun Wang. Performance analysis of heterogeneous cellular networks with HARQ under correlated interference. *IEEE Transactions on Wireless Communications*, 16(12):8377–8389, 2017.

[39] HE Shiwen, Yongming HUANG, WANG Haiming, and HONG Wei. Development trend and technological challenges of millimeter-wave wireless communication. *Telecommunications Science*, 33(6):11.

[40] Christopher Slezak, Menglei Zhang, Marco Mezzavilla, and Sundeep Rangan. 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)*, pages 1–5. IEEE, 2018.

[41] Hao Song, Xuming Fang, and Yuguang Fang. Millimeter-wave network architectures for future high-speed railway communications: Challenges and solutions. *IEEE Wireless Communications*, 23(6):114–122, 2016.

[42] Bruce Spang, Brady Walsh, Te-Yuan Huang, Tom Rusnock, Joe Lawrence, and Nick McKeown. Buffer sizing and video QoE measurements at Netflix. In *Proceedings of the 2019 Workshop on Buffer Sizing*, pages 1–7, 2019.

[43] Rakesh Taori and Arun Sridharan. Point-to-multipoint in-band mmwave backhaul for 5G networks. *IEEE Communications Magazine*, 53(1):195–201, 2015.

[44] Juan Pedro Thomas. China Mobile already operates 50,000 5G base stations: report. Accessed on 10.05.2022.

[45] Andras Varga. OMNeT++. In *Modeling and tools for network simulation*, pages 35–59. Springer, 2010.

[46] Ranysha Ware, Matthew K Mukerjee, Srinivasan Seshan, and Justine Sherry. 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*, pages 17–24, 2019.

[47] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 459–471, 2013.

[48] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. Pantheon: the training ground for Internet congestion-control research. In *2018 {USENIX} Annual Technical Conference ({USENIX} {ATC} 18)*, pages 731–743, 2018.

[49] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive congestion control for unpredictable cellular networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 509–522, 2015.

[50] Pei Zhou, Kaijun Cheng, Xiao Han, Xuming Fang, Yuguang Fang, Rong He, Yan Long, and Yanping Liu. IEEE 802.11 ay-based mmWave WLANs: Design challenges and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):1654–1681, 2018.

[51] Tommaso Zugno, Michele Polese, Natale Patriciello, Biljana Bojović, Sandra Lagen, and Michele Zorzi. Implementation of a Spatial Channel Model for ns-3. In *Proceedings of the 2020 Workshop on ns-3*, pages 49–56, 2020.

[52] Tommaso Zugno, Michele Polese, and Michele Zorzi. Integration of carrier aggregation and dual connectivity for the ns-3 mmWave module. In *Proceedings of the 10th Workshop on ns-3*, pages 45–52, 2018.

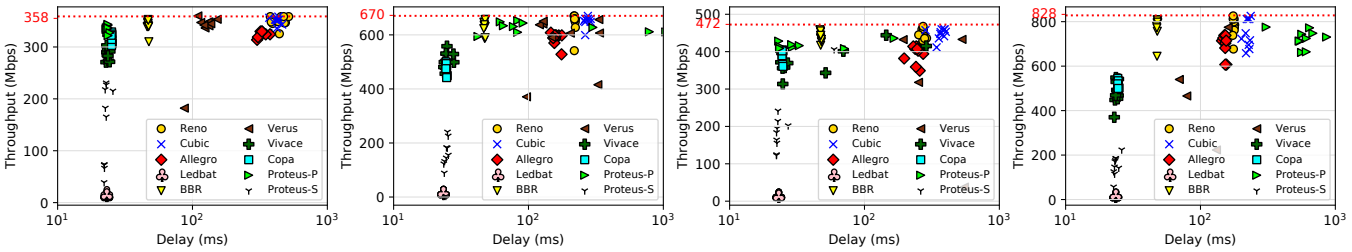
A Traces generation methodology

A.1 WiGig router (60 GHz)

These traces were all collected in an indoor office environment. The WiGig router was connected over Ethernet to a Linux machine, and over the WiGig link to an Acer TravelMate P648 laptop, which has a WiGig-enabled network card. A UDP sender at the Linux machine sent packets of fixed size (1500 bytes) over UDP at line rate to the laptop, and timestamp of each received packet was recorded on the laptop. This log was used to compute the inter-arrival times between the packets, which were then used to create a channel trace file that contains the number of arrivals in every unit of time (millisecond) for the entire duration of the trace. When emulating this channel link, the number of arrivals would correspond to the available bandwidth in that millisecond. We collected four WiGig channel traces – ***fan_running***: The laptop and the router are placed on the same desk, with a running fan placed in between them. The fan causes slight disturbances in the link; ***human_motion***: Random human motion is introduced between the laptop and router placed a few feet apart from each other in an office. The channel capacity is greatly affected by signal attenuation due to the presence of a human body; ***stationary***: The laptop is placed next to the router at line of sight. This trace alone was collected in a different indoor environment; ***walk_and_turn***: A person holding the laptop moves a few meters towards the router and then turns around.

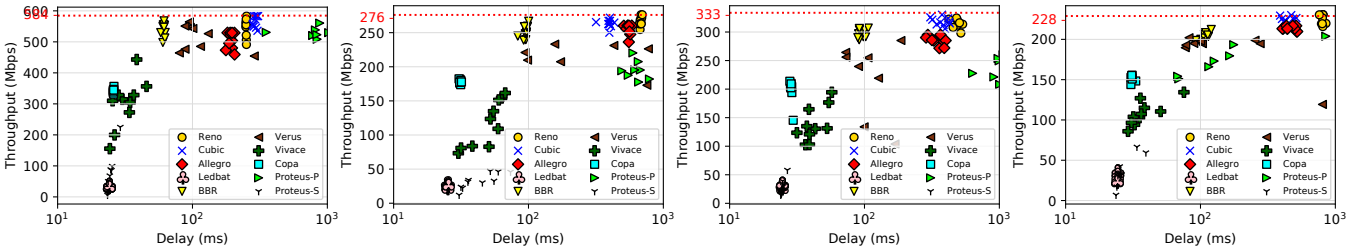
A.2 NS-3 scenarios

Although *Zeus* offers a tool to record real-world channel traces, this is an expensive approach. Therefore, *Zeus* also provides a customized NS-3 tool that enables the generation of 5G channel traces for different reference 5G mmWave scenarios. This allows us to define scenarios with desired characteristics aligned with the properties of CC solutions of interest, such as users' motion, capacity sharing or location (i.e., urban, rural). It also defines detailed communication configuration, including modulations and the number or type of antennas. Although NS-3 provides high flexibility in the generation of scenarios (written in `C++`), it requires in many cases a deep knowledge of the simulator structure and implemented models. We have defined reference scenarios with different characteristics and simplified utilities to customize its operation to overcome this. These utilities include cellular stack settings (including buffer lengths, scheduling, and antenna configuration), users' mobility configuration (predefined track, static, constant speed, and random walks), and generation of performance traces at different layers. In addition, each scenario can be executed with a different random seed to obtain various traces with the same characteristics.



(a) `Beach stationary` (b) `Park` (c) `City walking` (d) `City stationary`

Figure 16: Additional real 5G traces scatter plots



(a) `Office` (b) `Buildings scenario` (c) `Rural macro` (d) `Urban macro`

Figure 17: Additional ns-3 traces scatter plots

We have divided the ns-3 scenarios into two groups: (i) with buildings and predefined users' motion, and (ii) without buildings and with random users' motion. First, the two most straightforward scenarios (`Buildings`$_1$ and `Buildings`$_2$) comprise one user walking along a street between two buildings towards and away from a base station. Then, in the `Buildings`$_3$ scenario we increment the complexity of the setup by deploying a grid of 3×3 buildings and defining a different users track. The `StreetCanyon` scenario comprises two base stations, and the users move following a straight line getting close or far to each of the access elements. Unlike the previous configurations, in the latter the buildings are randomly deployed to obtain different topologies in this setup. All scenarios with buildings utilizes the urban macro (UMa) 3GPP propagation models, except the `StreetCanyon` scenario where the urban micro (UMi) *street canyon* 3GPP model is used. The second group of scenarios represents open areas with different characteristics. In particular, we exploit the UMa, rural macro (RMa) and office indoor hotspot (InH) mixed office 3GPP models to generate three scenarios with the same names (namely, `UMa`, `RMa`, `InhOfficeMixed`) [51]. Unlike the building scenarios, where the user mobility is predefined, in this case, users move following a random pattern with different average distances.

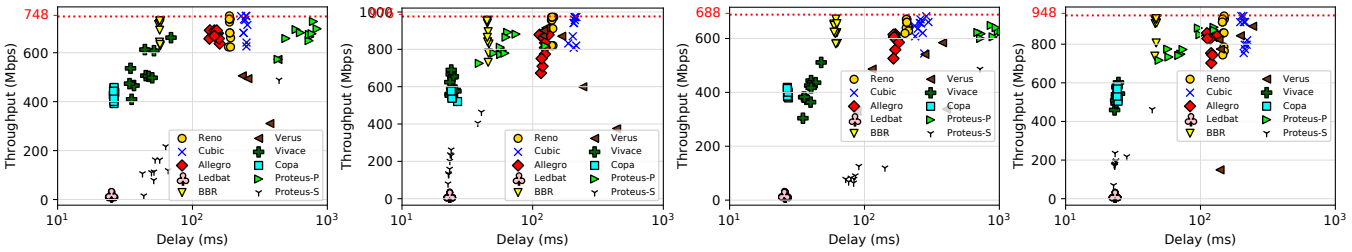
In all of the above cases, the traffic is sent from a server to the user with constant bit-rate, and a packet length of 1472 `bytes`. After the simulation, we obtain a trace that logs all the reception events at the application layer. Although the main objective of this module is the generation of the transport layer trace, the simulation of the scenarios also generates traces from lower layers. Among others, we have implemented a tailored trace generation for the link layer to record the RLC buffer's evolution, which can be seen as the bottleneck buffer when the radio conditions are poor.

Finally, to simplify the generation of traces and their analysis, we have developed a set of `Python` scripts to automate the execution of scenarios varying the data rate, simulation time, and random seed. As a result, we obtain separated trace files for each scenario and configuration. The scripts also use the generated traces to automatically create plots that show the temporal evolution of some parameters (i.e., communications capacity, SINR, RLC, buffer) and the scenario topology (i.e., buildings, access elements, and users position). As has been commented, we use UDP as a transport protocol in saturation. It permits the generation of traces that show the maximum achievable capacity.

In all cases, it is worth noting that *Zeus* is designed to be an extensible framework. Therefore, it contains client/server software and simulation tools to generate traces with different technologies and scenario characteristics.

B Additional 5G Channel Traces Evaluations

Figures 16, 17, 18 show the performance of these congestion control algorithms on 12 additional channel traces. These traces encompass a wide range of real world scenarios in order to simulate the different types of cellular network conditions



(a) Indoor random walking (b) Stationary (c) Walking 1 (d) Walking 2

Figure 18: Additional Wi-Gig traces scatter plots

a user might experience on a day-to-day basis. The results on these additional traces show the same trend as the traces discussed earlier where Cubic and Reno have high channel utilization with high delays, BBR has high channel utilization with significantly lower delays, Allegro has a similar but slightly lower throughput compared to Cubic with similar delays, Copa and Vivace consistently utilize a moderate portion of the channel capacity while keeping low delays, Proteus-P and Verus outperform Copa and Vivace in terms of throughput but experience very high delays in certain scenarios, and finally Ledbat and Proteus-S tend to have low throughput and low delays. It is interesting to note that, despite the channels being significantly different in terms of the amount of fluctuation in the channels, with some channels being stable and others being highly variable, and their average capacities which range from 200Mbps all the way to above 800 Mbps, most of the CC algorithms display a consistent pattern in their behavior across all traces.

C Additional harm analysis with different buffer sizes

The harm analysis as seen in Figures 19 and 20 sheds further light on how the CC algorithms are affected when sharing a link with TCP Cubic. There are some notable observations we can make from this analysis. When competing against itself, TCP Cubic shares the link equally amongst multiple flows. Regardless of buffer size, Copa does not harm the TCP flow in any of the tested scenarios. In a shallow buffer, BBR is able to cause significant throughput harm to Cubic but as the buffer sizes increase, the harm caused by BBR to cubic decreases. The same applies to Verus as well along with Allegro for the WiGig walking channel trace, however, in the Street Canyon channel trace Allegro consistently causes about 70% throughput harm to Cubic. Reno tends to cause more harm to TCP Cubic as buffer size increases, however, it does not inflict positive harm in any of the tested scenarios which means that the maximum harm inflicted by Reno to Cubic is when Reno and Cubic share the link equally. In terms of delay harm, all the algorithms inflict negative delay harm on Cubic in the tested scenarios except for the WiGig walking trace with low BDP where BBR, Verus, and Allegro inflict up to 50% average delay harm on Cubic.

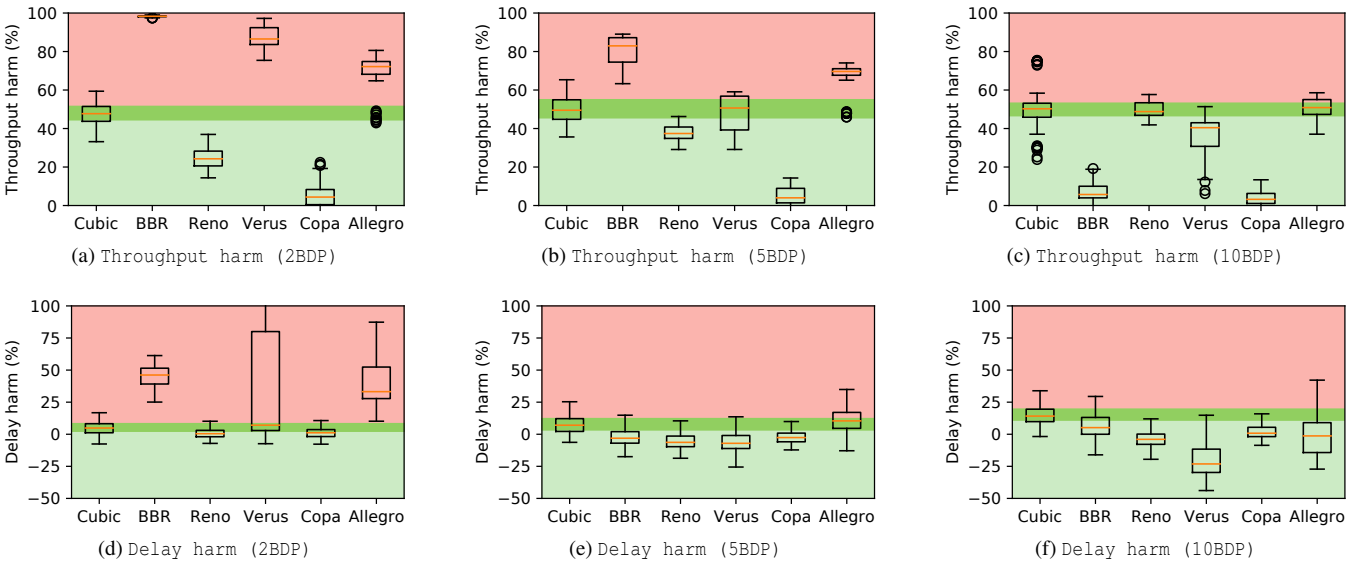
We excluded assessing the Verus harm on Cubic from the street canyon trace due to Verus's highly unstable performance. We noticed that Verus, in most cases, failed to exit its start-up phase.

D Self-harm of the CC protocols

This Annex shows the results related to the harm that a new flow of a CC protocol may have over existing flows of the same type.

First, Figure 21 depicts the self-harm of the more performing CC protocols, upon different buffer sizes, in the real 5G cellular channel trace. It is measured by comparing the performance of a single flow of each CC protocol with its performance when it shares the channel with another flow of the same type. We do not include Cubic's results since it has been already studied in detail in Figures 11, 12 and 13. In the upper figures, Figure 21a, 21b and 21c,we show the throughput harm and indicate with a dotted line the 50%, which would represent the case where both flows having same throughput. As can be seen, BBR exhibits a harm value slightly below 50 % for shallow buffers, indicating that the protocol performance is not much affected in that case. As we increment the BDP buffer size, BBR's throughput harm tends to 50%. In the case of Reno and Allegro, the results show that the throughput harm is very close to 50 % across the various bottleneck buffer size. Similarly, Verus always presents median throughput self-harm close to 50 %, but with significant variance, evincing its unpredictable behavior. On the other hand, Copa's self-harm is always below 50 % and relatively stable regardless of the bottleneck buffer size. Although Copa's throughput self-harm also has a large variance, the results show its robustness upon flows of the same type. It is worth noting that negative harm values indicate improved delays when sharing the bottleneck with the competing flows. In fact, these unpredictable results yielded by Verus and Copa would indicate some lack of adaptability to the channel.

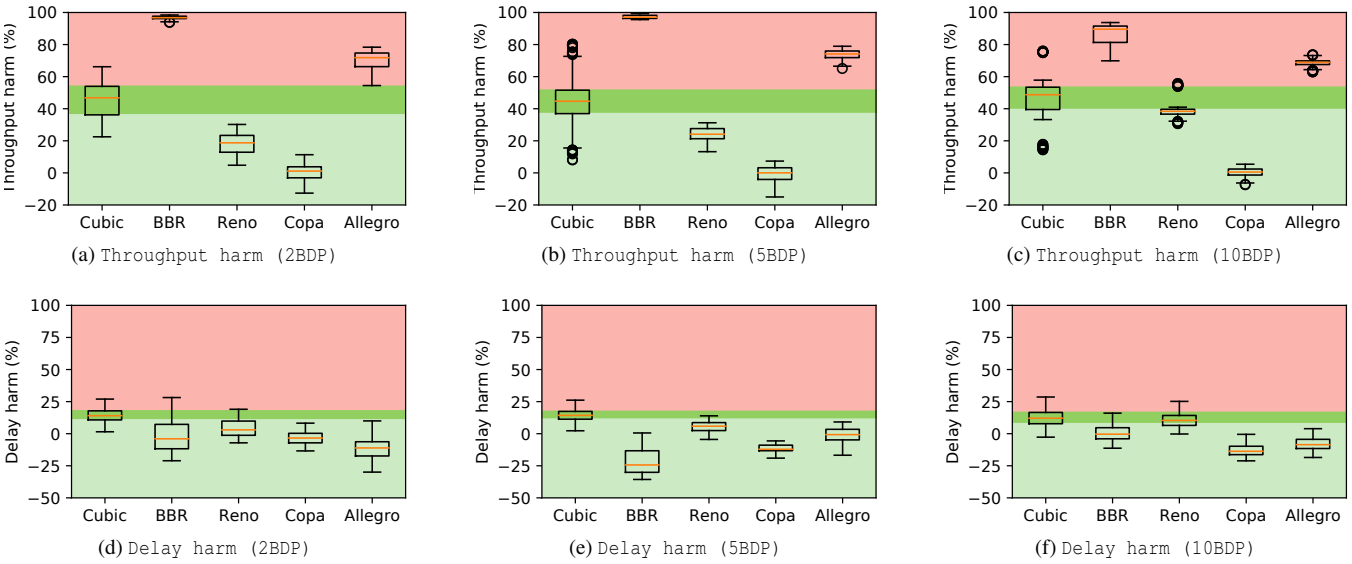
Figures 21d, 21e and 21f show the delay self-harm. BBR's delay self-harm strongly depends on buffer size, going from



(a) Throughput harm (2BDP) (b) Throughput harm (5BDP) (c) Throughput harm (10BDP)

(d) Delay harm (2BDP) (e) Delay harm (5BDP) (f) Delay harm (10BDP)

Figure 19: Effect of buffer size on the CC protocols' harm for the WiGig walking channel trace.



(a) Throughput harm (2BDP) (b) Throughput harm (5BDP) (c) Throughput harm (10BDP)

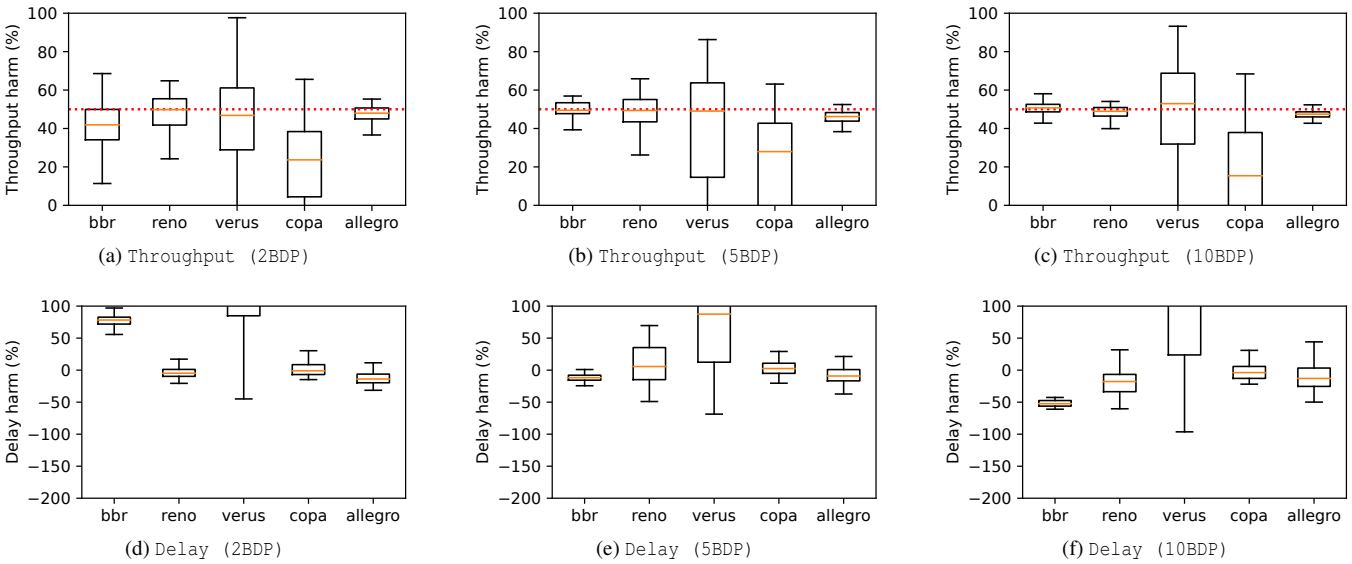
(d) Delay harm (2BDP) (e) Delay harm (5BDP) (f) Delay harm (10BDP)

Figure 20: Effect of buffer size on the CC protocols' harm for the street canyon channel trace.

around 75% (negative impact) for shallow buffers, to -50% (positive impact) in deeper ones. On the other hand, Reno, Copa and Allegro present a rather stable delay self-harm which is, in average, around 0 (no impact) for the different buffer sizes. Finally, we can see that Verus presents a high variance in this metric, as was already observed for the throughput self-harm.

To complement the previous results, in Figure 22 we plot the temporal evolution of the throughput and delay when a flow runs alone and when two flows share the bottleneck buffer. In all cases, the flow alone and the competing flows are represented with and without sub-indices, respectively. For instance, in Figure 21a the legend entry *BBR* refers to the performance of a BBR flow alone, and the legend entries *BBR1* and *BBR2* to the performance of the competing flows sharing the bottleneck buffer.

The results are aligned to the previous ones. BBR, Reno and Allegro, Figures 22a-22c, 22d-22f and 22m-22o respectively, show a good sharing of the channel capacity. As can be observed, the throughput with the competing flows is around half of that seen with a flow alone, and more importantly the competing flows have very similar performance. Also, in

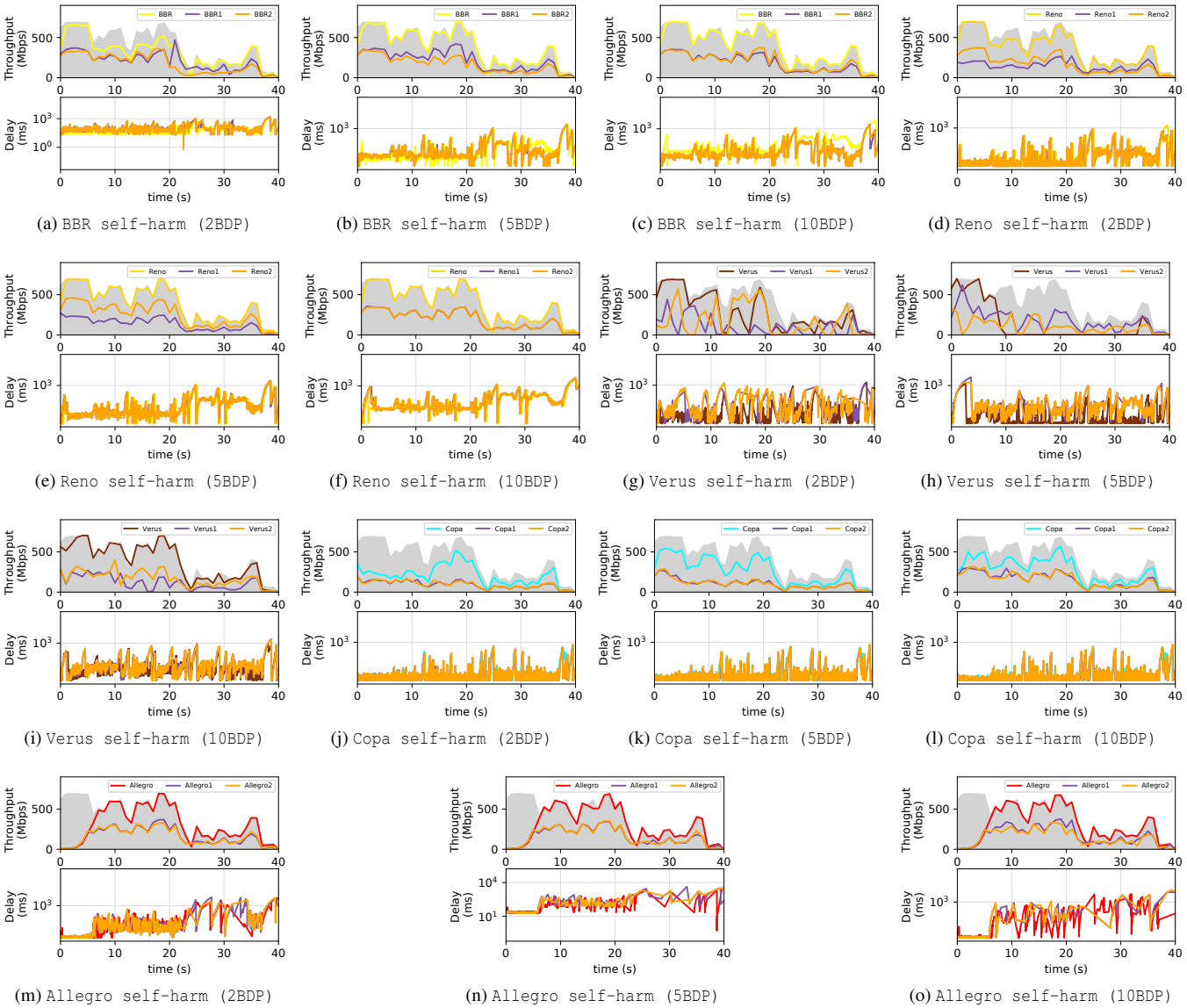


(a) Throughput (2BDP) (b) Throughput (5BDP) (c) Throughput (10BDP)

(d) Delay (2BDP) (e) Delay (5BDP) (f) Delay (10BDP)

Figure 21: Effect of buffer size on the CC protocols' self-harm for the 5G city driving channel trace.

the case of Copa, the competing flows have similar behavior. However, we see in Figure 21 that the throughput harm was, in general, below 50%. In Figures 22j-22l, we can observe that Copa with a single flow is far from fully achieving the channel capacity so that the competing flows can achieve a throughput above the middle of that reached by a single flow. It is worth noting that other metrics, like Jain's fairness index, would indicate a good sharing of the bottleneck buffer. As expected, Verus results shown in Figure 22g-22i reflect the unpredictable behavior of the protocol, where the performance of the competing flows is above that yielded by the flow alone. As can be observed in, for instance, Figure 22h, in the time slot between 10 and 20 seconds, the throughput reached by the flow alone was much below the channel capacity. However, when competing flows are deployed, one of them almost reaches the channel capacity, which would lead to improvement and so negative harm. In sum, the harm Verus' values shown in Figure 21 again reflects the lack of adaptation to the fluctuations of the channel capacity.



(a) BBR self-harm (2BDP) (b) BBR self-harm (5BDP) (c) BBR self-harm (10BDP) (d) Reno self-harm (2BDP)

(e) Reno self-harm (5BDP) (f) Reno self-harm (10BDP) (g) Verus self-harm (2BDP) (h) Verus self-harm (5BDP)

(i) Verus self-harm (10BDP) (j) Copa self-harm (2BDP) (k) Copa self-harm (5BDP) (l) Copa self-harm (10BDP)

(m) Allegro self-harm (2BDP) (n) Allegro self-harm (5BDP) (o) Allegro self-harm (10BDP)

Figure 22: Effect of buffer size on the CC protocols' self-harm for the 5G city driving channel trace.