RAN-aware Proxy-based Flow Control for High Throughput and Low Delay eMBB

Mamoutou Diarra
mamoutou.diarra@inria.fr
Ekinops, Inria, Université Côte d'Azur
Sophia Antipolis, France

Walid Dabbous
walid.dabbous@inria.fr
Inria, Université Côte d'Azur
Sophia Antipolis, France

Amine Ismail
amine.ismail@ekinops.com
Ekinops
Sophia Antipolis, France

Thierry Turletti
thierry.turletti@inria.fr
Inria, Université Côte d'Azur
Sophia Antipolis, France

ABSTRACT

5G enhanced Mobile broadband (eMBB) aims to provide users with a peak data rate of 20 Gbps in the Radio Access Network (RAN). However, since most Congestion Control Algorithms (CCAs) rely on startup and probe phases to discover the bottleneck bandwidth, they cannot quickly utilize the available RAN bandwidth and adapt to fast capacity changes without introducing large delay increase, especially when multiple flows are sharing the same Radio Link Control (RLC) buffer. To tackle this issue, we propose RAPID, a RAN-aware proxy-based flow control mechanism that prevents CCAs from overshooting more than the available RAN capacity while allowing near optimal link utilization. Based on analysis of up-to-date radio information using Multi-access Edge Computing (MEC) services and packet arrival rates, RAPID is able to differentiate slow interactive flows from fast download flows and allocate the available bandwidth accordingly. Our experiments with concurrent Cubic and BBR flows show that RAPID can reduce delay increase by a factor of 10 to 50 in both Line-of-Sight (LOS) and Non-LOS (NLOS) conditions while preserving high throughput.

KEYWORDS

TCP Congestion and Flow control, 5G millimeter wave, self-inflicted bufferbloat, eMBB, Multi-Access Edge Computing, Radio Network Information Service, Performance Enhancing Proxy

1 INTRODUCTION

Over the last years, Internet content and resources have increasingly been located close to the end users by means such as Content Delivery Networks (CDN) or edge computing. In such a context and as today's cellular networks rely on well-provisioned backhaul, the radio link between the user and the base station, as illustrated in Figure 1, most often becomes the bottleneck [12], mostly due to the inherent characteristics and limitations of cellular Radio Access Networks (RANs). Consequently, if not managed appropriately, such a bottleneck could hinder 5G performances, especially in enhanced Mobile Broadband (eMBB) scenarios.

Moreover, traditional end-to-end Congestion Control Algorithms (CCAs) are not designed to appropriately discover cellular RAN bandwidth without introducing large delay or bufferbloat. This phenomenon is observed in startup and steady states of most CCAs

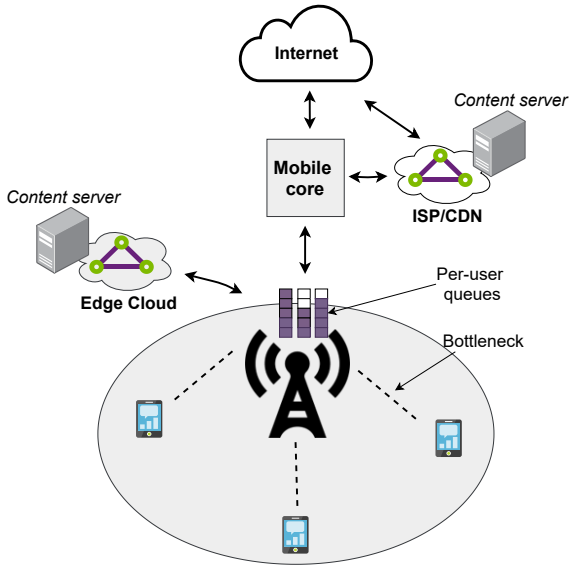


Figure 1: Bottleneck location in cellular networks.

(e.g., NewReno, Cubic) since they rely on probing techniques in order to gradually discover the available bandwidth. Although some end-to-end solutions such as model-based (e.g BBR [4]) and cross-layer CCAs (e.g., PBE-CC [31], CQIC [34], LCTCP [2], etc.) have been proposed in order to better utilize the RAN bandwidth while minimizing Round-Trip-Time (RTT) increase, the bufferbloat issues still persist since the proposed algorithms cannot perform well when competing with loss-based CCAs which keep introducing significant RTT increase due to the use of deep buffers in the base stations. Other alternatives to end-to-end solutions such as RAN-aware proxies (e.g., milliProxy [26]) have also been proposed in order to mitigate the negative effects of loss-based CCAs; however, these solutions can introduce high delay increase in case of multiple flows per user. In general, cross-layer CCAs and existing RAN-aware proxies adapt their sending rate to the RAN bandwidth allocated to the User Equipment (UE) without taking into account the number of active flows sharing this bandwidth in the UE. As a result, the per-user RLC queue (at the base station) can grow and cause self-inflicted bufferbloat when multiple flows progress

in parallel in the same UE.
Indeed, all the aforementioned limitations along with the diversity of congestion control algorithms on the Internet make it very challenging to address the self-inflicted bufferbloat problem at the CCA level. In such a context, it could be necessary to use a flow-level bandwidth shaping in order to limit the interference between the flows that are using different CCAs. As suggested in [15], a simple solution could be to allocate equal portions of the available bandwidth to the concurrent flows; however such a solution may result in link under-utilization since some flows (e.g., slow or app-limited flows) may require just a small fraction of the total bandwidth. With that in mind, we design and evaluate a solution that minimizes RTT increase (regardless of the CCA used by the end servers) and allocates the available RAN bandwidth by taking into account the nature of the active flows in the UE. Overall, our contributions in this paper can be summarized as follows:

- We propose a RAN-aware performance enhancing proxy that intercepts TCP connections and distributes proportionally the available RAN bandwidth among the active flows. On this purpose, we rely on the MEC Radio Network Information Service (RNIS) and on packets arrival rates to estimate the aggregated RAN bandwidth and categorize the concurrent TCP flows in the UE. This approach does not involve the end user unlike existing cross-layer CCAs that introduce computational overhead and additional power consumption at the UE as they require client-side modifications in order to introduce radio information into the TCP header [31, 34].
- We provide an implementation of our solution for the ns-3 network simulator on top of which we conduct various performance evaluation experiments with different CCAs under both LOS and NLOS radio conditions. We run our simulations on a cluster[1] and the scripts to reproduce all our scenarios/results are made available to the community [8].
- We highlight the limits of current CCAs over next-generation cellular networks and provide building blocks for their improvement.

The remainder of the paper is organized as follows. In Section 2 we present the motivations of this work and shed light on the self-inflicted bufferbloat problem and bandwidth under-utilization issue introduced by traditional CCAs in cellular networks. Section 3 presents the design and implementation details of RAPID, our proposed RAN-aware proxy. Section 4 describes the simulation environment, evaluates the performance of our solution in some mobile edge scenarios and discusses some observed limitations. Section 5 explores the related work and finally, we summarize our contributions and future research directions in Section 6.

2 MOTIVATIONS

The use of per-user queues in cellular networks has almost eliminated the likelihood of flow interactions and CCAs interference between separate users [12]. However, flows of the same user may still use different CCAs, and depending on the design and TCP friendliness of the involved CCAs, a large bufferbloat can be introduced [16, 27]. In fact, the probability of having multiple parallel flows on the same UE is quite high in today's cellular networks, mostly due to the increasing number of mobile services which create more opportunities for parallel data transfer [25]. However, this also increases the probability of self-inflicted bufferbloat since the parallel data flows on the UE may use different CCAs. This type of bufferbloat is called self-inflicted bufferbloat since it is introduced by the user itself. According to a recent study [22], over 30% of the top 20,000 Alexa websites are using Cubic and roughly 18% are currently based on BBR. This study also reveals that delay-based TCP variants such as Vegas/Veno and other unknown variants are used on 2.8 and 17.6% of the evaluated websites, respectively. Based on these findings, it appears that a large majority of current TCP traffic uses loss-based CCAs which are known to introduce excessive buffering in mobile networks. Furthermore, in mobile networks, due to the use of deep buffers in base stations, recent CCAs such as BBR that attempt to minimize delay increase are outperformed in terms of goodput by loss-based TCP variants when they compete for bandwidth. Other studies show that, in addition to being RTT unfair, BBR also suffers from bandwidth overestimation issues in fast-varying networks [6] and severe throughput collapse due to high delay variations over millimeter wave (mmWave) wireless links [18]. These issues along with the ever-growing number of CCAs that could potentially compete for bandwidth make it very challenging to solve the self-inflicted bufferbloat issue with a new CCA. It is also worth noting that the delay increase that may arise from these issues is particularly harmful for delay-sensitive flows and, therefore, greatly slow down web browsing as reported in [30].

The self-inflicted bufferbloat issue becomes even more significant in mmWave bands mostly due to the inherent properties of high frequencies. According to 3GPP TS 38.104, 5G New Radio (NR) can operate in two distinct frequency ranges, Frequency Range 1 (i.e., FR1 or sub-6GHz) and Frequency Range 2 (i.e., FR2 or 24.25 GHz to 52.6 GHz) which falls in the mmWave region. In fact, frequencies in the mmWave region are prone to high penetration losses due to obstacles blocking the Line-of-Sight. Therefore, in case of NLOS conditions, the base station buffers incoming packets which eventually leads to high delay increase and/or persistent bufferbloat [32].

Our main goal with this work is to mitigate the self-inflicted bufferbloat issues that arise from parallel flows or intermittent radio links using a new approach completely independent of the CCAs in use.

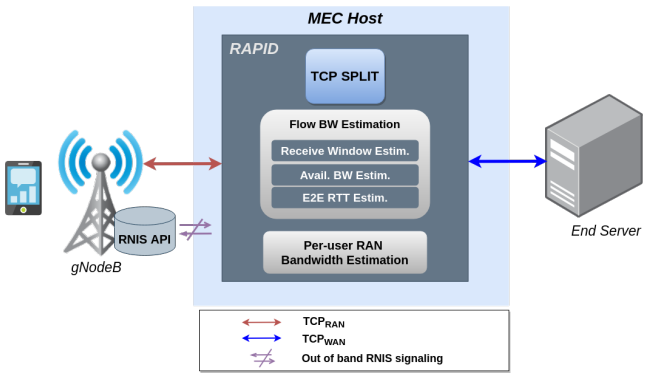


Figure 2: RAPID high-level architecture

[1]NEF cluster: https://wiki.inria.fr/ClustersSophia/Clusters_Home.

3 RAN-AWARE PROXY-BASED FLOW CONTROL

In this section, we first present the high-level design of our RAN-aware proxy called RAPID, and then show how it leverages actual RAN statistics and various transport layer related information in order to categorize flows and properly allocate the available bandwidth. RAPID's main originality w.r.t. solutions presented in Section 5 is the mitigation of self-inflicted bufferbloat while preserving high bandwidth regardless of the characteristics of the CCAs involved and the number of concurrent flows sharing the per-user queues.

3.1 Proxy architecture

RAPID is a TCP proxy that seats close to the base station, in the MEC host and that leverages up-to-date radio information exposed by the Radio Network Information Service (RNIS) [9], similar to RAVEN [28] and MELD (proposed in our previous work [7]). The use of RNIS in real-world experimentation has been demonstrated by FlexRAN [10], an open-source implementation of RNIS which allows a controller deployed somewhere in the network to collect radio information from some agents collocated with the deployed base stations. As illustrated in Figure 2, RAPID relies on several modules organized in layers, meaning that each module provides services to its upper module.

At a basic level, RAPID first splits each incoming end-to-end TCP flow from a UE in two separate connections, referred here as TCP_{RAN} (i.e., TCP connection between the UE and RAPID) and TCP_{WAN} (TCP connection between RAPID and the corresponding end server). After that, it invokes the per-flow bandwidth estimation module in order to compute the Bandwidth Delay Product (BDP) available to each intercepted flow. Finally, the computed BDPs are used to override the Receive Window (RW) value in the TCP acknowledgements towards the corresponding end servers, thereby limiting the bytes in flight from those servers to the BDPs estimated for their respective flows. It is worth noting that while RAPID has no control over the choice of the end-server CCA, it can still control all the parameters of the connections established in the RAN segment (i.e., TCP_{RAN}). Therefore, by default and in order to prevent excessive buffering in the proxy, TCP_{RAN} relies on a simplified CCA that always sets its congestion window to the estimated flow BDP.

3.2 RAN bandwidth estimation

As illustrated in Figure 2, RAPID periodically receives RAN statistics through the RNIS service. Similar to the statistics reported by FlexRAN agents [10], these statistics include the cell bandwidth, the sub-carrier spacing, the Radio Network Temporary Identifier (RNTI), the number of allocated Physical Resource Blocks (PRBs), the computed Transport Block Size (TBS) and the selected Modulation and Coding Scheme (MCS) information for all the UEs connected to the cell. This information allows RAPID to estimate the bandwidth allocated to each UE by the base station in units of PRBs. However, the number of allocated PRBs does not always reflect the achievable bandwidth since the MAC scheduler at the base station may also take into account the RLC buffer state [23] or the incoming data rate [31] when allocating resources to users. Therefore, in order to estimate the achievable bandwidth, we first calculate the number of unused PRBs, $Pu_{\text{i}}(t)$, by using the total number of PRBs in the cell N_{PRB}, the number of connected UE M and the number of PRBs allocated to each UE $Pa_{\text{i}}(t)$:

$$Pu_{\text{i}}(t) = N_{\text{PRB}} - \sum_{i=1}^{M} Pa_{\text{i}}(t)\,, \tag{1}$$

The expected number of PRBs, $Pe_{\text{i}}(t)$ for a given UE i is then computed as follows:

$$Pe_{\text{i}}(t) = Pa_{\text{i}}(t) + Pu_{\text{i}}(t)\,, \tag{2}$$

With the expected number of PRBs and the reported MCS values, RAPID computes the maximum transport block size at time t, $TBS_{\text{i}}(t)$, for each UE as detailed in 3GPP TS 38.214. Therefore, the highest achievable throughput $C_{\text{i}}(t)$ (in unit of $bits/s$) for a given UE, depending on the reported MCS, is expressed as:

$$C_{\text{i}}(t) = \frac{TBS_{\text{i}}(t)}{TTI}\,, \tag{3}$$

where TTI stands for the Transmission Time Interval (e.g., TTI = 125 μs for Numerology μ=3). It is worth noting that this real-time RAN bandwidth estimation makes it possible to detect fast capacity variations as well as bad radio conditions (e.g., NLOS conditions in mmWave), which, to some degree, can cause excessive buffering at the base stations.

3.3 Per-flow bandwidth allocation

The major contribution of RAPID with respect to other existing solutions is its bandwidth allocation scheme. Rather than relying on a static allocation scheme, the bandwidth estimation module distributes the aggregated bandwidth of the UE according to the requirements of the active flows in terms of bandwidth. On connection start, the initial Receive Window $RW_{\text{ij}}(t)$ of a given flow j of a UE i is computed by dividing the expected bandwidth estimated for the UE (in $bits/s$) by the number of active flows:

$$RW_{\text{ij}}(t) = \frac{C_{\text{i}}(t)}{N}\, RTTmin_{\text{ij}}(t)\,, \tag{4}$$

where N is the number of active flows in the UE and $RTTmin_{\text{ij}}(t)$ is the minimum end-to-end RTT of flow j at time t. This initial phase is followed by a dynamic allocation phase after K_{ij} RTT_{WAN}, where the allocated bandwidth changes depending on flow category. We define two distinct categories of flows: slow interactive flows (i.e., browsing, instant messaging, etc.) and fast download flows (e.g., HD streaming, file transfer, Augmented and Virtual Reality AR/VR, etc.). The former are sensitive to delay but require low bandwidth while the latter may require both high bandwidth and reasonable delay variations. In such a context, allocating the same amount of bandwidth to all flows not only penalizes fast download flows (since slow flows only require a small fraction of the bandwidth) but also results in radio link under-utilization and wastage of the bandwidth allocated to the UE. With that in mind, we devise a flow categorization scheme based on a common behavior of TCP flows during startup phase. In fact, since the congestion windows of most TCP flows follow an exponential increase pattern during the initial phase, we can compute the number K_{ij} of RTTs it takes to reach

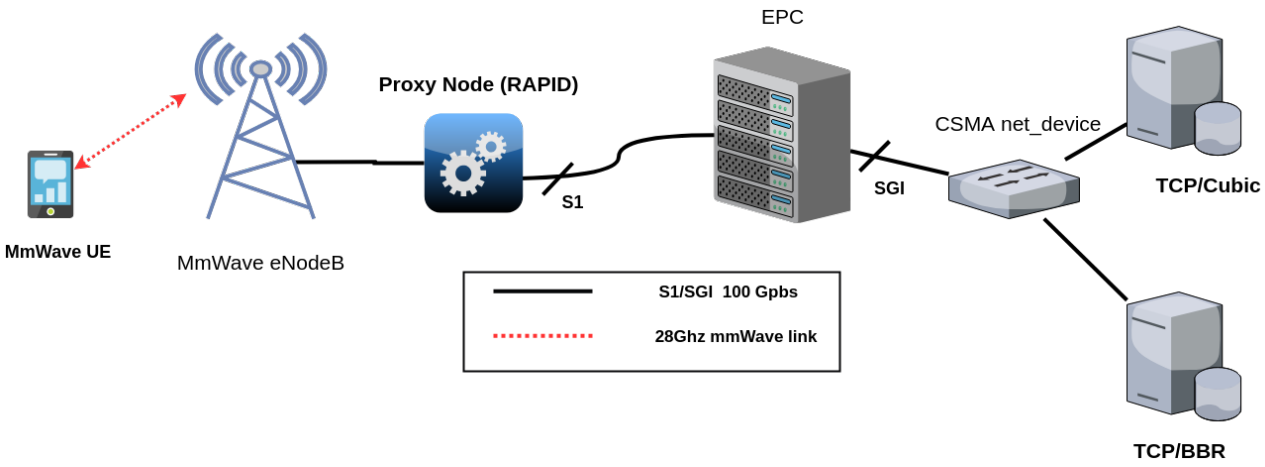


Figure 3: RAPID ns-3 experimentation testbed

$RW_{ij}(t)$ worth of bytes in flight as follows:

$$K_{ij}(t) = \log_2\left(\frac{RW_{ij}(t)}{MSS}\right) + 1\,, \tag{5}$$

Following the same exponential increase pattern, after K_{ij} RTTs, the expected arrival rate R_{ij} at TCP_{WAN} should be:

$$R_{ij}(t+K) = \frac{RW_{ij}(t)}{RTT_{WANij}}\,, \tag{6}$$

where MSS stands for the negotiated Maximum Segment Size and RTT_{WANij} is the current smoothed RTT of TCP_{WANij}. Therefore, after K_{ij} RTT_{WANij}, if the actual arrival rate, r_{ij}, is less than a certain percentage (set empirically to 80%) of the expected rate, the flow is flagged as slow; otherwise, it is considered as a fast download flow. The Receive Windows of slow flows are then computed as follows:

$$RW_{ij}(t+K) = \frac{\hat{r}_{ij}(t+K)}{R_{ij}(t+K)}\, RW_{ij}(t)\,, \tag{7}$$

where $\hat{r}_{ij}(t+K)$ is the exponential moving average of the arrival rate r_{ij} at time $(t+K)$. Using the exponential moving average of all the past arrival rates allows capturing irregular variations of the congestion window during the categorization period (i.e., during K RTTs). After each change of Receive Window, the available bandwidth not used by the UE is updated as follows:

$$A_i(t+K) = \sum_{j=1}^{Z} A_{ij}(t+K)\,, \tag{8}$$

$$A_{ij}(t+K) = |RW_{ij}(t) - RW_{ij}(t+K)|\,, \tag{9}$$

where A_{ij} is the unused bandwidth at the flow level, A_i is the aggregated unused bandwidth at the UE level and Z, the number of slow flows identified. In order to avoid wasting bandwidth, the available unused bandwidth is evenly distributed among the fast download flows of the UE. Therefore, the Receive Windows of these flows are updated as follows:

$$RW_{ij}(t+K) = RW_{ij}(t) + \frac{A_i(t+K)}{N-Z}\,. \tag{10}$$

The dynamic allocation phase is repeated for each active flow until the end of the connection in order to maximize the aggregated link utilization at the UE level. Indeed, the available bandwidth is distributed among the concurrent flows regardless of the CCA in use. However, CCAs with aggressive growth functions yield higher link utilization since they reach the target Receive Window faster.

4 EVALUATION

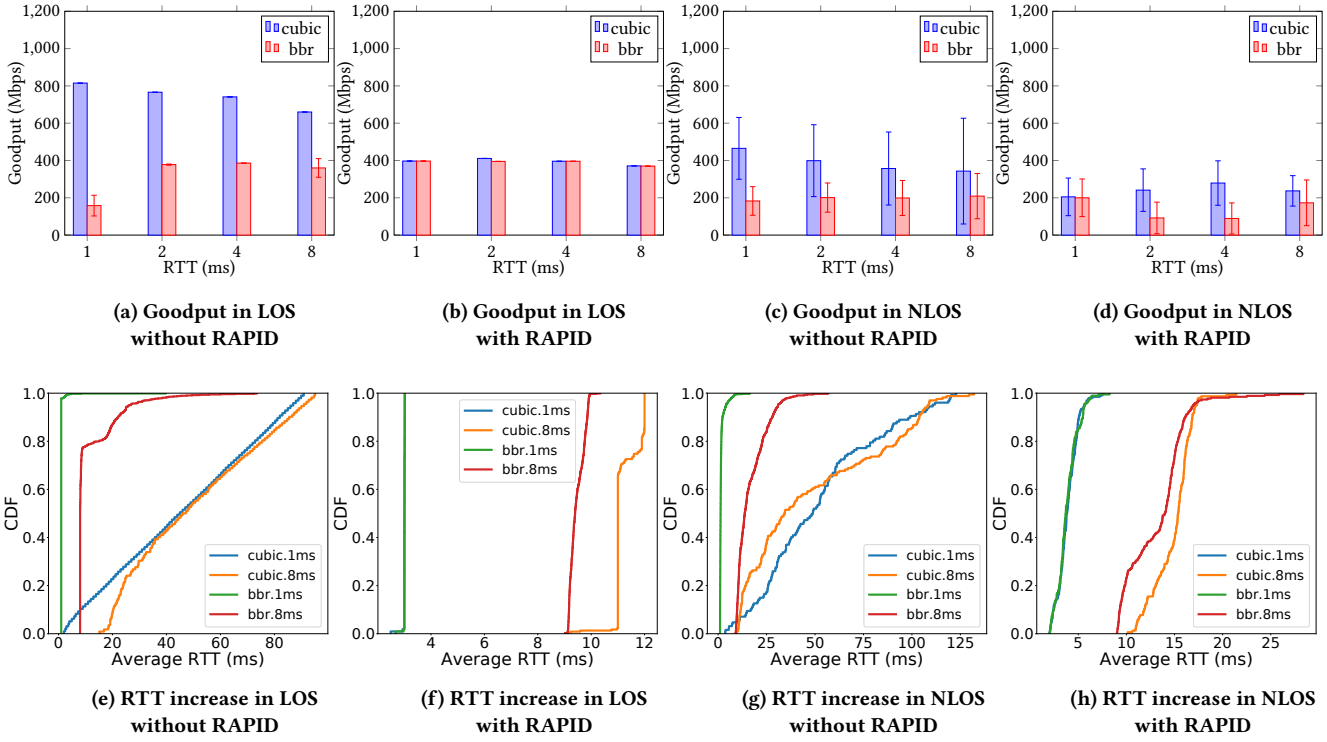
In this section, we first describe the implementation of our proxy in ns-3 [13]. Then, we evaluate its efficiency under some relevant scenarios that reproduce the issues mentioned in Sections 1 and 2. We repeat each scenario 10 times and show the observed results with a 95% confidence interval.

4.1 ns-3 experimentation testbed

The ns-3 mmWave module paves the way for the simulation of end-to-end 5G environments. Besides providing large bandwidths in the millimeter wave band for eMBB scenarios, it also enables the simulation of low-latency communications thanks to a customizable sub-carrier spacing (i.e., supporting various numerologies) in the RAN. With that in mind and in order to evaluate the performance of RAPID, we implement all the functional modules illustrated in Figure 2 in a proxy device collocated with a mmWave base station in ns-3. The proxy is connected to the core network via a 10 *Gbps* link in order to simulate a real-world fiber backhaul. Since Cubic and BBR are the dominant CCAs today [22], we simulate two TCP sources based on these algorithms as illustrated in Figure 3. Other simulation parameters are detailed in Tables 1 and 2.

4.2 MEC Scenarios

In this paper, we focus on evaluating the performance of RAPID in mobile edge settings since it is more suitable for throughput-intensive ultra-low latency applications (e.g., AR/VR, tactile Internet, etc.) that require both high throughput (e.g., from 100 *Mbps* to a few *Gbps*) and low latency (e.g., from 1 to 10 *ms*) [18]. Furthermore, although several studies have been conducted on the evaluation



(a) Goodput in LOS without RAPID **(b) Goodput in LOS with RAPID** **(c) Goodput in NLOS without RAPID** **(d) Goodput in NLOS with RAPID**

(e) RTT increase in LOS without RAPID **(f) RTT increase in LOS with RAPID** **(g) RTT increase in NLOS without RAPID** **(h) RTT increase in NLOS with RAPID**

Figure 4: Fast eMBB downloads scenario.

Table 1: Global simulation parameters

Parameters	Values
Carrier freq.	28 GHz
Bandwidth	200 MHz
Numerology	3
RLC mode	Ack. (AM)
gNB Height	10 m
UE Height	1.5 m
RLC Buffer	10 MB
MSS	1440 B
Initial Window	10 MSS
RAN Throughput	850 Mbps
Propagation Model	3GPP Umi Street-Canyon PropagationLossModel
3GPP Channel Scenario	Umi Street Canyon

Table 2: Mobility and blockage parameters

Parameters	Values
User speed	1 m/s
Propagation Model	mmWave3GPP Buildings PropagationLossModel
Number of buildings	8
Building sizes[2]	Jean-Medecin Avenue, Nice, FRANCE

of recent CCAs in 4G and 5G mmWave networks, to the best of our knowledge, there is no significant work on their evaluation in mobile edge and very low latency settings. Therefore, in our different experimentation, we consider only end-to-end RTT in the range of 1 to 10 *ms*.

In order to evaluate the performance of RAPID in different RTT and flow configurations, we consider a main single user scenario consisting of one UE receiving simultaneously several flows from different end servers. Depending on the characteristics of the concurrent flows (i.e., sheer download or slow/interactive downloads), such a basic scenario can highlight both the self-inflicted bufferbloat issue introduced by loss-based CCAs and the unfairness in terms of bandwidth when different CCAs compete in deep buffer environments. Based on our main scenario, we define two sub-scenarios that highlight the impact of RAPID's flow-level bandwidth allocation scheme on the global achievable performance in terms of goodput and delay:

Fast eMBB downloads: This first scenario aims to showcase the efficiency of RAPID in mitigating the interference between different CCAs (i.e., Cubic and BBR). To that end, the UE downloads simultaneously 200 *MB* from two TCP servers, one using Cubic and

the other BBR, in both LOS and NLOS conditions. The obtained goodput and the delay variations are observed for each flow in various end-to-end RTT configurations.

Mixed Fast and App-limited/Slow downloads: In this scenario, we highlight the negative effects of self-inflicted bufferbloat on app-limited flows (i.e., slow flows). The UE receives simultaneously a continuous flow (e.g., streaming) that uses Cubic and a slow flow (e.g., web browsing) that uses BBR. The slow flow is paced at 16 *Mbps* in order to simulate a continuous web browsing activity with 2 *MB* page sizes[3] and a Page Load Time (PLT) of 1 second, which corresponds to the limit of user's real-time perception [24]. We then evaluate the effect of bufferbloat on web browsing for different RTT values and radio conditions by monitoring the bandwidth share of the slow flow, which should remain around 16 *Mbps* for a reasonable real-time perception.

Note that the NLOS conditions during our experiments are simulated by reproducing the positions, sizes and heights of the buildings in Jean Medecin Avenue (a famous busy avenue in Nice, France).

4.3 Results

Figures 4a, 4c, 4e, 4g show a comparison of both goodput and end-end latency when RAPID is not used in the fast eMBB downloads scenario. At first, it can be seen that Cubic grabs more bandwidth than BBR in all the evaluated configurations regardless of the radio conditions. This outcome was indeed expected since several studies show that loss-based CCAs outperform BBR in terms of goodput in deep buffer environments [4, 14, 29]. Although most of these studies claim that the degree of fairness in this situation only depends on the bottleneck buffer size, our results show that the end-to-end RTT value greatly affects the bandwidth shares of the two CCAs in mobile edge settings. As illustrated in Figures 4a and 4c, the bandwidth share of BBR appears to increase as the end-to-end RTT increases while Cubic follows an opposite pattern. This phenomenon is simply due to the Hystart and startup phases of Cubic and BBR, respectively. Basically, the two CCAs always start with an exponential growth; however, in case of very small RTT (e.g., 1 *ms*), the BDP of the link is quickly reached. For instance, with the following parameters: initial congestion window = 10 *MSS*; RAN throughput = 850 *Mbps* and RTT = 1 *ms*, the bytes in flight for each flow exceed the BDP of the link just after 4 RTTs (i.e., more than 2 BDPs of inflight data is maintained). However, since BBR's bottleneck bandwidth estimation window is around six to ten RTTs by default [4], it greatly underestimates the available bandwidth due to the large accumulation of data at the RLC buffer introduced by Cubic (more than 5 BDPs of inflight data at 10 RTTs). On the other hand, as the end-to-end RTT increases, it takes more RTTs to reach the BDP of the link, which allows BBR to get a better estimate of the bottleneck bandwidth (i.e., before Cubic introduces large buffering) but causes Cubic to leave the exponential increase phase prematurely. In fact, Cubic's Hystart exits the slow start phase after a certain increase in RTT [11], meanwhile BBR's startup phase continues until it introduces more than two BDPs of inflight data [6]. Due to RTT increase, if BBR does not observe a lower minimum RTT (RTTmin) during a 10 seconds window, it enters a phase called probeRTT. In this phase the number of bytes in flight is reduced to 4 Maximum Segment Size (MSS) at least for 200 *ms* in order to capture the new RTTmin. During this period, Cubic continues to grab more bandwidth as BBR decreases its sending rate, which explains why loss-based CCAs eventually outperforms BBR in deep buffer environments. However, for short to medium flows that finish before BBR's probeRTT phase takes place (i.e., for download duration around 10*s*), as we increase the end-to-end RTT, BBR bandwidth share increases (even in very deep buffer environments) while Cubic share decreases due to the Hystart behavior. It can be seen from Figures 4e and 4g that excessive buffering causes a significant increase in RTT (from 100 to over 5000%) for both BBR and Cubic especially in case of NLOS conditions.

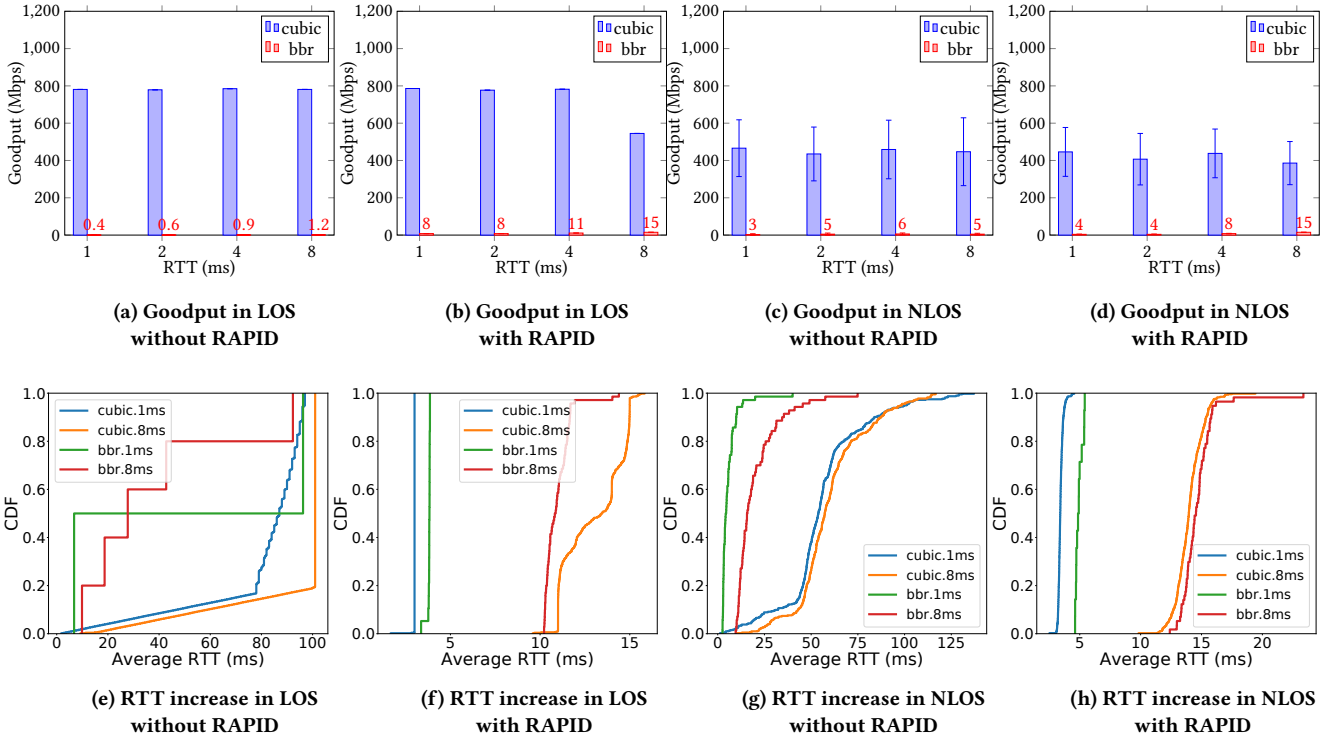
RAPID provides both flows with relatively fair bandwidth shares under LOS conditions as illustrated in Figure 4e and the RTT increase due to Cubic overshooting becomes almost negligible (see Figure 4f) thanks to RAN-aware flow control. However, in case of NLOS conditions, although RAPID reduces the self-inflicted bufferbloat by a factor of 50 as illustrated in Figure 4h, BBR is not able to fully exploit its allocated bandwidth share, as it is subject to throughput oscillations in case of fast variations in delay [18]. Figures 5a through 5h show the test results for the mixed fast/app-limited downloads scenario. In the cases where RAPID is not used (Figures 5a, 5c), it can be seen that the app-limited flow is unable to obtain its desired bandwidth share (i.e., 16 *Mbps*) due to the fast download flow that overshoots more than the BDP. As a result, the slow flow is limited at less than 1 *Mbps* in LOS conditions and around 5 *Mbps* in NLOS. The increase in BBR goodput in case of NLOS is simply due to Cubic's sending rate decrease caused by packet losses. In fact, even after Cubic halves its congestion window, the goodput achieved by the slow flow is still 3 times less than the desired goodput. As a result, the PLT for a 2 *MB* web page is around 3 to 16 seconds. Moreover, as depicted in Figure 5e, over 8000% end-to-end RTT increase can be observed for the slow flow in some cases. Such a large delay increase is unacceptable both for traditional web browsing and throughput-intensive ultra-low latency applications.

On the other hand, in the cases where RAPID is used (Figures 5b and 5d), the app-limited flow is allocated on average 11 *Mbps* in LOS and 8 *Mbps* in NLOS. Furthermore, unlike in the first scenario where RAPID equally shares the available bandwidth between the two download flows, here, the slow flow is automatically detected thanks to Equation 6 and is allocated a Receive Window proportional to its average data rate using Equation 7. As shown in Figures 5b and 5d, this mechanism allows the fast download flow to exploit all the available bandwidth not used by the app-limited flow while remaining bounded by the actual RAN capacity. This not only improves the link utilization but also reduces the RTT increase for both flows by a factor of 10 to 50. Overall, thanks to RAPID flow categorization and bandwidth allocation schemes, the PLT is reduced by 90% (i.e., from 16s to 1.5s).

Besides evaluating RAPID's behavior under basic network conditions, we also consider other parameters that may affect its performance in real-world deployments:

Scalability: In commercial mobile networks, there is no limitations on the number of flows a user can maintain in parallel. Therefore, in order to be efficient in such networks, RAPID must

[2]Google Earth™ is used to estimate the size of buildings.
[3]The average web page size on the Internet is around 1.5 *MB* [17].



(a) Goodput in LOS without RAPID **(b) Goodput in LOS with RAPID** **(c) Goodput in NLOS without RAPID** **(d) Goodput in NLOS with RAPID**

(e) RTT increase in LOS without RAPID **(f) RTT increase in LOS with RAPID** **(g) RTT increase in NLOS without RAPID** **(h) RTT increase in NLOS with RAPID**

Figure 5: Mixed Fast and App-limited/Slow downloads scenario.

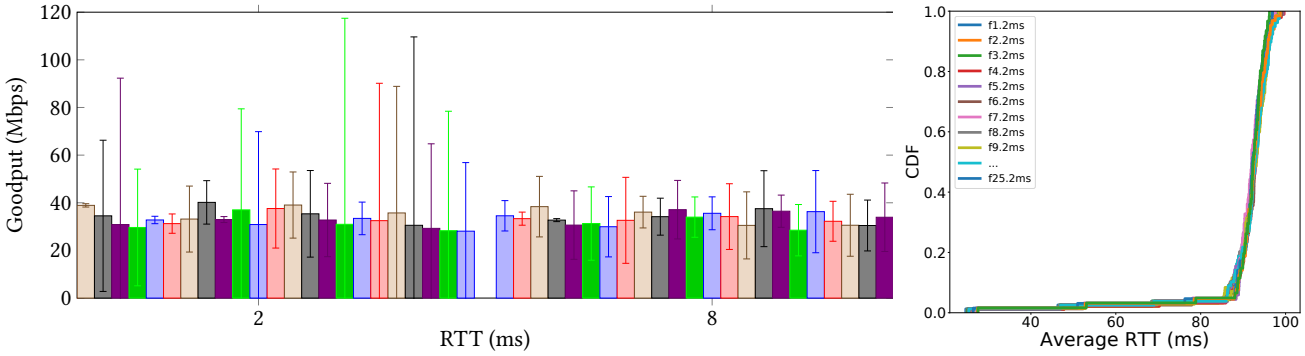
exhibit good performances regardless of the number of active flows per user. Figure 6 illustrates this behavior by showing the bandwidth allocation and delay distribution of 25 Cubic flows progressing in parallel in the same UE for both $2ms$ and $8ms$ end-to-end RTT configurations.

Packet loss: In most cases and as Equation 5 assumes, packet loss in the backhaul network is generally negligible. This owes to the fact that, in commercial deployments, backhaul link dimensioning is done in such a way that the peak data rate or at least the average data rate of the cell is supported [21]. After all, if the backhaul link limits the cell data rate, there would be no point for the operator to invest in large frequency spectrum. On the other hand, packet loss may still occur in the RAN segment because of bad radio conditions or buffer overflow. The latter is avoided when using the RAPID's flow control mechanism while the former is handled by RAPID's simplified CCA which always adapts its sending rate to the available RAN capacity regardless of random losses.

For space limitation reasons, we mainly focus in this paper on scenarios involving Cubic and BBR as they are currently the two main CCAs in use on the Internet. Nevertheless, we have also evaluated RAPID with other well-known CCAs such as NewReno, Vegas [3], Yeah [1] and Westwood [20] that still control a large portion of today's Internet traffic. For instance, a subset of these additional experiments are shown in Figure 7 where we evaluate a simple Fast-Download scenario in LOS condition involving NewReno, Vegas and Westwood in a $1ms$ RTT configuration. As expected, our experiments show that RAPID allows delay-based TCP variants like Vegas (known for their very poor performances when competing with loss-based CCAs) to achieve similar goodput as NewReno or Westwood, while avoiding high delay increase. Figures 7a and 7b show, for a single run, the bytes in flight of the 3 flows as they progress in parallel on the same UE. It can be seen from Figure 7d that RAPID equally shares the total bandwidth (i.e., 850 $Mbps$) between the 3 flows. All the flows get a similar completion time (around $6s$) as illustrated in Figure 7b and the overall delay increase is reduced by a factor of 10 to 90, see Figures 7e and 7f. As illustrated in Figure 7a, without RAPID, NewReno gets by default the lowest completion time (around $2s$) because of its aggressive congestion growth function. In other words, it always maintains a high number of bytes in flight which negatively impacts the other competing flows, and in particular Vegas, as the latter is delay-based. As a result, NewReno always grabs a larger share of the total bandwidth (see Figure 7c) at the cost of over 7000% increase in delay (i.e. from $1ms$ to $80ms$) and with a high degree of unfairness to the other flows. Thanks to RAPID, a certain degree of fairness is maintained between the competing flows. Despite their different designs, all the 3 CCAs are imposed a similar number of bytes in flight, which allows them to obtain the same share and to finish approximately at the same time, as illustrated in Figures 7b and 7d. Note that other additional scenarios involving the aforementioned CCAs as well as the results shown in this section can be reproduced with ns-3 using the codes and scripts that we have made publicly available [8].

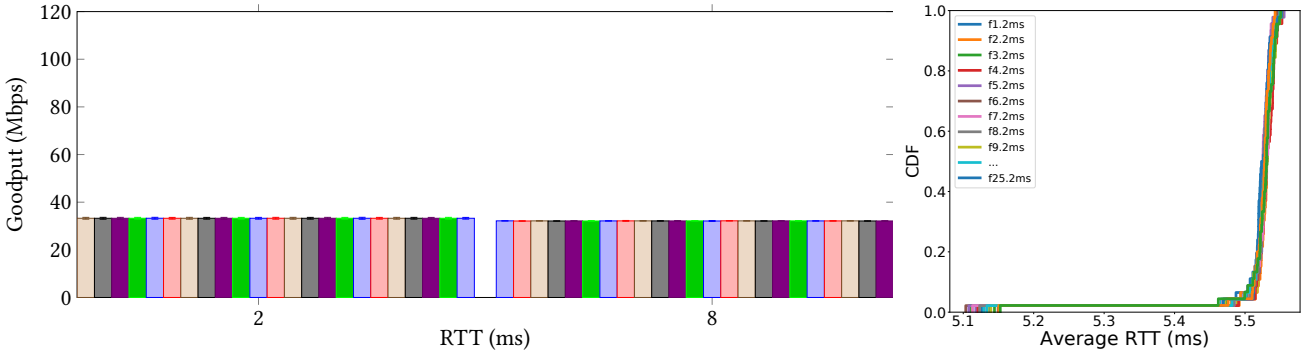
4.4 Overhead

To work properly, RAPID requires an out-of-band control channel in the backhaul as shown in Figure 2. This control channel is



(a) Goodput distribution of 25 Cubic flows without RAPID

(b) Corresponding RTT increase without RAPID



(c) Goodput distribution of 25 Cubic flows using RAPID

(d) Corresponding RTT increase with RAPID

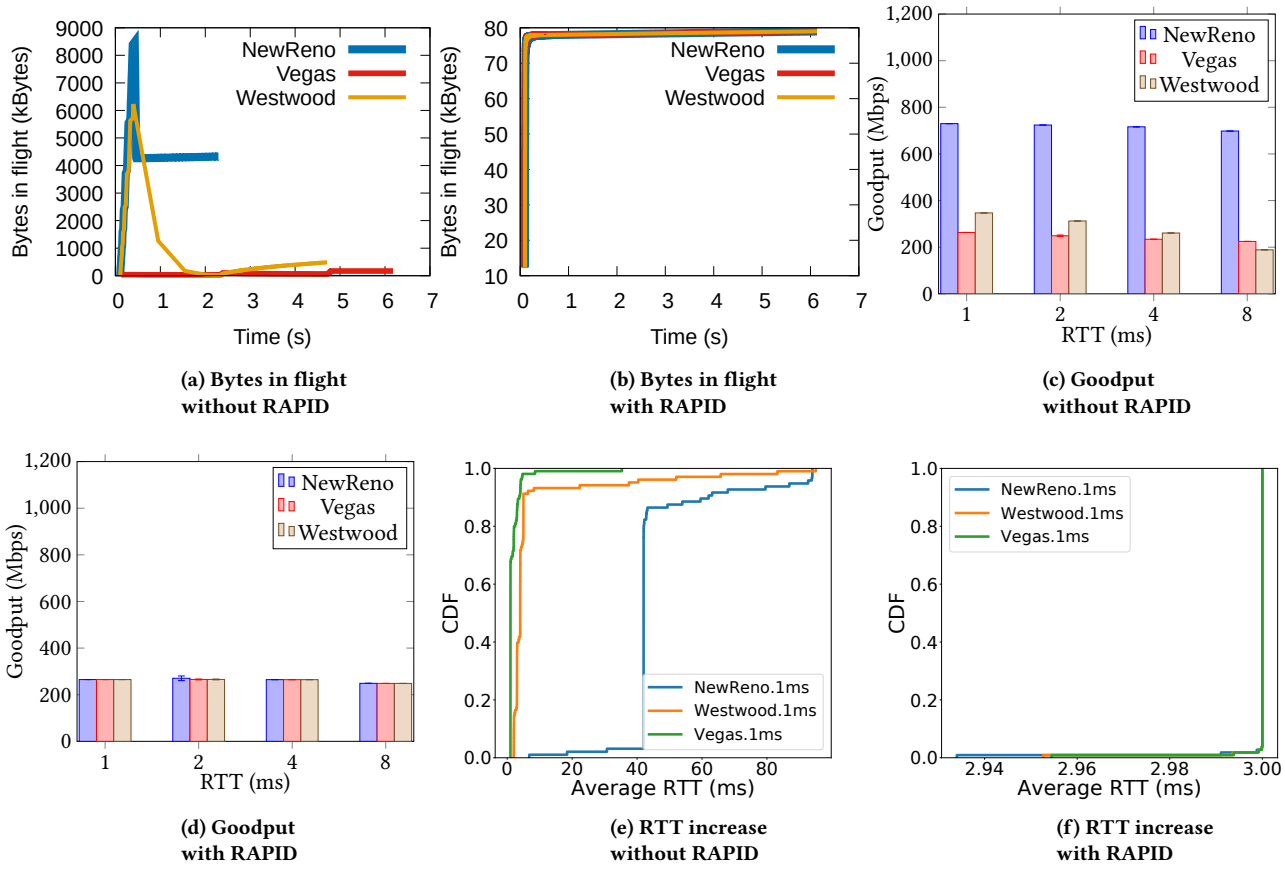
Figure 6: Fast-download with 25 Cubic flows progressing in parallel in the same UE

necessary in order to allow the continuous retrieval of radio information, therefore, its requirements in terms of bandwidth depend both on the size of the retrieved radio information and the configured polling period. In our simulations, we use a polling period at subframe granularity (i.e., radio information is fetched every ms) and the size of the retrieved information per UE including TCP and IP headers is around 170 *bytes*. In other words, each UE introduces an overhead of 1 to 1.5 *Mbps* in the control channel, which is close to the FlexRAN overhead in real world deployments [10]. We believe that such an overhead that accounts only for 0.13% of the achievable throughput (i.e., 850 *Mbps* in our case) and 10^{-8}% of a typical 10 *Gbps* backhaul, is negligible compared to the resulting 150% and 3600% BBR's goodput increase in fast-download and web browsing scenarios, respectively (see Figures 4b and 5b) as well as the overall end-to-end delay reduction in the range of 80 to 97%, see Figures 4h and 5h.

4.5 Discussion

Our simulation results demonstrate that RAPID significantly mitigates the self-inflicted bufferbloat issue in mobile networks, while preserving near optimal link utilization (Figures 4b, 5b). The idea is to impose a certain level of fairness on the concurrent flows depending on their nature and on the available radio bandwidth. However, even if all the flows are assigned the same Receive Window, their performance in terms of goodput and link utilization still depends on the design of their respective CCA. Indeed, depending on their growth functions, different CCAs will reach a given number of bytes in flight at different times. Therefore, CCAs with faster growth functions will yield better link utilization. In addition to that, it is worth noting that upcoming 5G and 6G networks are expected to deliver throughput in the range of several gigabits per second. So, with the design and growth functions of the current CCAs, it would be virtually impossible to reach such a high throughput in a couple of RTTs. Therefore, it becomes necessary to rethink congestion control approaches in order to maintain high link utilization with minimum latency, which is very challenging as indicated in [33], mostly because of the highly intermittent nature of mmWave capacity.

In such a situation, we believe that providing flow-level RTT control and bufferbloat mitigation with proxy-based solutions such as RAPID will significantly simplify the design of future CCAs by allowing them to focus only on fast convergence.



(a) Bytes in flight without RAPID

(b) Bytes in flight with RAPID

(c) Goodput without RAPID

(d) Goodput with RAPID

(e) RTT increase without RAPID

(f) RTT increase with RAPID

Figure 7: Concurrent NewReno, Vegas and Westwood flows

5 RELATED WORK

Authors in [26] proposed milliProxy, a performance enhancing proxy for 5G mmWave scenarios that regulates the sending rate using a variable Flow Window (FW). The FW is computed using the estimated RAN data rate (derived from channel quality information), the RLC buffer occupancy and the estimated RTT. Although milliProxy significantly mitigates bufferbloat in case of single flow, its FW computation scheme is not adapted for multi-flow scenarios.

In-band throughput guidance (TG) was introduced in [15]. This approach allows a functional element (TG provider) residing in the RAN to include RAN throughput information in the TCP header. This throughput information can then be exploited by a TCP server to regulate its sending rate. However, the authors indicate that, in case of multiple flows per bearer, each flow is assigned an equal share of the bearer capacity, which is not sufficient to ensure fairness since flows may have different requirements.

LCTCP proposed in [2] for 5G networks relies on a out-of-band signaling of queue occupancy information between the base station and the end server. The transmitted information is used by the server to adjust its sending rate so that the amount of packets queued at the base station is always aligned with the throughput and delay requirements of the application. However, although LCTCP outperforms conventional CCAs in some scenarios, it cannot deliver both high throughput and low delay simultaneously.

The authors in [19] proposed a TCP proxy for multi-connectivity enabled 5G mmWave network that speeds up congestion window growth and prevents RLC buffer overflow. The former is achieved by separating congestion events of the radio from wired segments while the latter relies on receive window modifications based on an estimation of the available proxy buffer space. Despite improving the download time in some scenarios, the proposed scheme is not adapted for bufferbloat mitigation. Indeed, the proxy buffer size is static and does not consider the actual RAN state, which could result in large delay increase especially in the mmWave band.

A proxy-based flow aggregation TCP proxy for GPRS was proposed in [5]. The solution proposed relies on a static estimation of the bottleneck BDP, which is not adequate for cellular networks because of the highly varying nature of the radio conditions.

6 CONCLUSION

In this paper we highlight the limits of per-user flow isolation in mitigating self-inflicted bufferbloat issues in case of multiple flows per user and intermittent mmWave links. By using up-to-date RAN statistics from MEC Radio Network Information Service and analyzing the behavior of active flows, the RAPID solution

prevents CCAs from exceeding the RAN capacity, thus, drastically reducing the delay increase regardless of the radio conditions while preserving good link utilization.

To the best of our knowledge, RAPID is the first attempt to mitigate self-inflicted bufferbloat and interference between CCAs at the flow level. We plan to further assess its performance over a real 5G deployment in the near future.

REFERENCES

[1] Andrea Baiocchi et al. 2007. YeAH-TCP: yet another highspeed TCP. In *Proc. PFLDnet*, Vol. 7. 37–42.

[2] J. D. Beshay et al. 2017. Link-Coupled TCP for 5G networks. In *IWQoS 2017*. 1–6.

[3] Lawrence S Brakmo et al. 1994. TCP Vegas: New techniques for congestion detection and avoidance. In *ACM SIGCOMM*. 24–35.

[4] Neal Cardwell et al. 2016. BBR: Congestion-Based Congestion Control. *ACM Queue* 14 (2016), 20 – 53.

[5] R. Chakravorty et al. 2005. Using TCP flow-aggregation to enhance data experience of cellular wireless users. *IEEE JSAC* 23, 6 (2005), 1190–1204.

[6] Federico Chiariotti et al. 2021. BBR-S: A Low-Latency BBR Modification for Fast-Varying Connections. *IEEE Access* 9 (2021), 76364–76378.

[7] Mamoutou Diarra et al. 2021. Cross-layer Loss Discrimination Algorithms for MEC in 4G networks. In *Proc. of IEEE HPSR '21*.

[8] Mamoutou Diarra et al. 2021. RAPID. https://github.com/madi223/RAPID

[9] ETSI. 2019. Multi-access Edge Computing (MEC); Radio Network Information API. In *GS MEC 012 V2.1.1*.

[10] Xenofon Foukas et al. 2016. FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks. In *ACM CoNEXT*.

[11] Sangtae Ha et al. 2011. Taming the Elephants: New TCP Slow Start. *Computer Networks* 55, 9 (June 2011), 2092–2110.

[12] Habtegebreil Haile et al. 2021. End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks. *Computer Networks* 186 (2021).

[13] Thomas R Henderson et al. 2008. Network simulations with the ns-3 simulator. In *ACM SIGCOMM demo*, Vol. 14.

[14] Mario Hock et al. 2017. Experimental evaluation of BBR congestion control. In *IEEE ICNP*. 1–10.

[15] Ankur Jain et al. 2015. Mobile throughput guidance inband signaling protocol. *IETF, work in progress*, 1–16.

[16] Haiqing Jiang et al. 2012. Tackling bufferbloat in 3G/4G networks. In *ACM IMC*. 329–342.

[17] Troy Johnson et al. 2014. Desktop and mobile web page comparison: characteristics, trends, and implications. *IEEE Communications Magazine* 52, 9 (2014).

[18] Rajeev Kumar et al. 2019. TCP BBR for Ultra-Low Latency Networking: Challenges, Analysis, and Solutions. In *IFIP Networking*. 1–9.

[19] Goodsol Lee et al. 2019. Simulation study of TCP proxy in multi-connectivity enabled 5G mmWave network. In *ICTC 2019*. 865–869.

[20] Saverio Mascolo et al. 2001. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *ACM MOBICsOM*. New York, NY, USA, 287–297.

[21] Esa Metsälä and Juha Salmelin. 2015. *Planning and Optimizing Mobile Backhaul for LTE*. 129–237. https://doi.org/10.1002/9781118924655.ch5

[22] Ayush Mishra et al. 2019. The great internet TCP congestion control census. *ACM POMACS* 3, 3 (2019), 1–24.

[23] SB Monikandan et al. 2017. A review of MAC scheduling algorithms in LTE system. *Int. J. Adv. Sci. Eng. Inf. Technol* 3 (2017), 1056–1068.

[24] Fiona Fui-Hoon Nah. 2004. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology* 23, 3 (2004), 153–163.

[25] Jukka K. Nurminen. 2010. Parallel Connections and their Effect on the Battery Consumption of a Mobile Phone. In *IEEE CCNC 2010*. 1–5.

[26] M. Polese et al. 2017. milliProxy: A TCP proxy architecture for 5G mmWave cellular systems. In *ACSSC 2017*. 951–957.

[27] R. Robert et al. 2016. Behaviour of Common TCP Variants over LTE. In *GLOBECOM 2016*. 1–7.

[28] D. Sabella et al. 2018. A Hierarchical MEC Architecture: Experimenting the RAVEN Use-Case. In *IEEE VTC Spring*.

[29] Yeong-Jun Song et al. 2020. BBR-CWS: Improving the Inter-Protocol Fairness of BBR. *Electronics* 9, 5 (2020), 862.

[30] Zhen Wang et al. 2011. Why Are Web Browsers Slow on Smartphones?. In *ACM HotMobile*. New York, NY, USA, 91–96.

[31] Yaxiong Xie et al. 2020. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements. In *ACM SIGCOMM*. 451–464.

[32] Menglei Zhang et al. 2016. Transport layer performance in 5G mmWave cellular. In *2016 IEEE INFOCOM WKSHPS*. 730–735.

[33] Menglei Zhang et al. 2019. Will TCP Work in mmWave 5G Cellular Networks? *IEEE Communications Magazine* 57, 1 (2019), 65–71.

[34] Z. Zhong et al. 2018. Performance evaluation of CQIC and TCP BBR in mobile network. In *Proc. of ICIN*. 1–5.